|| Jai Sri Gurudev ||

Sri Adichunchanagiri Shikshana Trust (R)

SJB INSTITUTE OF TECHNOLOGY

NOTES

Subject Name: Natural Language Processing

Subject Code: 18CS743

By

Faculty Name: Chetan R

Designation: Assistant Professor

Semester: 7

Department of Information Science & Engineering

Aca. Year: Odd Sem /2021-22

MODULE-1

Chapter-1 - Introduction

Natural Language Procesing (NLP) :-

Language is the primary means of communication used by humans. It is the tool we use to express the greater part of our ideas and emotions.

It shapes thought, has a structure and carries meaning.

"Natural Language procesing (NLP) is concerned with the development of computational models of aspects of human language procesing."

The two main reasons for development :-

1. Develop automated tools for language procesing.
2. Gain better understanding of human communication.

Building computational models with human language-procesing abilities requires a knowledge of how humans acquire, store and proces language.

CHETAN. R
Asst. Professor

Two major approaches to NLP :-

1. Rationalist approach.
2. Empiricist approach.

Rationalist approach :- Which assumes the existence of some language faculty in human brain. Supporters of this approach argue that it is not possible for children to learn complex thing like natural language from limited sensory inputs.

Empiricists approach :-

They do not believe in the existence of some language faculty. They believe in the existence of some general organization principles such as pattern recognition, generalization and association. Learning of detailed structures can, therefore, take place through the application of these principles on sensory inputs available to the child.

Origins of NLP :-

Natural language processing sometimes termed as natural language understanding - originated from machine translation research.

Natural Language Understanding involves in only interpretation of language.

Natural language processing includes both understanding and generation.

Computational linguistics is similar to theoretical and psycho-linguistics.

Theoretical Linguistics:

They mainly provide structural description of natural language and its semantics. They are not concerned with the actual processing of sentences or generation of sentences from structural description.

They are in a quest for principles that remain common across languages and identify rules that capture linguistic generalization.

Example, most languages have constructs like noun and verb phrases.

Theoretical linguists identify rules that describe and restrict the structure of languages.

CHETAN. R
Asst. Professor

Psycholinguistics:

They explain how humans produce and comprehend natural language. They are interested in the representation of linguistic structures as well as in the process by which these structures are produced. They rely primarily on empirical investigations to back up their theories.

Computational Linguistics :-

It is concerned with the study of language using computational models of linguistic phenomena. It deals with the application of linguistic theories and computational techniques for NLP.

In computational linguistics, representing a language is a major problem; most knowledge representations tackle only a small part of knowledge.

Computational models may be broadly classified under knowledge driven and data-driven categories.

Knowledge driven systems rely on explicitly coded linguistic knowledge, often expressed as a set of handcrafted grammar rules. Acquiring and encoding such knowledge is difficult.

Data driven approaches presume the existence of a large amount of data and usually employ some machine learning technique to learn syntactic patterns. The amount of human effort is less and performance of these systems is dependent on the quantity of the data. These systems are usually adaptive to noisy data.

Language and Knowledge :-

Language is the medium of expression in which knowledge is deciphered. Language, being a medium of expression, is the outer form of the content it expresses. The same content can be expressed in different languages. The meaning of one-language is written in the same language.

CHETAN. R
Asst. Professor

(1) Lexical Analysis :-

It is the simplest level, which involves in Analysis of words. Words are the most fundamental unit of any natural language text. Word level processing requires morphological knowledge. i.e. knowledge about the structure and formation of words from basic units.
The rules for forming words from morphemes are language specific.
This phase scans the source code as a stream of characters and converts it into meaningful lexemes. It divides the whole text into paragraphs, sentences and words.

② Syntactic Analysis :-

It consists of sequence of words as a unit, usually a sentence and finds its structure.

It decomposes a sentence into its constituents and identifies how they relate to each other. It captures grammaticality or non-grammaticality of sentences by looking at constraints like word order, number and case agreement.

This level of processing requires syntactic knowledge, i.e. knowledge about how words are combined to form larger units such as phrases and sentences.

For example, 'I went to the market' is a valid sentence whereas 'went the I market to' is not.

③ Semantic Analysis :-

Semantics is associated with the meaning of the language. It is concerned with creating meaningful representation of linguistic inputs.

The general idea of semantic interpretation is to take natural language sentences and map them onto some representation of meaning.

Defining meaning components is difficult as grammatically valid sentences can be meaningless.

The starting point of semantic analysis, has been lexical semantics. A word can have a number of possible meanings associated with it. But in a given context, only one of there meanings participates. Finding out the correct meaning of a particular use of word is necessary to find meaning of larger units.

CHETAN. R
Asst. Professor

Consider the following sentences:

Kabir and Ayan are married. —— (a)
Kabir and Suha are married. —— (b)

Both sentences have identical structures, and the meanings of individual words are clear. But most of us end up with two different interpretations.

We may interpret the second sentence to mean that Kabir and Saha are married to each other, but this interpretation does not occur for the first sentence. Syntatic structure and compositional semantics fail to explain there interpretations. We make use of pragmatic information.

(4) Discourse Analysis :-

Discourse-level procesing attempts to interpret the structure and meaning of even larger units, e.g. at the paragraph and document level, in terms of words, phrases, clusters

and sentences.

It requires the resolution of anaphoric references and identification of discourse structure. It also requires discourse knowledge, that is, knowledge of how the meaning of a sentence is determined by preceding sentences. In fact, pragmatic knowledge may be needed for resolving anaphoric references.

For example, in the following sentences, resolving the anaphoric reference 'they' requires pragmatic knowledge:

"The district administration refused to give the trade union permission for the meeting because they feared violence." — (a)

"The district administration refused to give the trade union permission for the meeting because they oppose government." — (b)

(5) Pragmatic Analysis :-

This is the highest level of processing which deals with the purposeful use of sentences in situations.

It requires knowledge of the world, i.e. knowledge that extends beyond the contents of the text.

CHETAN. R
Asst. Professor

The Challenges of NLP :-

There are number of factors that make NLP difficult.

① There relate to the problems of representation and interpretation. Language computing requires precise representation of content. Since natural languages are highly ambiguous and vague, achieving such representation can be difficult.

② The inability to capture all the required knowledge is another source of difficulty. It is almost impossible to embody all sources of knowledge that humans use to procen language.

③ The greatest source of difficulty in natural language is identifying its semantics. words alone do not make a sentence. Instead, it is the words as well as their syntactic and semantic relation that give meaning to a sentence.

A language keeps on evolving. New words are added continually and existing words are introduced in new context. For example, most newspapers and TV channels use 9/11 to refer to the terrorist act on the World Trade Centre in the USA in 2004.

④ Idioms, metaphor and ellipses add more complexity to identify the meaning of the written text. As an example consider the sentence:

"The old man finally kicked the bucket".

The meaning of this sentence has nothing to do with the words 'kick' and 'bucket' appearing in it.

⑤ Quantifier-scoping is another problem. The scope of quantifiers (the, each, etc), is often not clear and poses problem in automatic processing.

⑥ The ambiguity of natural languages is another difficulty. Lexical Ambiguity:-

→ The first level of ambiguity arises at the word level. Without much effort, we can identify words that have multiple meanings associated with them.

Example:-

Manya is looking for a match.

In the above example, the word match refers to that either Manya is looking for a partner or Manya is looking for a match. (Cricket or other match)

Solution:-

Part-of-speech tagging and word sense disambiguation.

② Syntactic Ambiguity :-

Syntactic ambiguity exists in the presence of two or more possible meanings within the sentence.

Example:

I saw the girl with the binocular.

In the above example, did I have the binoculaes? or did the girl have the binoculaes?

CHETAN. R
Asst. Professor

③ Refeential Ambiguity :

Refeerential ambiguity exists when you are refeering to something using the pronoun.

Example: Kiran went to Sunita. She said, "I am hungey".

In the above sentence, you do not know that who is hungey either Kiran or Sunita.

Language and Grammar

Automatic proceuing of language requires the rules and exceptions of a language to be explained to the computer. Grammar consists of a set of rules that allow us to parse and generate sentences in a language. Rules relate information to coding devices at the language level not at the world- knowledge level.

The world knowledge affects both the coding and the coding convention blurs the boundary between syntax and semantics.

Types of Grammars

Transformational Grammar (Chomsky 1957)

Lexical functional Grammar (Kaplan and Bresnan 1982)

Government and binding (Chomsky 1981)

Generalized Phare structure Grammar.

Dependency Grammar

Panirian Grammar

Tree-adjoining Grammar (Joshi 1985)

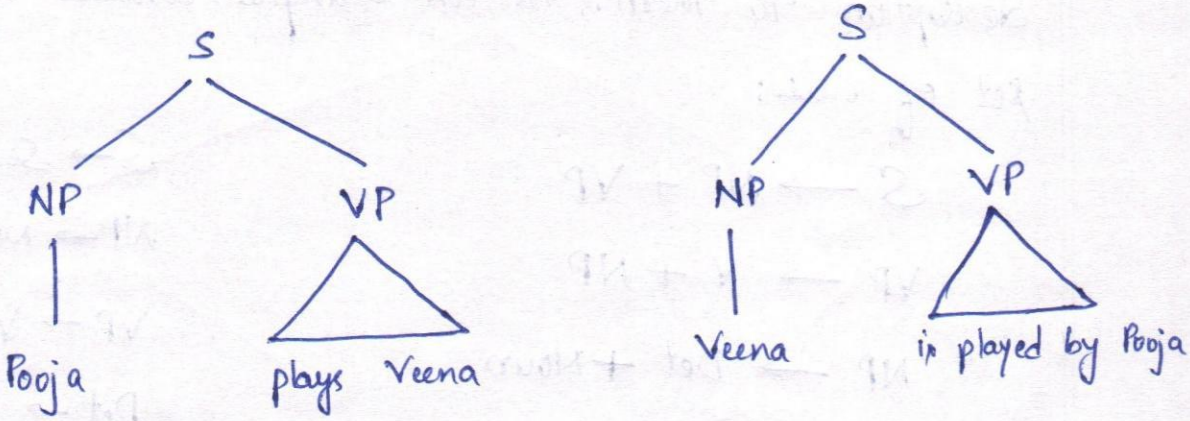
Some of there grammars focus on deeivation and other focus on relationships.

The greatest contribution to grammar comes from Noam Chomsky, who proposed a hierarchy of formal grammar based on level of complexity.

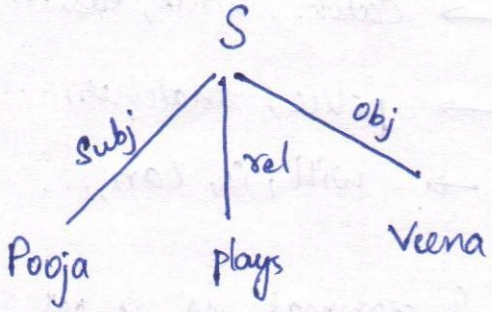
There grammars are phare structure rules.

Generatic grammar basically refers to any grammar that uses a set of rules to specify or generate all and only grammatical sentences in a language.

Transformational grammar, each sentence in a language has two levels of representation, namely, a deep structure and surface structure.



Surface structure



Deep structure

CHETAN. R
Asst. Professor

Transformational grammar has three components:

1. Phrase structure grammar.
2. Transformational rules
3. Morphophonemic rules - These rules match each sentences representation to a string of phonemes.

Each of there components consists of a set of rules. Phrase structure grammar consists of rules that generate natural language sentences and assign a structural description to them. As an example, consider the following set of rules:

$S \longrightarrow NP + VP$

$VP \longrightarrow V + NP$

$NP \longrightarrow Det + Noun$

$V \longrightarrow Aux + Verb$

$Det \longrightarrow$ the, a, an, ...

$Verb \longrightarrow$ catch, write, eat, ...

$Noun \longrightarrow$ police, snatcher, ...

$Aux \longrightarrow$ will, is, can, ...

S → Sentence

NP → Noun Phrase

VP → Verb Phrase

Det → Determiner.

Transformational grammar is a set of transformation rules, which transform one phrase-maker into another phrase-maker. There rules are applied on the terminal string generated by phrase structure rules. Unlike phrase structure rules, transformational rules are heterogeneous and may have more than one symbol on their left hand side. There rules are used to transform one surface representation into another. e.g. active sentence into passive one.

The rule relating active and passive sentences is:

$$NP_1 - Aux - V - NP_2 \rightarrow NP_2 - Aux + be + en - V - by + NP_1$$

Transformational rules can be obligatory or optional. An Obligatory transformation is one that ensures agreement in number of subject and verb.

An optional transformation is one that modifies the structure of a sentence while preserving its meaning.

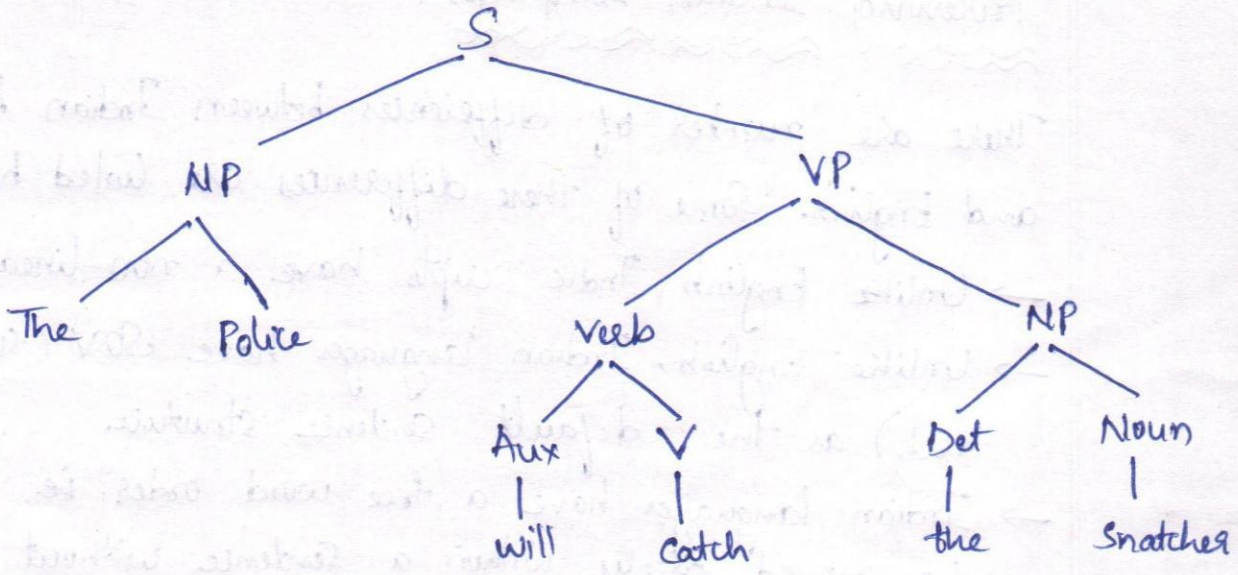
Morphophonemic rules match each sentence representation to a string of phonemes.

CHETAN. R
Asst. Professor

Consider the active sentence:

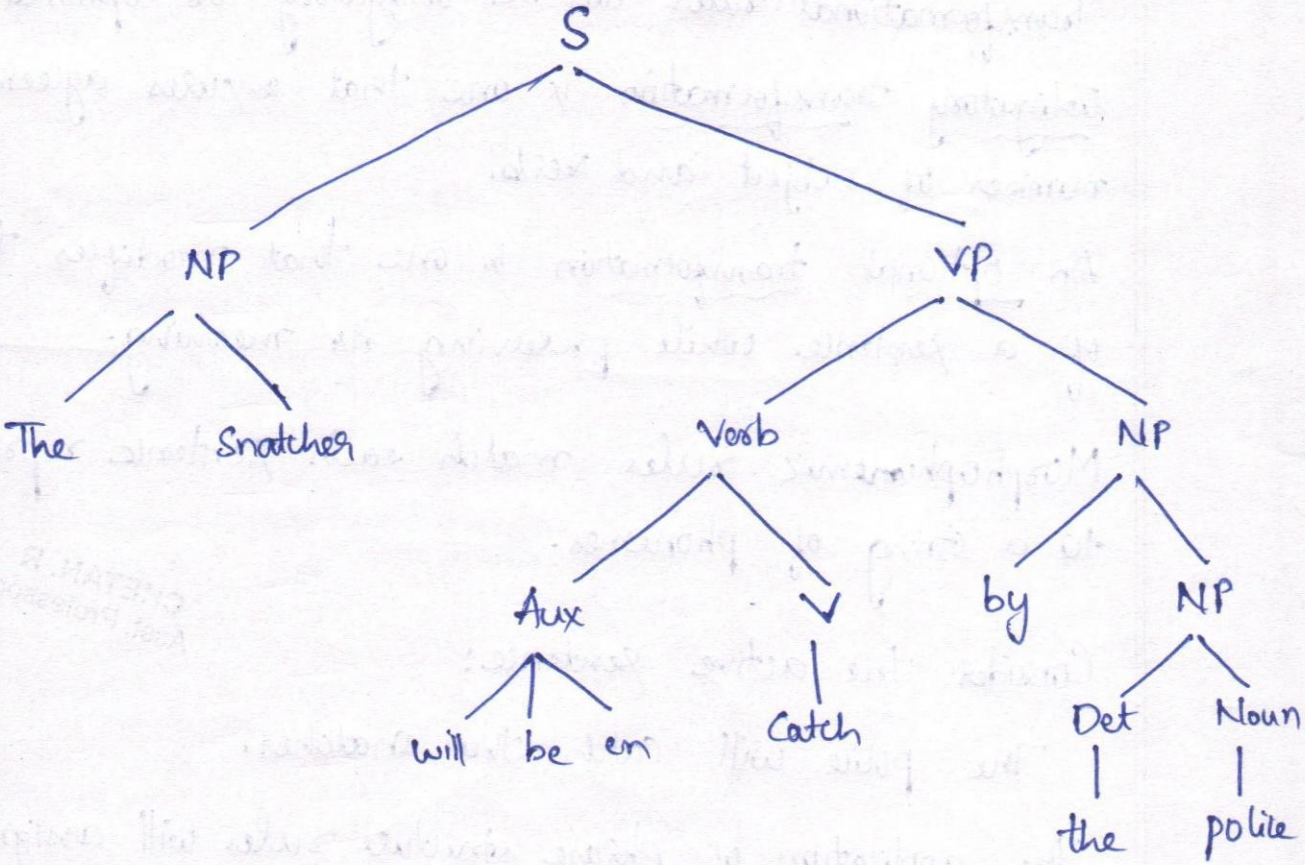
The police will catch the snatcher.

The application of phrase structure rules will assign the structure shown below:



The passive transformation rules will convert the sentence into:

The + culprit + will + be + en + catch + by + police.



Procesing Indian Languages:-

There are number of differences between Indian Languages and English. Some of these differences are listed here:

→ Unlike English, Indic scripts have a non-linear structure.

→ Unlike English, Indian languages have SOV (Subject-Object-Verb) as the default sentence structure.

→ Indian languages have a free word order, i.e., words can be moved freely within a sentence without changing the meaning of the sentence.

→ Spelling standardization is more subtle in Hindi than in English.

→ Indian languages have a relatively rich set of morphological variants.

→ Indian languages make extensive and productive use of complex predicates (CPs).

→ Indian languages use post-position (Karakas) case markers instead of prepositions.

→ Indian languages use verb complexes consisting of sequences of verbs. e.g. गा रहा है and खेल रही है. The auxiliary verbs in this sequence provide information about tense, aspect, modality, etc.

NLP Applications :-

The applications utilizing the NLP include the following:

1. Machine Translation: This refers to automatic translation of text from one human language to another. In order to carry out these translation, it is necessary to have an understanding of words and phrases, grammars of the two languages involved, semantics of the language and world knowledge.

2. Speech Recognition :- This is the process of mapping acoustic speech signals to a set of words. The difficulties arise due to wide variations in the pronunciation of words, homonym and acoustic ambiguities.

3. Speech Synthesis :- It refers to automatic production of speech. Such systems can read out your mails on telephone, or even read out a story books.

4. Natural Language Interfaces to Databases :- It allow quesying a structured database using natural language sentences.

5. Information Retrieval :- This is concerned with identifying documents relevant to a user's query. Indexing, word sense disambiguation, query modification, and knowledge bases have also been used in IR system to enhance performance.

6. Information Extraction :- It captures and outputs factual information contained within a document.

7. Question Answering :- It attempts to find the precise answer, or at least the precise portion of text in which the answer appears.

8. Text Summarization :- This deals with the creation of summaries of documents and involves syntactic, semantic and discourse level procesing of text.

Some Successful Early NLP Systems:-

1. ELIZA (Weizenbaum)

ELIZA is one of the earliest natural language understanding programs. It uses syntactic patterns to mimic human conversation with the user. Here is a sample conversation.

Eliza: Hello, I am Eliza. How may I help you?

User: I am feeling a little bit sleepy.

Eliza: How long have you been feeling a little bit sleepy?

User: For almost half an hour.

Eliza: Please go on.

2. SysTran (System Translation)

The first SysTran machine translation system was developed in 1969 for Russian - English translation. SysTran also provided the first on-line machine translation service called Babel Fish, which is used by AltaVista search engines for handling translation requests from users.

3. TAUM METEO

This is a natural language generation system used in Canada to generate weather reports. It accepts daily weather data and generates weather reports in English and French.

4. SHRDLU (Winograd 1972)

This is a natural language understanding system that simulates actions of a robot in a block world domain. It uses syntactic parsing and semantic reasoning to understand instructions. The user can talk to robot to manipulate blocks, to tell the blocks configurations and to explain its reasoning.

5. LUNAR (Woods 1977)

This was an early question answering system that answered questions about moon rock.

INFORMATION RETRIEVAL :-

The availability of a large amount of text in electronic form has made it extremely difficult to get relevant information. Information retrieval system aims at providing a solution to this.

The term information is being used here to reflect 'subject matter' or the 'content' of some text. The focus is on the communication taking place between human beings as expressed through natural languages. Information is always associated with some data: we are concerned with text only.

The word 'retrieval' refers to operation of accessing information from some computer-based representation. Retrieval of information thus requires information to be processed and stored. Not all the information represented in computable form is retrieved. Instead, only the information relevant to the needs expressed in the form of query is located. Information retrieval is concerned with the organization, storage, retrieval and evaluation of information relevant to the query.

CHETAN. R
Asst. Professor

Information retrieval deals with unstructured data. The retrieval is performed based on the content of the document rather than on its structure. The IR systems usually return a ranked list of documents. The IR components have been traditionally incorporated into different types of information systems including database management systems, bibliographic text retrieval systems, question answering systems and more recently in search engines.

Current approaches for accessing large text collections can be broadly classified into two categories.

(1) Consists of approaches that construct topic hierarchy. e.g. Yahoo. This helps the user locate documents of interest manually by traversing the hierarchy.

(2) Consists of approaches that rank the retrieved documents according to relevance.

Issues in Information Retrieval:-

1. Choose a representation of the document.

Most human knowledge is coded in natural language which is difficult to use as knowledge representation language for computer systems.

Most of the current retrieval models are based on keyword representation. This representation creates problems during retrieval due to polysemy, homonymy and synonymy.

Polysemy: involves the phenomenon of a lexeme with multiple meaning.

Homonymy: is an ambiguity in which words that appear the same have unrelated meanings.

Synonymy : creates problem when a document is indexed with one term and the query contains a different term, and the two term share a common meaning.

Another problem with keyword-based retrieval is that it ignores semantic and contextual information in the retrieval process. This information is lost in the extraction of keywords from the text and cannot be recovered by the retrieval algorithms.

CHETAN. R
Asst. Professor

(2) Inappropriate characterization of queries by the user. The user may fail to include relevant terms in the query or may include irrelevant terms. Inappropriate or inaccurate queries lead to poor retrieval performance.

(3) Matching query representation with that of the document is another issue.

(4) Selection of the appropriate similarity measure is a crucial issue in the design of IR systems.

(5) Evaluating the performance of IR systems is also a major issue. Recall and precision are the most widely used measures of effectiveness.

⑥ The major goal of IR is to search a document in a manner relevant to the query, understanding what constitutes relevance is also an important issue.

⑦ The size of document collections and the varying needs of users also complicate text retrieval.

MODULE - 1

Chapter - 2 LANGUAGE MODELLING

A model is a description of some complex entity or process. A Language model is thus a description of language. Indeed, natural language is a complex entity and in order to process it through a computer-based program, we need to build a representation of it. This is known as Language Modeling.

Language modeling can be viewed either as a problem of grammar inference or a problem of probability estimation.

CHETAN. R
Asst. Professor

There are 2 approaches for Language Modeling:

1. Grammar-based Language model

It uses the grammar of a language to create its model. It attempts to represent the syntactic structure of a language. Grammar consists of hand-coded rules defining the structure and ordering of various constituents appearing in a linguistic unit.
For example, a sentence usually consists of noun phrase and a verb phrase. The grammar based approach attempts to utilize this structure and also the relationships between these structures.

(2) Statistical Language Modelling.

It creates a language model by training it from a corpus. In order to capture regularities of a language, the training corpus needs to be sufficiently large.

"Statistical language modelling is the attempt to capture regularities of natural language for the purpose of improving the performance of various natural language applications."

Statistical language modelling is one of the fundamental tasks in many NLP applications, including speech recognition, spelling correction, handwriting recognition and machine translation.

Various Grammar-Based Language Models

(1.) Generative Grammar

Noam Chomsky in 1957 in his book Syntactic Structures wrote that we can generate sentences in a language if we know a collection of words and rules in that language. Only those sentences that can be generated as per the rules are grammatical. This point of view has dominated computational linguistics and is called generative grammar.

Language is a relation between the sound and its meaning. Thus any model of a language should also deal with the meaning of its sentences.

(2) Hierarchical Grammar

Chomsky described classes of grammars in a hierarchical manner, where the top layer contained the grammars represented by its sub classes.

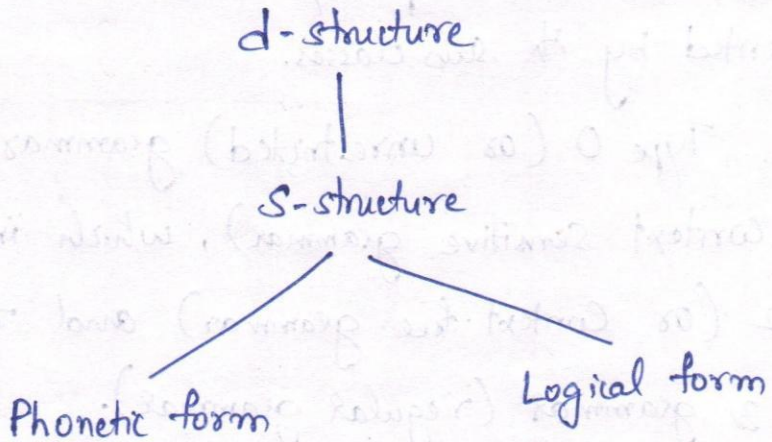
Hence, Type 0 (or unrestricted) grammar contains Type 1 (or context sensitive grammar), which in turn contains Type 2 (or Context-free grammar) and that again contains Type 3 grammar (regular grammar).

(3) Government and Binding

A common viewpoint taken by linguists is that the structure of a language can be understood at the level of its meaning, particularly while resolving structural ambiguity. However, the sentences are given at the syntactical level and the transformation from meaning to syntax or vice versa is not well understood.

Transformational grammars assume two levels of existence of sentences one at the surface level and the other at the deep root level.

Government and Binding theories have renamed them as S-level and d-level and identified two more levels of representation called phonetic form and logical form.



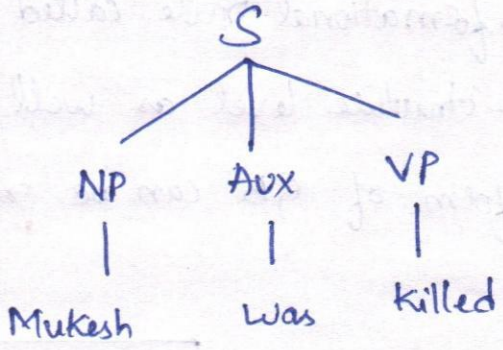
Transformational grammars have hundreds of rewriting rules, which are generally language-specific and also construct-specific. Generation of a complete set of coherent rules may not be possible.

In GB if we define rules for structural units at the deep level, it will be possible to generate any language with fewer rules. There deep-level structures are abstractions of noun-phrase, verb-phrase, etc. common to all languages.

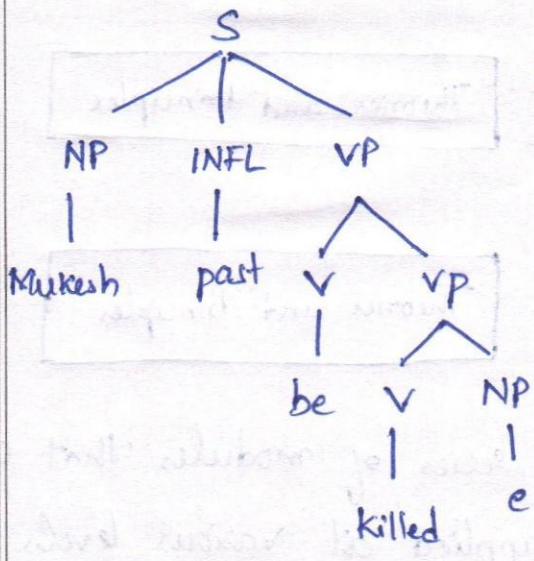
Let us take an example to explain d- and s-structures.

"Mukesh was killed"

(i) In transformational grammar, this can be represented as S-NP Aux VP as given below:



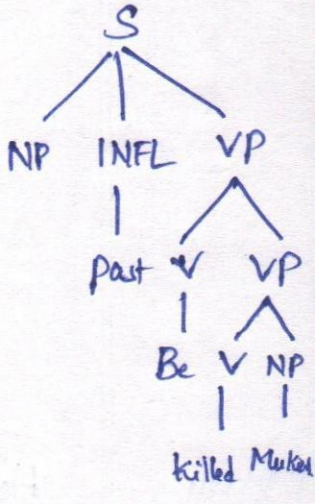
(ii) In GB, the s-structure and d-structure are as follows:



(a) Surface Structure

Mukesh was killed
(e) killed Mukesh
e) past kill Mukesh

INFL: Inflection
NP: Noun phrase
VP: Verb phrase
e: empty.



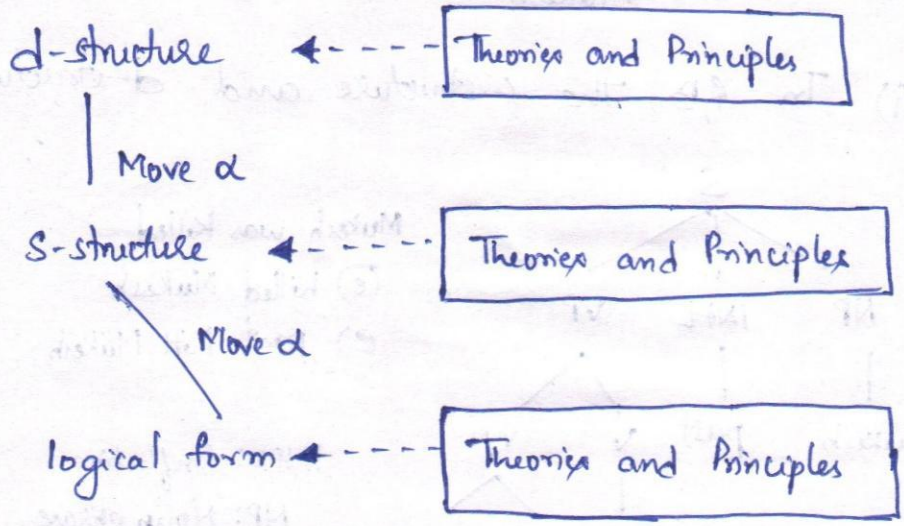
(b) Deep Structure.

Components of GB

Government and binding (GB) comprises a set of theories that map the structures from d-structure to s-structure and to logical form (LF).

A general transformational rule called 'Move α' is applied at d-structure level as well as at s-structure level.

The simplest form of GB can be represented by below figure:



Hence GB consists of 'a series of modules that contain constraints and principles applied at various levels of the representations and transformational rule, Move α.

The GB considers all three levels of representations (d-, s-, and LF) as syntactic and LF is also related to meaning or semantic-interpretive mechanisms.

The GB applies the same Move α transformation to map d-levels to s-levels or s-levels to LF Level. LF level helps in quantifier scoping and also in handling various sentence constructions such as passive or interrogative constructions.

Example:-

"Two countries are visited by most travellers".

Its two possible logical forms are:

LF1: $[_S$ Two countries are visited by $[_{NP}$ most travellers$]]$

LF2: Applying Move α

$[_{NP}$ Most travellers$_i$ $][_S$ two countries are visited by $e_i]$

In LF1, the interpretation is that most travellers visit the same two countries.

In LF2, when we move [most travellers] outside the scope of the sentence, the interpretation can be that most travellers visit two countries, which may be different for different travellers.

One of the important concepts in GB is that of constraints. It is the part of the grammar which prohibits certain combinations and movements; Otherwise Move α can move anything to any possible position. Thus GB. is basically the formulation of theories or principles which create constraints to disallow the construction of ill-formed sentences.

The organization of GB is given below:

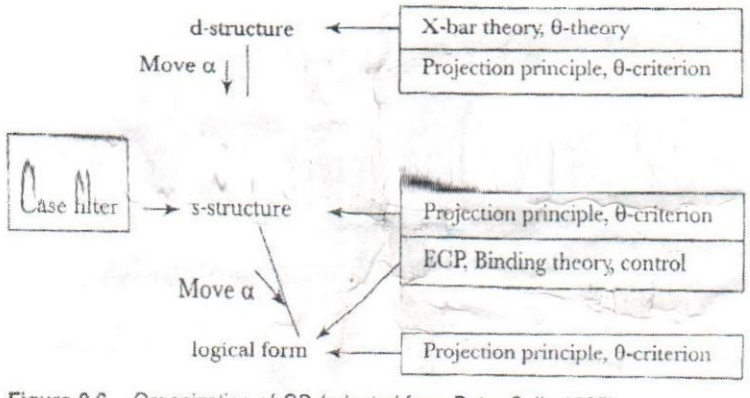


Figure 2.6 Organization of GB (adapted from Peter Sells 1985)

$\bar{X}$ Theory :-

The $\bar{X}$ theory is one of the central concepts in GB. Instead of defining several phrase structures and the sentence structure with separate sets of rules. $\bar{X}$ theory defines them both as maximal projections of some head. Noun Phrase (NP), Verb Phrase (VP), adjective phrase (AP) and prepositional phrase (PP) are maximal projections

of noun (N), verb (V), adjective (A) and preposition (P) respectively and can be represented as head X of their corresponding phrases (where $X = \{N, V, A, P\}$).

Even the sentence structure can be regarded as the maximal projection of inflection (INFL).

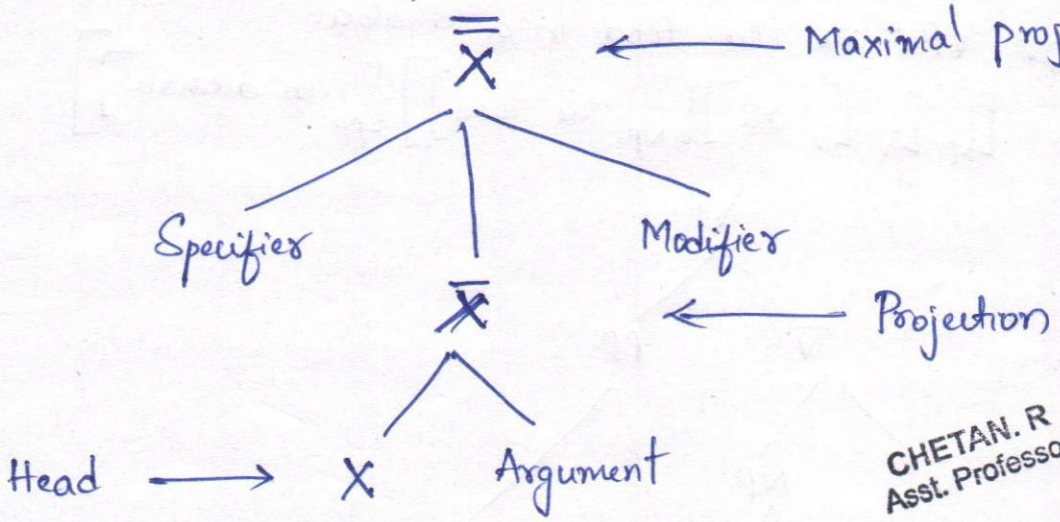
The GB envisages projections at two levels

→ projection of head at semi-phrasal level denoted by $\overline{X}$

→ maximal projection at the phrasal level denoted by $\overline{\overline{X}}$.

The first level projection is denoted as S and second level projection is denoted by S'.

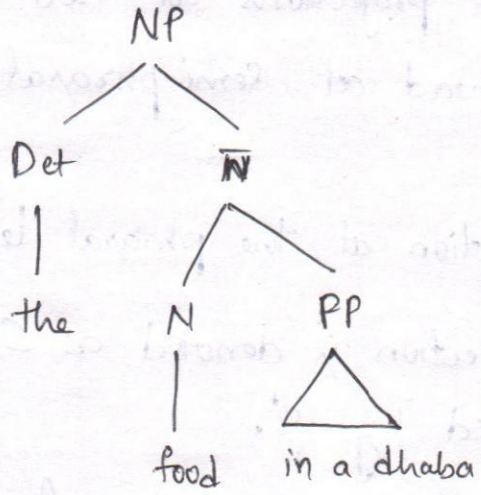
The below figure depicts the general and particular structures with examples.



Next, we consider the representation of the NP, the food in a dhaba. This is followed by the representation of VP, AP and PP structure.

1. NP: the food in a dhaba

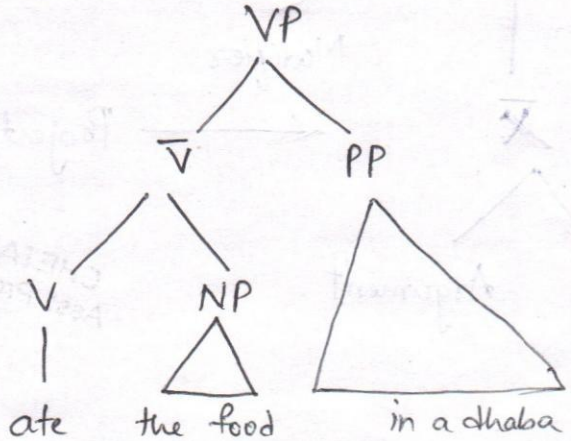
$[_{NP}$ the $[_{N}$ food $]_{pp}$ $[$in a dhabha$]$



NP structure

2. VP: ate the food in a dhaba.

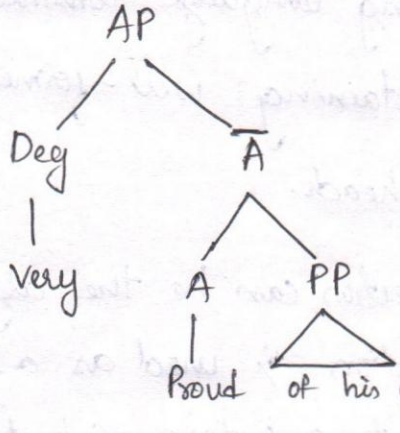
$[_{VP}$ $[_{\bar{V}}$ $[_{V}$ ate$][_{NP}$ the food$]]$ $[_{PP}$ in a dhaba$]]$



VP structure

3. AP: very proud of his country.

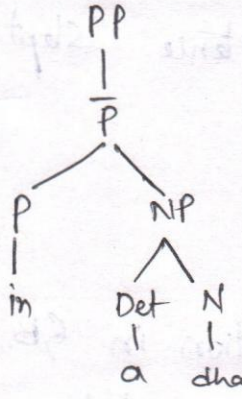
$[_{AP}\ [_{Deg}\ very][_{\bar{A}}\ [_{A}\ proud][_{PP}\ of\ his\ country]]]$



CHETAN. R
Asst. Professor

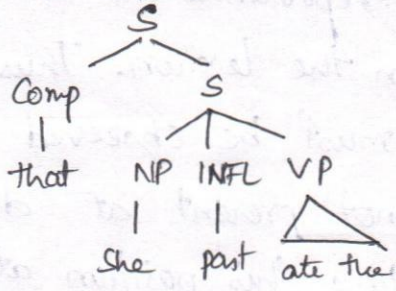
4. PP: in a dhaba

$[_{PP}\ [_{\bar{P}}\ [_{P}\ in][_{NP}\ [_{Det}\ a][_{N}\ dhabha]]]]$



5. S: that she ate the food in a dhaba.

$[_{\bar{S}}\ [_{COMP}\ that][_{S}\ [_{Det}\ She][_{INFL}\ past][_{VP}\ ate\ the\ food\ in\ a\ dhaba]]]$



Sub-categorization :-

GB does not consider traditional phrase structure as an appropriate device for defining language constructs. It places the burden of ascertaining well-formedness to sub-categorization frames of heads.

In principle any maximal projection can be the argument of a head, but subcategorization is used as a filter to permit various heads to select a certain subset of the range of maximal projections.

Example:- (i) the verb "eat" can sub-categorize for NP, whereas the verb 'sleep' cannot.

(ii) "ate food" is well-formed but sentence "slept the bed" is not.

Projection Principle :-

The projection principle is a basic notion in GB, places a constraint on the three syntactic representations and their mapping from one to the other.

The principle states that representations at all syntactic levels are projections from the lexicon. Thus, lexical properties of categorical structure must be observed at each level.

Suppose "the object" is not present at d-level, then another NP cannot take this position at S-level.

Theta-Theory (θ-Theory) or Theory of Thematic Relations

Sub-Categorization only places a restriction on syntactic categories which a head can accept.
GB puts another restriction on the lexical heads through which it assigns certain roles to its arguments. These roles are pre-assigned and cannot be violated at any syntactical level as per the projection principle. These role assignments are called theta-roles and are related to 'Semantic-selection'.

CHETAN. R
Asst. Professor

Theta-role and Theta-criterion :-

There are certain thematic roles from which a head can select. These are called θ-roles and they are mentioned in the lexicon. Example, the verb 'eat' can take arguments with θ-roles "('Agent', 'Theme')".

Agent is a special type of role which can be assigned by a head to outside arguments whereas other roles are assigned within its domain.
Hence in "Mukesh ate food", the verb 'eat' assigns the 'Agent' role to 'Mukesh' and 'Theme' role to 'food'.
Theta-Criterion states that 'each argument bears one and only one θ-role and each θ-role is assigned to one and only one argument'.

C-Command and Government :-

C-command - defines the scope of maximal projection. It is a basic mechanism through which many constraints are defined on Move α.

If any word or phrase (say α or β) falls within the scope of and is determined by a maximal projection, we say that it is dominated by the maximal projection.

If there are two structures α and β related in such a way that 'every maximal projection dominating α dominates β' we say that α C-commands β, and this is the necenary and sufficient condition (iff) for C-commands.

Government

α governs β iff:

α C-commands β

α is an X and every maximal projection dominating β dominates α.

Movement, Empty Category, and Co-indexing :-

In GB, Move α is described as 'move anything anywhere' though it provides restrictions for valid movements.

In GB, the active to passive transformation is the result of NP movement as shown in sentence below: Another well-known movement is the wh-movement, where wh-phrase is moved as follows.

What did Mukesh eat?
[Mukesh INFL eat what]

In the projection principle, lexical categories must exist at all the three levels. This principle, when applied to some cases of movement leads to the existence of an abstract entity called empty category.

In GB, there are 4 types of empty categories, two being empty NP positions called wh-trace and NP trace and the remaining two being pronouns called small 'pro' and big 'PRO'. This division is based on two properties - anaphoric (+a or -a) and pronominal (+p or -p).

Wh-trace	-a, -p
NP-trace	+a, -p
Small 'pro'	-a, +p
big 'PRO'	+a, +p

Co-indexing is the indexing of the subject NP and AGR at d-structure which are preserved by Move α operations at s-structure.

When an NP-movement takes place, a trace of the movement is created by having an indexed empty category (e_i) from the position at which the movement began to the corresponding indexed NP.

For defining constraints to movement, the theory identifies two positions in a sentence. Positions assigned θ-roles are called θ-positions, while others are called $\bar{\theta}$-positions.

Core grammatical positions are called A-positions and the rest are called $\bar{A}$-positions.

Binding Theory

Binding is defined by Sells (1985) as follows:

α binds β iff

α C-commands β, and

α and β are co-indexed

[e_i INFL Kill Mukesh]

[Mukesh$_i$ was killed (by e_i)]

Mukesh was killed.

Empty clause (e_i) and Mukesh (NP_i) are bound. This theory gives a relationship between NPs.

Binding theory can be given as follows:

(a) An anaphor (+a) is bound in its governing category.

(b) A pronominal (+p) is free in its governing category.

(c) An R-expression (-a, -p) is free.

Example:-

A: Mukesh$_i$ knows himself$_i$

B: Mukesh$_i$ believes that Amrita knows him$_i$

C: Mukesh believes that Amrita$_j$ knows Nupur$_k$

Similar rules apply on empty categories also:

NP-trace: +a, -p: Mukesh$_i$ was killed e_i

wh-trace: -a, -p: who$_i$ does he$_i$ like e_i

Empty Category Principle (ECP)

The proper government is defined as:

α properly governs β iff:

α governs β and α is lexical (i.e. N, V, A or P) or

α locally A-binds β

The ECP says 'A trace must be properly governed.'

This principle justifies the creation of empty categories during NP-trace and Wh-trace and also explains the subject/object asymmetries to some extent. As in the following sentences:

(a) What$_i$ do you think that Mukesh ate e_i?

(b) What$_i$ do you think Mukesh ate e_i?

Bounding and Control Theory:

In English, the long distance movement for complement clause can be explained by bounding theory if NP and S are taken to be bounding nodes. The theory says that the application of Move α may not cross more than one bounding node. The theory of control involves syntax, semantics and pragmatics.

Case Theory and Case Filter :-

In GB, Case theory deals with the distribution of NPs and mentions that each NP must be assigned a case. In English, we have the nominative, objective, genitive etc., cases which are assigned to NPs at particular positions. Indian languages are rich in case-markers, which are carried even during movements.

Case Filter :-

An NP is ungrammatical if it has phonetic content or if it is an argument and is not case-marked.

Phonetic content here, refers to some physical realization, as opposed to empty categories. Thus, case filters restrict the movement of NP at a position which has no case assignment. It works in a manner similar to that of the θ-criterion.

GB presents a model of the language which has three levels of Syntactic representations.

→ It assumes phrase structures to be the maximal projection of some lexical head and in a similar fashion, explains the structure of a sentence or a clause.

→ It assigns various types of roles to these structures and allows them a broad kind of movement called Move α.

→ It then defines various types of constraints which restrict certain movements and justifies others.

CHETAN. R
Asst. Professor

(4) Lexical Functional Grammar (LFG) Model

LFG represents sentences at two syntactic levels - Constituent structure (c-structure) and functional structure (f-structure).

Kaplan proposed a concrete form for the register names and values which became the functional structures in LFG. On the other hand, Bresnan was more concerned with the problem of explaining some linguistic issues such as active/passive and dative alternations, in transformational approach. She proposed that such issues can be dealt with by using lexical redundancy rules.

The term 'Lexical Functional' is composed of two terms: the 'functional' part is derived from 'grammatical functions', such as subject and object, or roles played by various arguments in a sentence.
The 'lexical' part is derived from the fact that the lexical rules can be formulated to help define the given structure of a sentence and some of the long distance dependencies, which is difficult in transformational grammars.

C-structure and f-structure in LFG

The C-structure is derived from the usual phrase and sentence structure syntax as in CFG. The grammatical functional role cannot be derived directly from phrase and sentence structure, functional specifications are annotated on the nodes of C-structure, which when applied on sentences, results in f-structure.

Hence f-structure is the final product which encodes the information obtained from phrase and sentence structure rules and functional specifications.

Example :-

"She saw stars in the sky".

CFG rules to handle this sentence are:

$S \longrightarrow NP\ VP$

$VP \longrightarrow V\ \{NP\}\ \{NP\}\ PP^*\ \{S'\}$

$PP \longrightarrow P\ NP$

$NP \longrightarrow Det\ N\ \{PP\}$

$S' \longrightarrow Comp\ S$

where
S: Sentence
P: preposition
S': clause
{}: optional
*: Phrase can appear any number of times including blanks
V: verb
N: noun
Comp: Complement

When annotated with functional specifications, the rules become:

Rule 1: $S \longrightarrow$ NP (↑ subj = ↓) VP (↑ = ↓)

Rule 2: VP → V {NP} (↑ obj = ↓) {NP} (↑ obj2 = ↓) PP* (↑ (↓ case) = ↓) {S'} (↑ comp = ↓)

Rule 3: PP → P NP (↑ obj = ↓)

Rule 4: NP → {Det} N {PP} (↑ Adjunct = ↓)

Rule 5: S' → Comp S (↑ = ↓)

Here ↑ refers to the f-structure of the mother node that is on the left hand side of the rule.

The ↓ symbol refers to the f-structure of the node under which it is denoted.

Hence, in Rule 1, (↑ subj = ↓) indicates that the f-structure of the first NP goes to the f-structure of the subject of the sentence, while (↑ = ↓) indicates that the f-structure of the VP node goes directly to the f-structure of the Sentence. VP.

Consistency:- In a given f-structure, a particular attribute may have at the most one value. Hence, while unifying two f-structures, if the attribute Num has value SG in one and PL in the other, it will be rejected.

Completeness: When an f-structure and all its subsidiary f-structures contain all the functions that their predicates govern, then and only then is the f-structure complete.

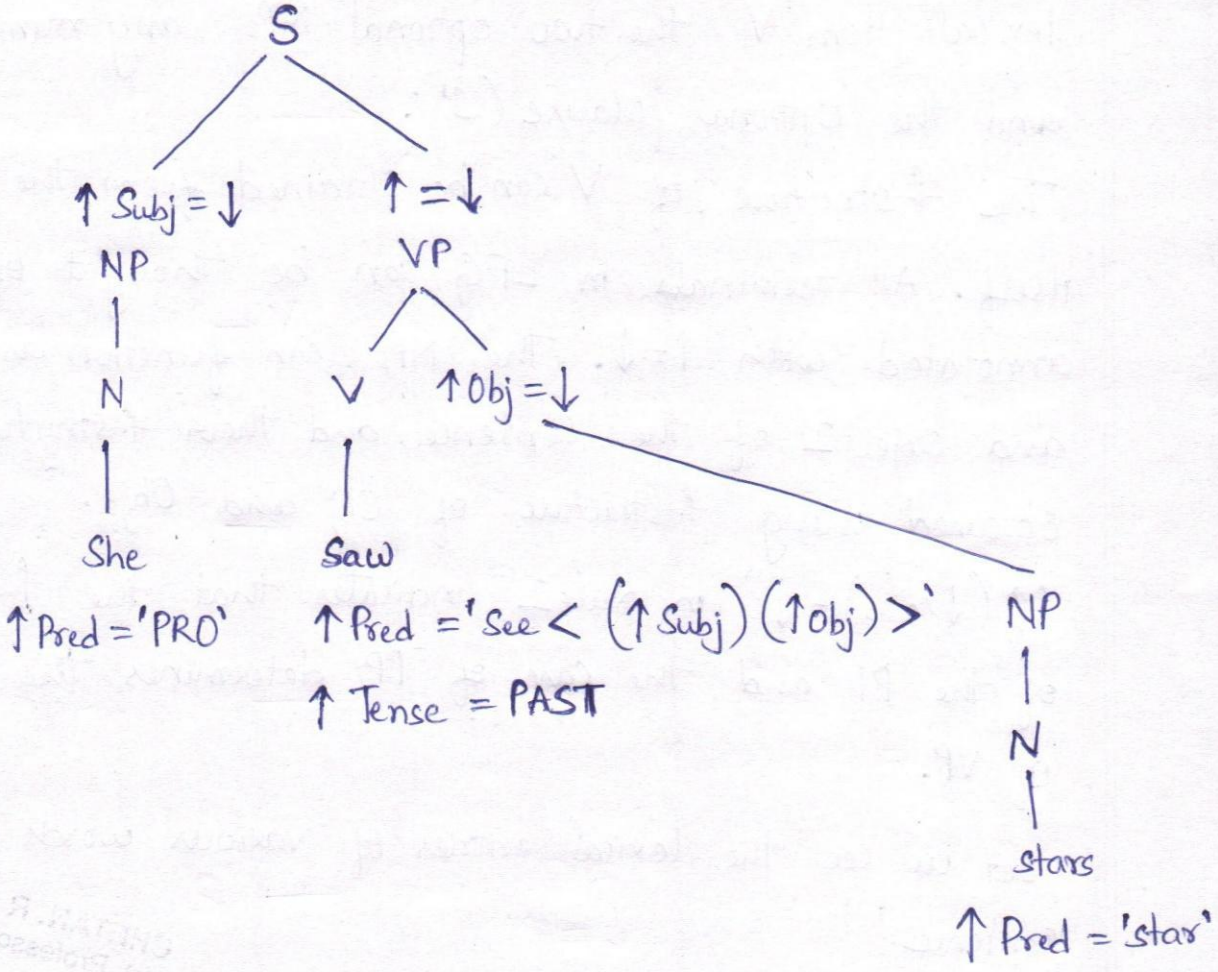
Coherence: Coherence maps the completeness property in the reverse direction. It requires that all governable functions of an f-structure and all its subsidiary f-structures must be governed by their respective predicates.
Hence in f-structure of a sentence, an object cannot be taken if its verb does not allow that object.

Lexical Rules in LFG

Different theories have different kinds of lexical rules and constraints for handling various sentence-constructs.

→ In GB, to express a sentence in its passive form, the verb is changed to its participial form and the ability of the verb to assign case and external θ-role is taken away.

C-structure of sentence is given below:



The f-structure is the set of attribute-value pairs, represented as

$$\begin{bmatrix} \text{Subj} & \begin{bmatrix} \text{Pers} & 3 \\ \text{Num} & \text{SG} \\ \text{Gen} & \text{FEM} \\ \text{Case} & \text{NOM} \\ \text{Pred} & \text{'PRO'} \end{bmatrix} \\ \text{Obj} & \begin{bmatrix} \text{Pers} & 3 \\ \text{Num} & \text{PL} \\ \text{Pred} & \text{'Star'} \end{bmatrix} \\ \text{Pred} & \text{'See'} \langle \uparrow \text{subj}) \ (\uparrow \text{obj}) \rangle \end{bmatrix}$$

Example :-

Active : तारा हँसी
Taaraa hansii
Tara laughed

Causative : मोनिका ने तारा को हँसाया
Monika ne Tara ko hansaayaa
Monika Subj Tara Obj laugh-cause-past
Monika made Tara to laugh.

Active : ↑ Pred = 'Laugh <↑ Subj>'

Causative : ↑ Pred = 'cause <(↑ Subj) (↑ Obj) (Comp)>'

Long Distance Dependencies and Coordination

In GB, when a category moved, it creates an empty category.

In LFG, Unbounded movement and coordination is handled by the functional identity and by correlation with the corresponding f-structure.

Example : Consider the wh-movement in the following sentence.

Which picture does Tara like-most?

The f-structure can be represented as

$$\begin{bmatrix} \text{Focus} & \begin{bmatrix} \text{Obl}_{\text{th}} & \begin{bmatrix} \text{pred} & \text{'PRO'} \\ \text{Refl} & + \end{bmatrix} \\ \text{Pred} & \text{'picture} \langle (\text{Obl}_{\text{th}}) \rangle' \end{bmatrix} \\ \text{Subj} & [\text{Pred} \quad \text{'Tara'}] \\ \text{Obj} & [\quad\quad] \\ \text{Pred} & \text{'like} \langle (\uparrow \text{Subj}) (\uparrow \text{Obj}) \rangle' \end{bmatrix}$$

→ In LFG, the verb is converted to the participial form but the sub-categorization is changed directly.

Consider the following example:

Active: Tara ate the food.

Passive: The food was eaten by Tara.

Active: ↑ Pred = 'eat < (↑ Subj) (↑ Obj) >'

Passive: ↑ Pred = 'eat < (↑ Obl_{ag}) (↑ Subj) >'

Here, Oblag represents Oblique agent phrase. Similar rules can be applied in active and dative constructs for the verbs that accept two objects.

Active: Tara gave a pen to Monika.

Passive: Tara gave Monika a pen.

Active: ↑ Pred = 'give < (↑ Subj) (↑ Obj_2) (↑ Obj) >'

Passive: ↑ Pred = 'give < (↑ Subj) (↑ Obj) (↑ Obl_{go}) >'

Here, Oblgo stands for Oblique goal phrase. Similar rules are also applicable to the procen of causativization. This can be seen in Hindi where the verb form is changed as follows:

हँसना —Causativization→ हँसाना

Laugh — Laugh-cause-past

made to laugh

The auxilary verbs follow the main verb. In Hindi, they remain as separate words, whereas in south Indian languages they combine with the main verb.
For example:

खा रहा है	करता रहा है
Khaa raha hai	Kartaa rahaa hai
eat-ing	doing been has
eating	has been doing

In Hindi, some verbs (main), e.g., give (देना), take (लेना), also combine with other verbs (main) to change the aspect and modality of the verbs.

Example

उसने खाना खाया।	उसने खाना खा लिया।
Usne Khanaa Khaayaa	Usne Khaanaa Kha Liyaa
He (Subj) food ate	He (Subj) food eat taken
He ate food	He ate food
वह चला	वह चल दिया
He moved.	He move given
	He moved.

⑤ Paninian Framework

Paninian grammar was written by Panini in 500 BC in Sanskrit, the framework can be used for other Indian languages and possibly some Asian languages as well.

Unlike English, Asian languages are SOV (Subject-Object-Verb) ordered and inflectionally rich. The inflections provide important syntactic and semantic cues for language analysis and understanding. The Paninian framework takes advantages of there features.

Some Important Features of Indian Languages

Indian languages have traditionally used oral communication for knowledge propagation. In Hindi, we can change the position of subject and object. For example:

(a) माँ बच्चे को खाना देती है।
Maan Bachche ko khanaa detii hai
Mother child to food give-(s)
Mother gives food to the child.

(b) बच्चे को माँ खाना देती है।
Bachche ko Maan khanaa detii hai
Child to mother food give-(s)
Mother gives food to the child.

Vibhakti literally means inflection, it refers to word (noun, verb or other) groups based on either on case endings, or post-positions or compound verbs or main and auxiliary verbs etc. Instead of talking about NP, VP, AP, PP, etc., word groups are formed based on various kinds of markers.

Karaka literally means Case. Paninian Grammar has its own way of defining Karaka relations. These relations are based on the way the word groups participate in the activity denoted by the verb group.

Karaka Theory :-

It is the central theme of PG framework. Karaka relations are assigned based on the roles played by various participants in the main activity. These roles are reflected in the case markers and post-position markers.

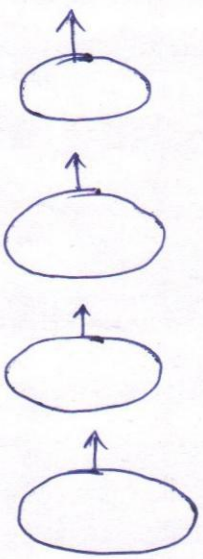
We will discuss the various karakas, such as Karta (subject) Karma (Object), Karana (instrument), Sampradana (beneficiary) Apadan (separation) and Adhikaran (locus).

In Indian languages, the nouns are followed by post-positions instead of prepositions. They generally remain as separate words in Hindi, except in the case of pronouns, for example

रेखा के पिता	उसके पिता
Rekha ke pita	Uske pita
Rekha of father	
Father of Rekha	Her (His) father.

Layered Representation in PG

The GB theory represents three syntactic levels: deep structure, surface structure and logical form, where the LF is nearer to semantics. Paninian grammar framework is said to be syntactico-semantic, that is one can go from surface layer to deep semantics by passing through intermediate layers. The language can be represented as follows:



for example,

माँ ने थाली से खाना उठाकर बच्चे को दिया।

Maan ne thaali se khana uthakar bachche ko diyaa.

The mother gave food to the child taking it up from the plate.

Here thaali is the Apaadan.

'Adhikaran' is the locus of Karta or karma. aangan (courtyard) is the Adhikaran.

Issues in Paninian Grammar

The two problems challenging linguists are:

(i) Computational implementation of PG and

(ii) Adaptation of PG to Indian, and other similar languages.

The approach 'Utsarga-Apvada' where rules are arranged in multiple layers in such a way that each layer consists of rules which are in exception to rules in higher layer. Thus, as we go down the layer, more particular information is derived.

To explain various Karaka relations, let us consider the example.

माँ बच्ची को आँगन में हाथ से रोटी खिलाती है।

Maan bachchi ko aangan mein hath se rotii khilaathi hei

Mother child-to courtyard-in hand-by bread feed (s)

The mother feeds bread to the child by hand in the courtyard.

The first important Karak is subject, called 'Karta' in PG.

Karta is defined as the noun group which is most independent. Karta has generally 'ne' or 'ϕ' case marker.

It is an independent entity in the activity denoted by the main verb. In the above sentence 'maan' (mother) is Karta.

Karma is similar to object and is the locus of the result of the activity. In sentence rotii (bread) is the Karma.

Another Karaka relation is 'Karan' (instrument) which is a noun group through which the goal is achieved. In the sentence haath (hand) is the Karan.

'Sampradan' is the beneficiary of the activity.

e.g. bachchi (child).

'Apaadaan' denotes separation and the marker is attached to the part that serves as a reference point.

A model that limits the history to the previous one word only is termed as bi-gram ($n=1$) model.

A model that conditions the probability of a word to the previous two words, is called a tri-gram model ($n=2$).

bi-gram: $$P(s) \approx \prod_{i=1}^{n} P(w_i / w_{i-1})$$

tri-gram: $$P(s) \approx \prod_{i=1}^{n} P(w_i / w_{i-2}, w_{i-1})$$

As an example, the bi-gram approximation of P(east / The Arabian knights are fairy tales of the) is

P(east/the)

Whereas a tri-gram approximation is

P(east/ of the).

We estimate n-gram parameters using the maximum likelihood estimation technique. We count a particular n-gram in the training corpus and divide it by the sum of all n-grams that share the same prefix.

$$P(w_i / w_{i-n+1}, \ldots, w_{i-1}) = \frac{C(w_{i-n+1}, \ldots, w_{i-1}, w_i)}{\sum_{w} C(w_{i-n+1}, \ldots, w_{i-1}, w)}$$

STATISTICAL LANGUAGE MODEL :-

A statistical language model is a probability distribution $P(s)$ over all possible word sequences.

n-gram model :-

The goal of statistical language model is to estimate the probability of a sentence. This is achieved by decomposing sentence probability into a product of conditional probabilities using the chain rule. as follows:

$$P(s) = P(w_1, w_2, w_3, \ldots, w_n)$$

$$= P(w_1)\, P(w_2/w_1)\, P(w_3/w_1, w_2)\, P(w_4/w_1\, w_2\, w_3) \ldots P(w_n/w_1\, w_2 \ldots w_{n-1}))$$

$$= \prod_{i=1}^{n} P(w_i/h_i)$$ where h_i is history of word w_i defined as $w_1\, w_2 \cdots w_{i-1}$

In order to calculate sentence probability we need to calculate the probability of a word, given the sequence of words preceding it.

An n-gram model simplifies the task by approximating the probability of a word given all the previous words by the conditional probability given previous $n-1$ words only.

$$P(w_i/h_i) = P(w_i / w_{i-n+1} \cdot w_{i-1})$$

Test Sentence(s): The Arabian Knights are the fairy tales of the east.

$P(\text{The}/\langle s\rangle) \times P(\text{Arabian}/\text{the}) \times P(\text{knights}/\text{Arabian}) \times P(\text{are}/\text{knights})$
$\times P(\text{the}/\text{are}) \times P(\text{fairy}/\text{the}) \times P(\text{tales}/\text{fairy}) \times P(\text{of}/\text{tales})$
$\times P(\text{the}/\text{of}) \times P(\text{east}/\text{the})$

$= 0.67 \times 0.5 \times 1.0 \times 1.0 \times 0.5 \times 0.2 \times 1.0 \times 1.0 \times 1.0 \times 0.2$

$= 0.0067$

CHETAN. R
Asst. Professor

The n-gram model suffers from data sparseness problem. An n-gram that does not occur in the training data is assigned zero probability, so that even a large corpus has several zero entries in its bi-gram matrix.
This is because of the assumption that the probability of occurence of a word depends only on the preceding word which is not true in general.

A number of smoothing techniques have been developed to handle the data sparseness problem, the simplest of these being add-on smoothing.

"Smoothing in general refers to the task of re-evaluating zero-probability or low-probability n-grams and assigning them non-zero values."

The sum of all n-grams that share first n-1 words is equal to the count of the common prefix $w_{i-n+1}, \ldots, w_{i-1}$.

So, we rewrite the previous expression as follows:

$$P(w_i / w_{i-n+1}, \ldots, w_{i-1}) = \frac{C(w_{i-n+1}, \ldots, w_{i-1}, w_i)}{C(w_{i-n+1}, \ldots, w_{i-1})}$$

Example :-

Training set:

The Arabian Knights

These are the fairy tales of the east

The stories of the Arabian knights are translated in many languages.

Bi-gram model:

P(the/<s>) = 0.67

P(Arabian/the) = 0.4

P(knights/Arabian) = 1.0

P(are/there) = 1.0

P(tales/fairy) = 1.0

P(east/the) = 0.2

P(are/knights) = 1.0

P(many/in) = 1.0

P(languages/many) = 1.0

P(the/are) = 0.5

P(of/tales) = 1.0

P(stories/the) = 0.2

P(translated/are) = 0.5

P(fairy/the) = 0.2

P(the/of) = 1.0

P(of/stories) = 1.0

P(in/translated) = 1.0

Example, Consider that the number of n-grams that occur 4 times is 25,108 and the number of n-grams that occur 5 times is 20,542. Then the Smoothed count for 5 will be

$$\frac{20542}{25108} \times 5 = 4.09$$

Caching Technique:

Another improvement over basic n-gram model is caching. The frequency of n-gram is not uniform across the text segments or corpus.

Certain words occur more frequently in certain segments and rarely in others.

The cache model combines the most recent n-gram frequency with the standard n-gram model to improve its performance locally.

Add one Smoothing :-

This is the simplest smoothing technique. It adds a value of one to each n-gram frequency before normalizing them into probabilities. Thus the conditional probability becomes:

$$P(w_i / w_{i-n+1}, \ldots, w_{i-1}) = \frac{C(w_{i-n+1}, \ldots, w_{i-1}, w_i)}{C(w_{i-n+1}, \ldots, w_{i-1}) + V}$$

where V is the vocabulary size, i.e., size of the set of all the words being considered.

The add-one smoothing is not considered as good smoothing technique. It assigns the same probability to all missing n-grams, even though some of them could be more intuitively appealing than others.

Good-Turing Smoothing

It adjusts the frequency f of an n-gram using the count of n-grams having a frequency of occurence f+1.

It converts the frequency of an n-gram from f to f* using the following expression:

$$f^* = (f+1)\frac{n_{f+1}}{n_f}$$

where n_f is the number of n-grams that occur exactly f times in the training corpus.

MODULE-2

SYNTACTIC ANALYSIS

CHETAN. R
Asst. Professor

The word "Syntax" refers to the grammatical arrangement of words in a sentence and their relationship with each other.

The objective of syntactic analysis is to find the syntactic structure of the sentence. This structure is usually depicted as a tree. Nodes in the tree represent the phrases and leaves correspond to the words. The root of the tree is the whole sentence.

Identifying the syntactic structure is useful in determining the meaning of the sentence. The identification is done using a process known as parsing.

CONTEXT-FREE GRAMMAR

CFG was first defined for natural language by chomsky and used for the Algol progeamming language by Backus and Naur. It consists of four components:

1. A set of non-terminal symbols, N
2. A set of terminal symbols, T
3. A designated start symbol, S, that is one of the symbols from N.
4. A set of productions, P, of the form:
   $A \longrightarrow \alpha$ where $A \in N$ and α – teeminal & non-terminals

The rule $A \rightarrow \alpha$ says that constituent A can be rewritten as α. This is also called the phrase structure rule. It specifies which elements can occur in a phrase and in what order.

Example, the rule $S \rightarrow NP\ VP$ states that S consists of NP followed by VP.

A language is usually defined through the concept of derivation. The basic operation is that of rewriting a symbol appearing on the left hand side of production by its right hand side.

A CFG can be used to generate a sentence or to assign a structure to a given sentence. When used as a generator, the arrows in the production rule may be read as 'rewrite the symbol on the left with symbols on the right'. Consider the toy grammar shown below:

R1: $S \rightarrow NP\ VP$

R2: $NP \rightarrow N$

R3: $NP \rightarrow Det\ N$

R4: $VP \rightarrow V\ NP$

R5: $VP \rightarrow V$

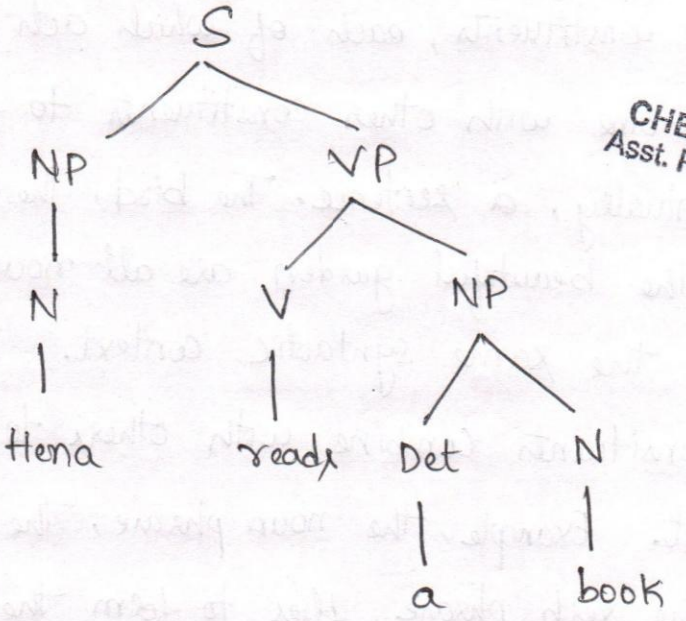
R6: $N \rightarrow$ Hena | She

R7: $V \rightarrow$ reads | sings | sleeps.

CHETAN. R
Asst. Professor

The symbol S can be rewritten as NP VP using Rule 1, then using rules R2 and R4, NP and VP are rewritten as N and V NP respectively. NP is then rewritten as Det N (R3). Finally, using rules R6 and R7, we get the sentence:

"Hena reads a book."



The parse tree for the above representation using bracketed notation is as follows:

$$[_S [_{NP} [_N \text{Hena}]] [_{VP} [_V \text{reads}] [_{NP} [_{Det} \text{a}] [_N \text{book}]]]]$$

CONSTITUENCY

Words in a sentence are not tied together as a sequence of part-of-speech. Language puts constraints on word order. For example, certain words go together with each other more than with others, and seem to behave as a unit. The fundamental idea of syntax is that words group together to form constituents, each of which acts as a single unit. They combine with other constituents to form larger constituents and eventually, a sentence. The bird, The rain, The Wimbledon court, The beautiful garden are all noun phrases that can occur in the same syntactic context.

These consituents combine with others to form a sentence constituent. Example, the noun phrase, The bird, can combine with the verb phrase, flies to form the sentence,

"The bird flies".

Phrase Level Constructions

One of the simplest ways to decide whether a group of words is a phrase, is to see if it can be substituted with some other group of words without changing the meaning. If such a substitution is possible then the set of words forms a phrase. This is called the substitution test.

Consider the sentence:

Hena reads a book.

We can substitute a number of other phrases:

Hena reads a storybook

Those girls read a book

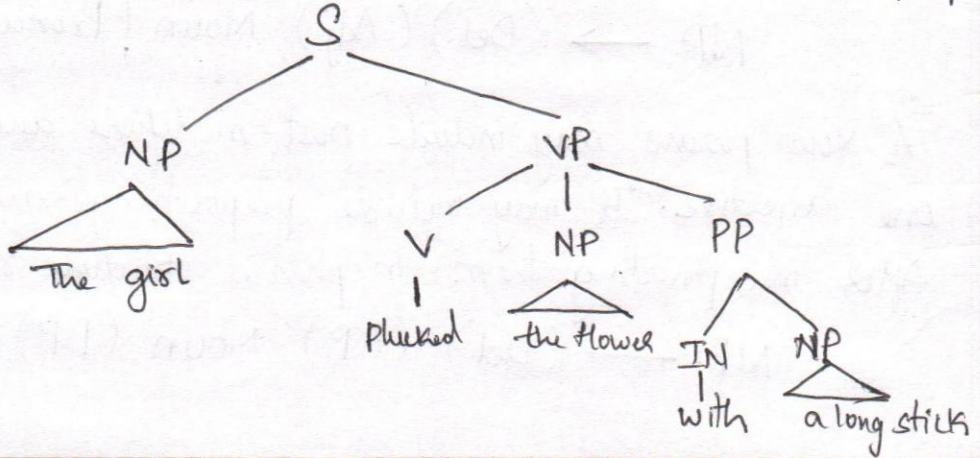
She reads a comic book

CHETAN. R
Asst. Professor

We can easily identify the constituents that can be replaced for each other in these sentences. There are Hena, She and Those girls and a book, a storybook and a comic book.

Phrase types are named after their head, which is the lexical category that determines the properties of the phrase. If the head is a noun, the phrase is called a noun phrase, if the head is a verb, the phrase is called a verb phrase and so on.

Figure shows a sentence with a noun, verb and preposition phrase.



Noun Phrase

A noun phrase is a phrase whose head is a noun or a pronoun, optionally accompanied by a set of modifiers. It can function as subject, object or complement.

The modifiers of a noun phrase can be determiners or adjective phrases.

These structures can be represented using the phrase structure rule of the form $A \rightarrow BC$ where A can be rewritten as two constituents B and C.

Phrase structure rules for a noun phrase are as follows:

NP ⟶ Pronoun

NP ⟶ Det Noun

NP ⟶ Noun

NP ⟶ Adj Noun

NP ⟶ Det Adj Noun

CHETAN. R
Asst. Professor

Single phrase structure rule is as follows:

NP ⟶ (Det) (Adj) Noun | Pronoun

A Noun phrase may include post-modifiers and more than one adjective. It may include prepositional and adjective phrases. After incorporating them in phrase structure rule we get.

NP ⟶ (Det) (AP) Noun (PP).

The following are a few examples of noun phrases:

They - pronoun phrase.

The foggy morning - determiner, an adjective and a noun.

Chilled water - adjective phrase and a noun.

A beautiful lake in Kashmir - determiner, adjective, noun and prepositional phrase.

CHETAN. R
Asst. Professor

Cold banana shake - adjective followed by a sequence of nouns.

A noun sequence is termed as nominal. None of the phrase structure rules are able to handle nominals. So, we modify our rules to cover this situation.

NP → (Det) (AP) Nom (PP)

Nom → Noun | Noun Nom

A noun phrase can act as a subject, an object or a predicate. The following sentences demonstrate each of these uses.

The foggy damped weather disturbed the match. — (a)

I would like a nice cold banana shake. — (b)

Kula botanical garden is a beautiful location. — (c)

In (a) the noun phrase acts as a subject, (b) acts as object and in (c) it is a predicate.

Verb Phrase :-

Verb Phrase is headed by a verb. There is a fairly wide range of phrases that can modify a verb. This makes Verb phrases a bit more complex. The verb phrase organizes various elements of the sentence that depend syntactically on the verb.

CHETAN. R
Asst. Professor

Examples of verb phrases :

Khushbu slept — (a)
The boy kicked the ball — (b)
Khushbu slept in the garden — (c)
The boy gave the girl a book. — (d)
The boy gave the girl a book with blue cover. — (e)

In sentence (a) verb phrase can have a verb.
In (b) verb is followed by an NP.
In (c) verb is followed by a PP.
In (d) verb is followed by two NPs.
In (e) a verb is followed by two NPs and a PP.
In general, the number of NPs in a VP is limited to two, whereas it is possible to add more than two PPs.

$$VP \longrightarrow Verb\ (NP)\ (NP)\ (PP)^*$$

Prepositional Phrase

They are headed by preposition. They consist of a preposition, possibly followed by some other constituent, usually a noun phrase.

"We played volleyball on the beach".

We can have a preposition phrase that consists of just a preposition.

"John went outside".

The phrase structure rule that captures the above eventualities is as follows.

$$PP \longrightarrow Prep\ (NP)$$

CHETAN. R
Asst. Professor

Adjective Phrase :-

They are headed by adjectives. APs consists of an adjective, which may be preceded by an adverb and followed by a PP.

Example:- Ashish is clever.
The train is very late.
My sister is fond of animals.

The phrase structure is:

$$AP \longrightarrow (Adv)\ Adj\ (PP)$$

Adverb Phrase :-

It is headed by an adverb, possibly preceded by a degree of adverb. Ex:-

Time passes very quickly.

AdvP ⟶ (Intens) Adv

CHETAN. R
Asst. Professor

Sentence Level Constructions :-

A sentence can have varying structure. The four commonly known structures are:

(i) Declarative structure:. Sentences have a subject followed by a predicate. The subject of a declarative sentence is a noun phrase and the predicate is a verb phrase.

Ex:- "I Like horse riding".

Phrase structure rule :- S ⟶ NP VP

(ii) Imperative Structure :- Sentences begin with a verb phrase and lack subject. The subject of there types of sentence is implicit and is understood to be 'you'.

There sentences are used for commands and suggestions.

The phrase structure rule :- S ⟶ VP

Ex:- Look at the door.
Give me the book.
Stop talking.

(iii) Yes-no question structure:

Sentences with yes-no question structure ask questions which can be answered using yes or no. There sentences begin with an auxiliary verb, followed by a subject NP, followed by a VP.

Phrase structure rule :- $S \longrightarrow Aux\ NP\ VP$

Ex:- Do you have a red pen?

Is there a vacant quarter?

Is the game over?

Can you show me your album?

(iv) Wh-question structure:-

The sentences with there structures are more complex. There sentences begin with a wh-words - who, which, where, what, why and how. A wh-question may have a wh-phrase as a subject or may include another subject.

Phrase structure rule:- $S \longrightarrow$ Wh-NP VP

or

$S \longrightarrow$ Wh-NP Aux NP VP

Ex:- Which team won the match?

Which cameras can you show me in your shop?

Co-ordination

Coordination refers to conjoining phrases with conjunctions like 'and', 'or', and 'but'.

For example, a coordinated noun phrase can consist of two other noun phrases separated by a conjunction 'and', as in

I ate [NP [NP an apple] and [NP a banana]].

Illy, verb phrases and prepositional phrases can be conjoined as follows:

It is [VP [VP dazzling] and [VP raining]].

Not only that, even a sentence can be conjoined.

[S[S I am reading the book] and [S I am also watching the movie]].

Conjunction rules for NP, VP and S can be built as follows:

NP ⟶ NP and NP

VP ⟶ VP and VP

S ⟶ S and S

CHETAN. R
Asst. Professor

Agreement:-

Most verbs uses two different forms in present tense - one for third person, singular subjects, and the other for all other kinds of subjects.

CHETAN. R
Asst. Professor

The third person singular (3sg) form ends with a -s whereas the non-3sg does not.

Whenever there is a verb that has some noun acting as a subject, this agreement has to be confirmed.

Ex:-

Does [NP Priya] sing? — 4.5a

Do [NP they] eat? — 4.5b

In the first sentence, the subject NP is singular hence, -es form of 'do' is used. The second sentence has a plural NP subject. Hence, the form 'do' is being used. Sentences in which subject and verb do not agree are ungrammatical. Ex:-

[Does] they eat? — 4.5c

[Do] she sings? — 4.5d

This lead to problem known as over-generation.

Rules that handle the yes-no questions of Example 4.5 are as follows: $S \longrightarrow$ Aux NP VP

To take care of the subject-verb agreement, we replace this rule with a pair of rules as follows:

$S \longrightarrow 3sg\ Aux\ 3sg\ NP\ VP$

$S \longrightarrow Non3sg\ Aux\ Non3sg\ NP\ VP$

We could add rules for the lexicon like these:

$3sg\ Aux \longrightarrow does \mid has \mid can$

$Non3sg\ Aux \longrightarrow do \mid have \mid can$

CHETAN. R
Asst. Professor

Illy rules for 3sg NP and Non3sg NP are as follows:

$3sg\ NP \longrightarrow (Det)\ (AP)\ SgNom\ (PP)$

$Non3sg\ NP \longrightarrow (Det)\ (AP)\ PLNom\ (PP)$

$SgNom \longrightarrow SgNoun \mid SgNoun\ SgNom$

$PLNom \longrightarrow PLNoun \mid PLNoun\ PLNom$

$SgNoun \longrightarrow Priya \mid lake \mid banana \mid sister \mid \ldots$

$PLNoun \longrightarrow Children \mid \ldots$

① We solve the problem of over-generation by introducing new grammatical categories corresponding to each such constraint. This result in an explosion in the number of grammar rules and loss of generality.

② An alternative solution is to associate each non-terminal of the grammar with feature structures.

Feature Structures :-

Feature structures are sets of feature-value pairs. They can be used to efficiently capture the properties of grammatical categories.

Features are simply symbols representing properties that we wish to capture.

Ex:- The number property of a noun phrase can be represented as NUMBER feature, and the value it can take is SG (Singular) and PL (Plural).

Values can be either atomic symbols or feature structures.

Feature structures are represented by a matrix-like diagram called attribute value matrix (AVM).

$$\begin{bmatrix} \text{FEATURE}_1 & \text{VALUE}_1 \\ \text{FEATURE}_2 & \text{VALUE}_2 \\ \dots & \dots \\ \text{FEATURE}_n & \text{VALUE}_n \end{bmatrix}$$

CHETAN. R
Asst. Professor

An AVM consisting of a single NUMBER feature with the value SG is represented as follows:

$$\begin{bmatrix} \text{NUMBER} & \text{SG} \end{bmatrix}$$

The value of feature can be left unspecified as shown below:

$$\begin{bmatrix} \text{NUMBER} & [\,] \end{bmatrix}$$

The feature structure can be used to encode the grammatical category of a constituent and the features associated with it.

Ex, the following structure represents the third person singular noun phrase.

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{NUMBER} & \text{SG} \\ \text{PERSON} & 3 \end{bmatrix}$$

Illy, a third person plural noun phrase can be represented as follows:

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{NUMBER} & \text{PL} \\ \text{PERSON} & 3 \end{bmatrix}$$

The values of CAT and PERSON features remain the same in both structures. A feature can have another feature structure as its value. Consider the case of combining the NUMBER and PERSON features into single AGREEMENT. feature

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{AGREEMENT} & \begin{bmatrix} \text{NUMBER} & \text{PL} \\ \text{PERSON} & 3 \end{bmatrix} \end{bmatrix}$$

In order for feature structures to be useful, we must be able to perform operations on them. The two most important operations we need to perform are merging the information content of the two structures that are similar and rejecting structures that are incompatible.

The technique that is used to perform these operations is called unification.

Unification is implemented as a binary operator (⊔) that takes two feature structures as arguments and returns a merged feature structure if they are compatible, Otherwise reports a failure.

1. [NUMBER PL] ⊔ [NUMBER PL] = [NUMBER PL]
2. [NUMBER PL] ⊔ [NUMBER []] = [NUMBER PL]
3. [NUMBER PL] ⊔ [NUMBER SG] Fails.

CHETAN. R
Asst. Professor

PARSING

CHETAN. R
Asst. Professor

A CFG defines the syntax of a language but does not specify how structures are assigned. The task that uses the rewrite rules of a grammar to either generate a particular sequence of words or reconstruct its derivation is termed parsing.

A phrase structure tree constructed from a sentence is called a parse.

The syntactic parser is thus responsible for recognizing a sentence and assigning a syntactic structure to it.

It is possible for many different phrase structure trees to derive the same sequence of words. This means a sentence can have multiple parses. This phenomenon is called syntactic ambiguity.

Garden pathing is another phenomenon related to syntactic parsing. It refers to the process of constructing a parse by exploring the parse tree along different paths, one after the other till, eventually the right one is found.

Finding the right parse can be viewed as a search process. The search finds all trees whose root is the start symbol S and whose leaves cover exactly the word in the input.

The search space in this conception corresponds to all possible parse trees defined by the grammar.

The following constraints guide the search process.

1. Input: The first constraint comes from the words in the input sentence. A valid parse is one that covers all the words in a sentence. Hence, these words must constitute the leaves of the final parse tree.

2. Grammar: The second kind of constraint comes from the grammar. The root of the final parse tree must be the start symbol of the grammar.

These two constraints give rise to the two most widely used search strategies by parsers, namely, top-down or goal-directed search and bottom-up or data-directed search.

CHETAN. R
Asst. Professor

Top-down Parsing :-

The top-down parsing starts its search from the root node S and works downwards towards the leaves.

CHETAN. R
Asst. Professor

The underlying assumption here is that the input can be derived from the designated start symbol, S, of the grammar.

The next step is to find all sub-trees which can start with S. To generate the sub-trees of the second level search, we expand the root node using all the grammar rules with S on their left hand-side. Likewise, each non-terminal symbol in the resulting sub-trees is expanded next using the grammar rules having a matching non-terminal symbol on their left hand side.

The right hand side of the grammar rules provide the nodes to be generated, which are then expanded recursively. As the expansion continues, the tree grows downward and eventually reaches a state where the bottom of the tree consist only of part-of-speech categories.

At this point, all trees whose leaves do not match words in the input sentence are rejected, leaving only trees that represent succesful parses.

A succesful parse corresponds to a tree which matches exactly with the words in the input sentence.

CHETAN. R
Asst. Professor

Sample grammar

S → NP VP
S → VP
NP → Det Nominal
NP → Noun
NP → Det Noun PP
Nominal → Noun
Nominal → Noun Nominal
VP → Verb NP
VP → Verb
PP → Preposition NP
Det → this | that | a | the
Verb → sleeps | sings | open | saw | paint
Preposition → from | with | on | to
Pronoun → she | he | they.

Sentence:- Paint the door

Level I : S

Level II : S → NP VP

Level III:

S → NP VP → Det Nominal

S → NP VP → Pronoun

S → NP VP → Det Noun PP

S → NP VP → Noun

S → VP → Verb NP

S → VP → Verb

S → VP → Verb NP → Paint, Det → the, Nominal → Noun → door

← Correct parse Tree.

Bottom-up Parsing :-

A bottom-up parser starts with the words in the input sentence and attempts to construct a parse tree in an upward direction towards the root.

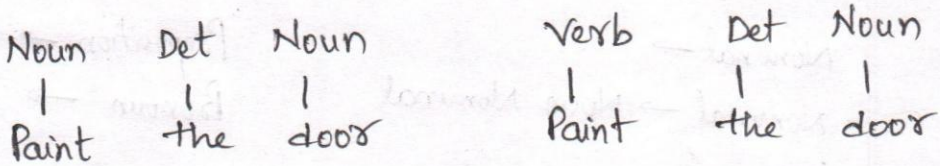
At each step, the parser looks for rules in the grammar where the right hand side matches some of the portions in the parse tree constructed so far, and reduces it using the left hand side of the production.

The parse is considered successful if the parser reduces the tree to the start symbol of the grammar.

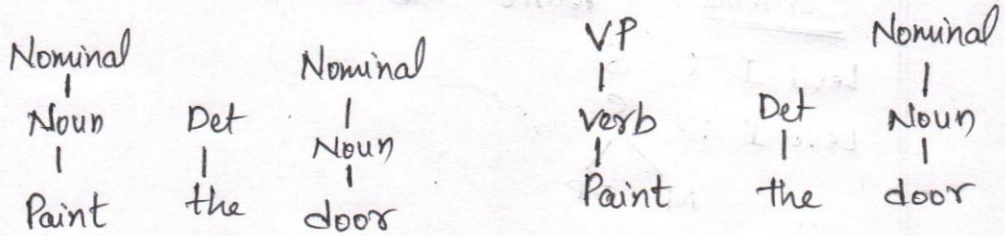
CHETAN. R
Asst. Professor

Level I: Paint the door

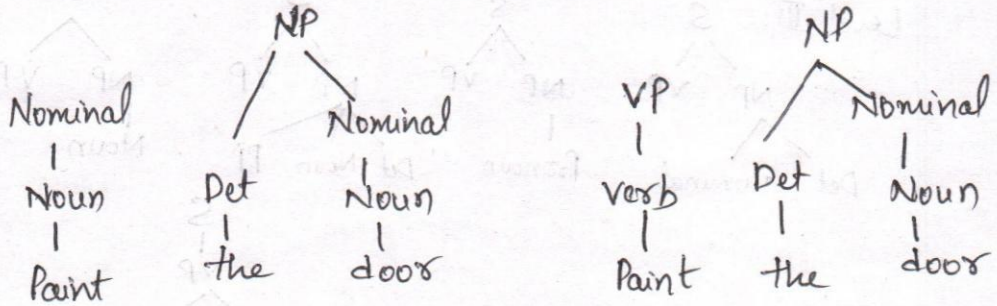
Level II:



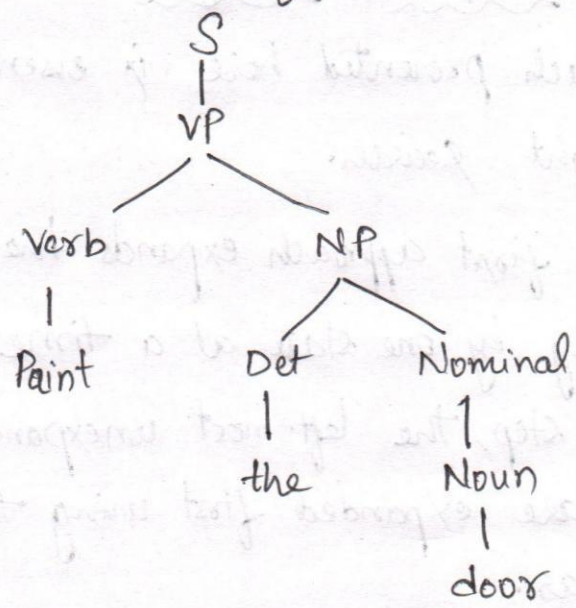
Level III:



Level IV:



The correct parse tree for the sentence:



Advantages and Disadvantages :-

Top down search starts generating trees with start symbol of grammar, it never wastes time exploring a tree leading to a different root.

It wastes considerable time exploring S trees that eventually result in words that are inconsistent with the input.

Bottom-up parser never explores a tree that does not match the input.

It wastes time generating trees that have no chance of leading to an S-rooted tree.

CHETAN. R
Asst. Professor

A Basic Top-Down Parser:

The approach presented here is essentially a depth first, left to right search.

The depth first approach expands the search space incrementally by one state at a time.

At each step, the left-most unexpanded leaf nodes of the tree are expanded first using the relevant rule of the grammar.

The left most node is selected for expansion as it determines the order in which input words needs to be considered.

When a state arrives that is inconsistent with the input the search continues by returning to the most recently generated and unexplored tree.

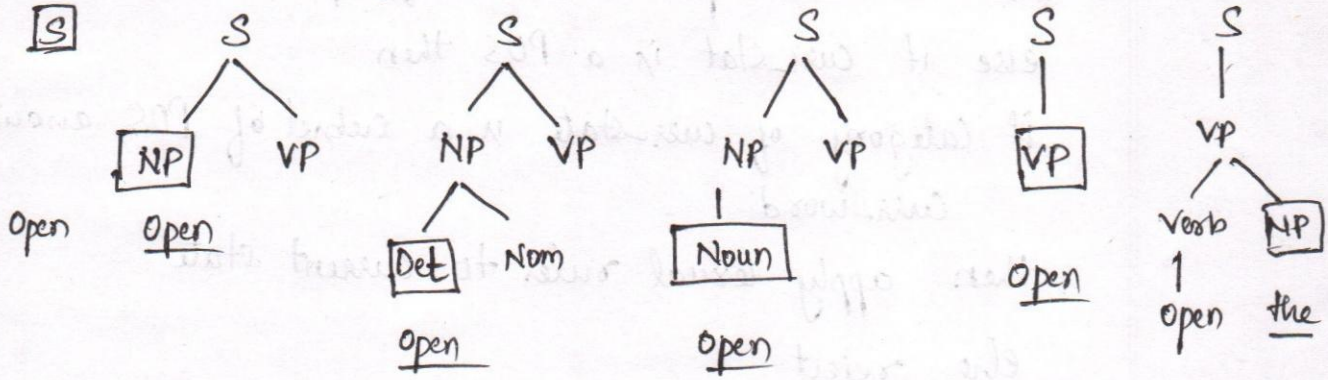
The steps of the algorithm are given below:

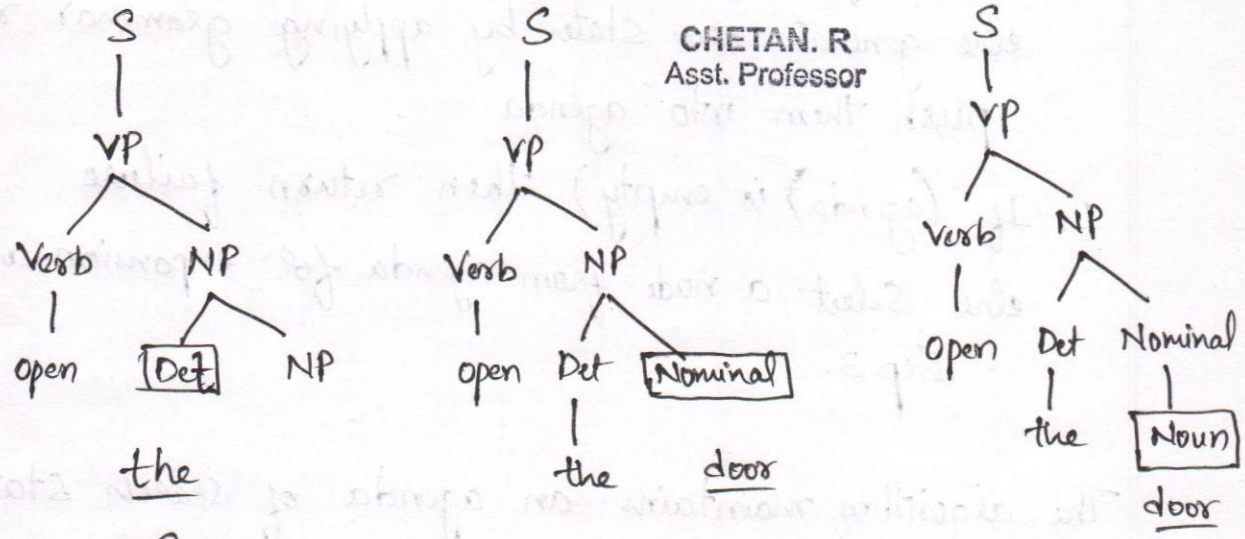
1. Initialize agenda.

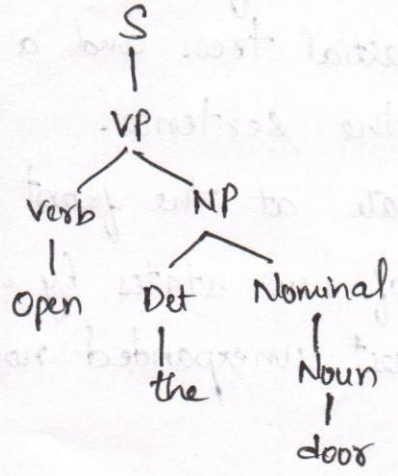
2. Pick a state, let it be curr_state, from agenda.
3. If (curr_state) represents a succesful parse then return parse tree
   else if curr_stat is a POS then
   if category of curr_state is a subset of POS associated with curr_word
   then apply lexical rules to current state
   else reject
   else generate new states by applying grammar rules and push them into agenda
4. If (agenda is empty) then return failure
   else Select a node from agenda for expansion and go to step 3.

The algorithm maintains an agenda of search states. Each search state consists of partial trees and a pointer to the next input word in the sentence.
The algorithm starts with the state at the front of the agenda and generates a set of new states by applying grammar rule to the left-most unexpanded node of the tree associated with it.

The newly generated states are put on the agenda in the order defined by the textual order of the grammar rules used to create them. The process continues until either a successful parse tree is discovered or the agenda is empty, indicating a failure.





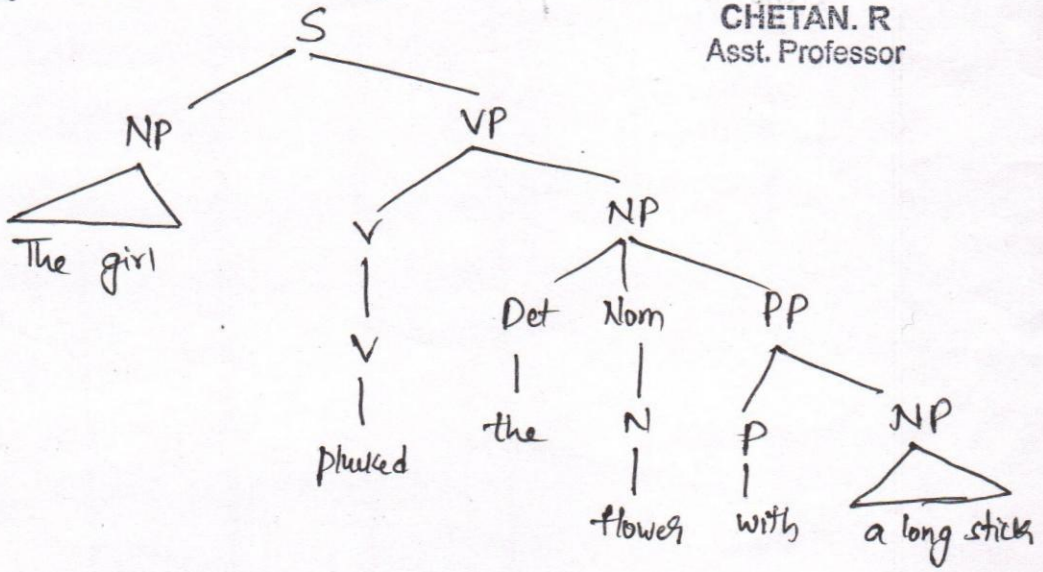


Disadvantages :-

1. Left recursion, which causes the search to get stuck in an infinite loop. This problem arises if the grammar is left recursive that, is, it contains a non-terminal A, which derives, in one or more steps, a string beginning with same non-terminal, i.e., $A^* \Longrightarrow A\beta$ for some β.

2. Structural ambiguity which occurs when a grammar assigns more than one parse to a sentence.
There are two types of structural ambiguity :-

@ Attachment ambiguity - A sentence contains this ambiguity if a constituent fits more than one position in a parse tree.
There are two ways of generating the prepositional phrase with a long stick, in the sentence,
"The girl plucked the flower with a long stick".



(2) Coordination ambiguity occurs when it is not clear which phrases are being combined with a conjunction like and.

Example, The phrase "beautiful hair and eyes" may have the structure [beautiful hair] and [eyes] or [beautiful hair] and [beautiful eyes].

(3) A sentence may have local ambiguity resulting in inefficient parsing. Local ambiguity occurs when certain parts of a sentence are ambiguous.

Example, Paint the door is unambiguous, but during parsing it is not known whether the first word Paint is a verb or a noun.

CHETAN. R
Asst. Professor

(4) Repeated Parsing :- The parser often builds valid trees for portions of the input that it discards during backtracking. These have to be rebuilt during subsequent steps in the parse.

Earley Parser :-

The Earley parser implements an efficient parallel top-down search using dynamic programming. It builds a table of sub-trees for each of the constituents in the input. This way, the algorithm eliminates the repetitive parse of a constituent which arises from backtracking and successfully reduces the exponential-time problem to polynomial time.

The most important component of this algorithm is the Earley chart that has $n+1$ entries, where n is the number of words in the input.

The chart contains a set of states for each word position in the sentence.

The algorithm makes a left to right scan of input to fill the elements in this chart.

It builds a set of states, one for each position in the input string, that describe the condition of the recognition process at that point in the scan.

The states in each entry provide the following information.

1. A sub-tree corresponding to a grammar rule.
2. Information about the progress made in completing the sub-tree.
3. Position of the sub-tree with respect to input.

A state is represented as a dotted rule and a pair of numbers representing starting position and the position of dot.

$$A \longrightarrow X_1 \ldots \bullet C \ldots X_m, [i,j]$$

where the dot (•) represents the position in the rule's right hand side and the two numbers (i and j) represent where the state begins and where the dot lies. A dot at the right end of the rule represents a succesful parse of the associated non-terminal.

Earley Parsing

CHETAN. R
Asst. Professor

Input: Sentence and the Grammar

Output: Chart

```
Chart[0] ← S' → S, [0,0]
n ← length(sentence) //number of words in the sentence
for i=0 to n do
    for each state in chart[i] do
    if (incomplete(state) and next category is not a POS) then
     predictor(state)
    else if (incomplete(state) and next category is a part of speech)
        Scanner(state)
    else
        completer(state)
    end if
  end if
 end for
end for return
```

Procedure predictor $(A \rightarrow X_1 \ldots \bullet B \ldots X_m, [i,j])$

for each rule $(B \rightarrow \alpha)$ in G do

insert the state $B \rightarrow \bullet \alpha, [j,j]$ to chart $[j]$

End

Procedure scanner $(A \rightarrow X_1 \ldots \bullet B \ldots X_m, [i,j])$

If B is one of the part of speech associated with word $[j]$ then

Insert the state $B \rightarrow$ word $[j] \bullet, [j, j+1]$ to chart $[j+1]$

End

Procedure Completer $(A \rightarrow X_1 \ldots \bullet, [j,k])$

for each $B \rightarrow X1 \ldots \bullet A \ldots, [i,j]$ in chart $[j]$ do

insert the state $B \rightarrow X1 \ldots A \bullet \ldots [i,k]$ to chart $[k]$

End

CHETAN. R
Asst. Professor

The algorithm uses three operations to process states in the chart. These are:

→ Predictor

→ Scanner

→ Completer

Predictor :-

The Predictor generates new states representing potential expansion of the non-terminal in the left-most derivation.

It is applied to every state that has a non-terminal to the right of the dot, when the category of that non-terminal is different from the part-of-speech.

The application of this operator results in the creation of as many new states as there are grammar rules for the non-terminal.

These new states are placed into the same chart entry as the generating state.

Their start and end positions are at the point where the generating state ends.

If $A \rightarrow X_1 \ldots \bullet B \ldots X_m, [i,j]$

Then for every rule of the form $B \rightarrow \alpha$, the operation adds to chart $[j]$, the state

$$B \rightarrow \bullet \alpha, [j,j]$$

For example, when the generating state is $S \rightarrow \bullet$ NP VP, $[0,0]$ the predictor adds the following states to chart $[0]$:

NP $\rightarrow \bullet$ Det Nominal, $[0,0]$
NP $\rightarrow \bullet$ Noun, $[0,0]$
NP $\rightarrow \bullet$ Pronoun, $[0,0]$
NP $\rightarrow \bullet$ Det Noun PP, $[0,0]$

CHETAN. R
Asst. Professor

Scanner

A scanner is used when a state has a part-of-speech category to the right of the dot.

It examines the input to see if the part-of-speech appearing to the right of the dot matches one of the part-of-speech associated with the current input.

If yes, then it creates a new state using the rule that follows generation of the input word with this part-of-speech.

It advances the pointer over the predicted input category and adds it to the next chart entry.

If the state is $A \longrightarrow \ldots \bullet a \ldots, [i, j]$ and 'a' is one of the part-of-speech associated with w_j, then it adds a $\longrightarrow \ldots w_j \bullet [i, j]$ to chart $[j+1]$.

Example: When the state NP $\longrightarrow \bullet$ Det Nominal, [0, 0] is processed, the parser finds a part-of-speech category next to the dot.

It checks if the category of the current word (curr_word) matches with the expectation in the current state.

If yes, then it adds the new state Det $\rightarrow$ curr_word $\bullet$, [0, 1] to the next chart entry.

Completer

The completer is used when the dot reaches the right end of the rule. The presence of such a state signifies succenful completion of the parse of some grammatical category.

It identifies all previously generated states that expect this grammatical category at this position in the input and creates new states by advancing the dots over the expected category.

All these newly generated states are inserted in the current chart entry.

If $A \rightarrow \ldots \bullet, [j,k]$, then the computer adds $B \rightarrow \ldots A \bullet \ldots [i,k]$ to chart $[k]$ for all states $B \rightarrow \ldots A \bullet \ldots, [i,j]$ in chart $[j]$.

An item is added to set only if it is not already in the set.

Example :- Let us trace the algorithm using the sentence "Paint the door". The sequence of states created by the parser is shown in figure below:

CHETAN. R
Asst. Professor

Chart [0]	S0	S′ → • S	[0,0]
Dummy		start	state
	S1	S → • NP VP	
	S2	S → • VP	
	S3	NP → • Det Nominal	
	S4	NP → • Noun	
	S5	NP → • Pronoun	
	S6	NP → Det Noun PP	
	S7	VP → • Verb NP	
	S8	VP → • Verb	
Chart [1]	S9	Noun → paint •	[0,1]
	S10	Verb → paint •	[0,1]
	S11	NP → Noun •	[0,1]
	S12	VP → Verb• NP	[0,1]
	S13	VP → Verb •	[0,1]
	S14	S → NP • VP	[0,1]
	S15	NP → • Det Nominal	[1,1]
	S16	NP → • Noun	[1,1]
	S17	S → VP •	[0,1]
	S18	VP → • Verb NP	[1,1]
	S19	VP → • Verb	[1,1]
Chart [2]	S20	Det → the •	[1,2]
	S21	NP → Det • Nominal	[1,2]
	S22	Nominal → • Noun	[2,2]
	S23	Nominal → • Noun Nominal	[2,2]
Chart [3]	S24	Noun → door •	[2,3]
	S25	Nominal → Noun •	[2,3]
	S26	NP → Det Nominal•	[1,3]
	S27	S → NP • VP	[0,3]
	S28	VP → Verb NP •	[0,3]
	S29	VP → • Verb NP	[3,3]
	S30	VP → • Verb	[3,3]
	S31	S → VP •	[0,3]

The CYK Parser :-

The CYK (Cocke - Younger - Kasami) is a dynamic programming parsing algorithm.

It follows a bottom-up approach in parsing. It builds a parse tree incrementally. Each entry in the table is based on previous entries.

The process is iterated until the entire sentence has been parsed.

The CYK algorithm assumes the grammar to be in Chomsky normal form (CNF). A CFG is in CNF if all the rules are of only two forms:

$$A \longrightarrow BC$$

$$A \longrightarrow w, \text{ where } w \text{ is a word}$$

CHETAN. R
Asst. Professor

The algorithm first builds parse trees of length one by considering all rules which could produce words in the sentence being parsed. Then, it constructs the most probable parse for all the constituents of length two. The parse of shorter constituents constructed in earlier iterations can be used in constructing the parse of longer constituents.

The basic CYK algorithm is also a chart-based algorithm. A non-terminal is stored in the $[i, j]$th entry of the chart if and only if, $A \Rightarrow w_i \cdot w_{i+1} \cdots w_{i+j-1}$.

The chart is triangular.

The algorithm builds smaller constituents before attempting to construct larger ones.

First, the terminal derivation rules of the grammar are used to generate the $[i, 1]^{th}$ entries. These entries represent non-terminals that derive the individual words appearing in the sentence, $w_{i1} = w_i$, for $1 \leq i \leq n$, where n is the length of the sentence.

$A \Rightarrow w_{i1}$ if $A \rightarrow w_i$ is a rule in the grammar.

It then continues with sub-string of length two, three and so on.

For every non-terminal A in the grammar, the algorithm determines if $A^* \Rightarrow w_{ij}$.

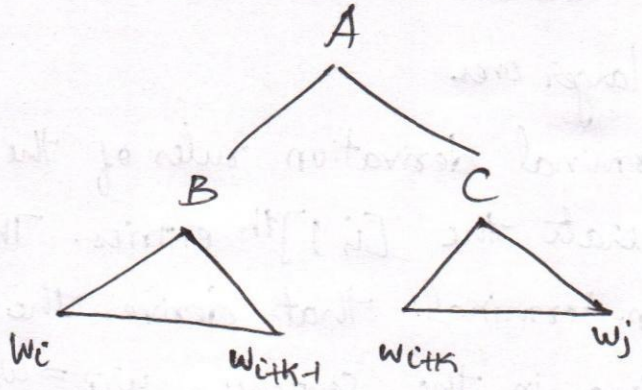
More formally,

$A^* \Rightarrow w_{ij}$ if

1. $A \rightarrow B\ C$ is a rule in grammar
2. $B^* \Rightarrow w_{ik}$ and
3. $C^* \Rightarrow w_{kj}$

For a sub-string w_{ij} of length j starting at i, the algorithm considers all possible ways of breaking it into two parts w_{ik} and w_{kj}.

Finally, since $s = w_{1n}$, we have to verify that $S^* \Rightarrow w_{1n}$, i.e., the start symbol of the grammar derives w_{1n}.



Let $w = w_1\ w_2\ w_3\ w_i \cdots w_j \cdots w_n$

and $w_{ij} = w_i \cdots w_{i+j-1}$

// Initialization step

for $i := 1$ to n do

for all rules $A \rightarrow w_i$ do

chart $[i, 1] = [A]$

// Recursive step

for $j = 2$ to n do

for $i = 1$ to $n - j + 1$ do

begin

chart $[i,j] = \phi$

for $k = 1$ to $j - 1$ do

chart $[i,j]$:= chart $[i,j] \cup \{A \mid A \rightarrow BC$ is a production and $B \in$ chart $[i,k]$ and $C \in$ chart $[i+k, j-k]\}$

end

if $S \in$ chart $[1,n]$ then accept else reject.

To give a better understanding of the whole idea, we work out an example. Consider the following simplified grammar in CNF:

S → NP VP	Verb → wrote
VP → Verb NP	Noun → girl
NP → Det Noun	Noun → essay
Det → an | the	

The sentence to be parsed is: "The girl wrote an essay".

	1	2	3	4	5
1	Det → The	NP → Det Noun			S → NP VP
2	Noun → Girl				
3	Verb → wrote		VP → Verb NP		
4	Det → an	NP → Det Noun			
5	Noun → essay				

CHETAN. R
Asst. Professor

PROBABILISTIC PARSING

A statistical parser works by assigning probabilities to possible parses of a sentence and returning the most likely parse as the final one. More formally, given a grammar G, sentence s and a set of possible parse trees of s which we denote by $\tau(s)$, a probabilistic parser finds the most likely parse φ' of s as follows:

$$\varphi' = \arg\max_{\varphi \in \tau(s)} P(\varphi|s)$$
$$= \arg\max_{\varphi \in \tau(s)} P(\varphi, s)$$
$$= \arg\max_{\varphi \in \tau(s)} P(\varphi)$$

CHETAN. R
Asst. Professor

In order to construct a statistical parser, we have to first find all possible parses of a sentence, then assign probabilities to them, and finally return the most probable parse.

Advantages:

1. Probabilistic parsers offers is removal of ambiguity for parsing.
2. The search becomes more efficient.

A probabilistic context-free grammar (PCFG) is a CFG in which every rule is assigned a probability. It extends the CFG by augmenting each rule $A \rightarrow \alpha$ in set of productions P, with a conditional probability p:

$$A \rightarrow \alpha \ [p]$$

where p gives the probability of expanding a constituent using the rule. $A \rightarrow \alpha$.

A PCFG is defined by the pair (G, f), where G is a CFG and f is a positive function defined over the set of rules such, that, the sum of the probabilities anociated with the rules expanding a particular non-terminal is 1.

$$\sum_{\alpha} f(A \rightarrow \alpha) = 1$$

CHETAN. R
Asst. Professor

An example of PCFG is shown below:

S → NP VP	0.8	Det → that	0.2	Preposition → from	0.3
S → VP	0.2	Det → a	0.25	Preposition → with	0.25
NP → Det Noun	0.4	Det → the	0.35	Preposition → on	0.2
NP → Noun	0.2	Noun → paint	0.25	Preposition → to	0.25
NP → Pronoun	0.2	Noun → door	0.25	Pronoun → she	0.35
NP → Det Noun PP	0.2	Noun → bird	0.25	Pronoun → he	0.35
VP → Verb NP	0.5	Noun → hole	0.25	Pronoun → they	0.25
VP → Verb	0.3	Verb → sleeps	0.2		
VP → VP PP	0.2	Verb → sings	0.2		
PP → Preposition NP	0.1	Verb → open	0.2		
Det → this	0.2	Verb → saw	0.2		
		Verb → paint	0.2		

We can verify that for each non-terminal, the sum of probabilities is 1.

$$f(S \rightarrow NP\ VP) + f(S \rightarrow VP) = 1$$

$$f(NP \rightarrow Det\ Noun) + f(NP \rightarrow Noun) + f(NP \rightarrow Pronoun) + f(NP \rightarrow Det\ Noun\ PP) = 1$$

$$f(VP \rightarrow Verb\ NP) + f(NP \rightarrow Verb) + f(VP \rightarrow VP\ PP) = 1.0$$

$$f(Det \rightarrow this) + f(Det \rightarrow that) + f(Det \rightarrow a) + f(Det \rightarrow the) = 1.0$$

$$f(Noun \rightarrow paint) + f(Noun \rightarrow door) + f(Noun \rightarrow bird) + f(Noun \rightarrow hole) = 1.0$$

CHETAN. R
Asst. Professor

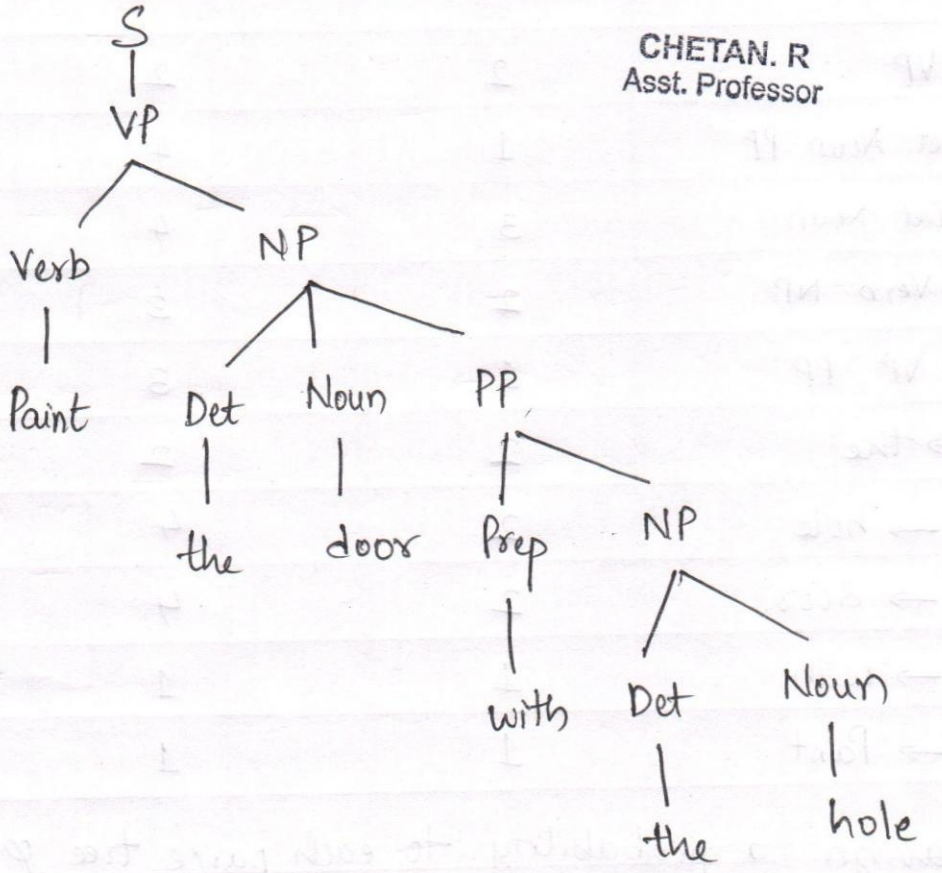
Estimating Rule Probabilities :-

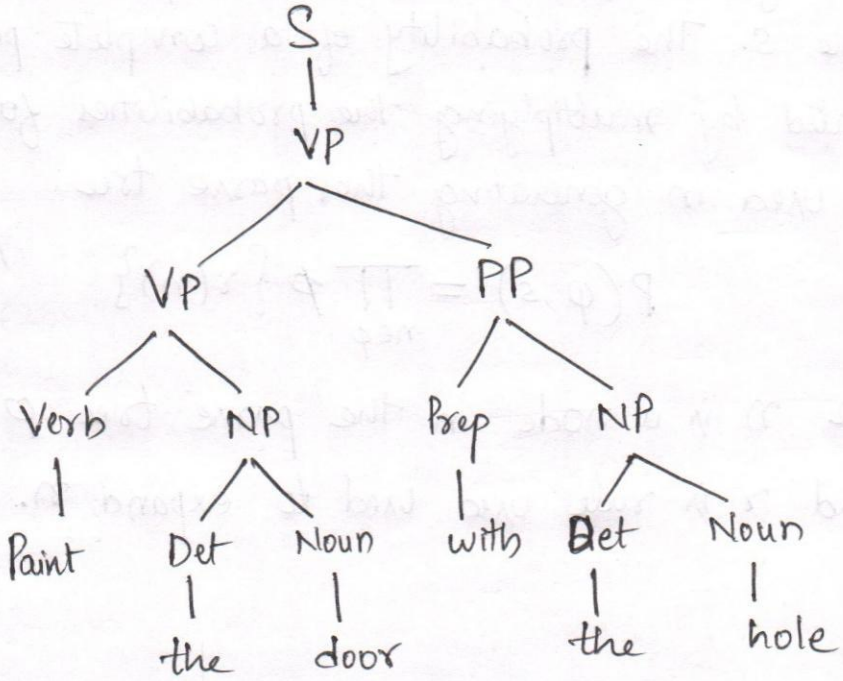
One way to estimate probabilities for a PCFG is to manually construct a corpus of a parse tree for a set of sentences, and then estimate the probabilities of each rule being used by counting them over the corpus. The MLE estimate for a rule $A \rightarrow \alpha$ is given by the expression

$$P_{MLE}(A \rightarrow \alpha) = \frac{Count(A \rightarrow \alpha)}{\sum_{\alpha} Count(A \rightarrow \alpha)}$$

If our training corpus consists of two parse trees as shown below:

CHETAN. R
Asst. Professor





We will get the estimates as shown below:

Rule	Count($A \rightarrow \alpha$)	Count A	MLE estimates
S → VP	2	2	1
NP → Det Noun PP	1	4	0.25
NP → Det Noun	3	4	0.75
VP → Verb NP	2	3	0.66
VP → VP PP	1	3	0.33
Det → the	2	2	1
Noun → hole	2	4	0.5
Noun → door	2	4	0.5
Prep → with	1	1	1
Verb → Paint	1	1	1

We assign a probability to each parse tree φ of a sentence s. The probability of a complete parse is calculated by multiplying the probabilities for each of the rules used in generating the parse tree:

$$P(\varphi, s) = \prod_{n \in \varphi} p\{r(n)\}$$

CHETAN. R
Asst. Professor

where n is a node in the parse tree φ
and r is rule used used to expand n.

For a sentence, the parse trees generated by a PCFG are the same as those generated by a corresponding CFG. However, the PCFG assigns a probability to each parse. The probability of the two parse trees of the sentence Paint the door with the hole can be computed as follows:

$$P(t1) = 0.2 * 0.5 * 0.2 * 0.2 * 0.35 * 0.25 * 1.0 * 0.25 * 0.4 * 0.35 * 0.25$$
$$= 0.0000030625$$

$$P(t2) = 0.2 * 0.2 * 0.5 * 0.2 * 0.4 * 0.35 * 0.25 * 1 * 0.25 * 0.4 * 0.35 * 0.25$$
$$= 0.0000001225$$

In this example, the first tree has a higher probability leading to correct interpretation.

We can calculate probability to a sentence s by summing up probabilities of all possible parses associated with it.

$$P(s) = \sum_{t \in \tau(s)} P(t,s)$$

$$= \sum_{t \in \tau(s)} P(t)$$

Thus, the sentence will have the probability

$$P(t1) + P(t2) = 0.0000030625 + 0.0000001225$$
$$= 0.0000042875$$

Parsing PCFGs

Given a PCFG, a probabilistic parsing algorithm assigns the most likely parse φ' to a sentence s.

$$\varphi' = \text{argmax}_{T \in \tau(S)} P(T|S)$$

CHETAN. R
Asst. Professor,

where $\tau(S)$ is the set of all possible parse trees of s.

As in the basic CYK parsing algorithm, $w = w_1 w_2 w_3 \ldots w_i \ldots w_j \ldots w_n$ represents a sentence consisting of n words.

Let $\varphi[i,j,A]$ represent the maximum probability parse for a constituent with non-terminal A spanning words $i, i+1$, up to $i+j-1$.

This means it is a sub-tree rooted at A that derives sequence of j words beginning at position i and has a probability greater than all other possible sub-trees deriving the same word sequence.

An array named BP is used to store back pointers. These pointers allow us to recover the best parse. The output for a succesful parse is the maximum probability parse $\varphi[1,n,S]$, which corresponds to a parse tree rooted at S and spanning the sequence of words $w_1, w_2, w_3 \ldots w_n$, i.e. the whole sentence.

Initialization:

for $i := 1$ to n do

for all rules $A \longrightarrow w_i$ do

$\varphi[i, 1, A] = P(A \longrightarrow w_i)$

Recursive Step:

for $j = 2$ to n do

for $i = 1$ to $n - j + 1$ do

begin

$\varphi[i, 1, A] = \phi$

for $K = 1$ to $j - 1$ do

$\varphi'[i, j, A] = \max_K \varphi'[i, K, B] \times \varphi'[K, j, C] \times P(A \rightarrow BC)$,

Such that $A \rightarrow BC$ is a production rule in grammar

$BP[i, j, A] = \{K, A, B\}$

CHETAN. R
Asst. Professor

end

The first step is to generate items of length 1. However, in this case, we have to initialize the maximum probable parse trees deriving a string of length 1, with the probabilities of the terminal derivation rules used to derive them.

$$\varphi[i, 1, A] = P(A \longrightarrow w_i).$$

IIIly recursive step involves breaking a string into all possible ways and identifying the maximum probable parse.

$$\varphi'[i, j, A] = \max_K \varphi'[i, K, B] \times \varphi'[K, j, C] \times P(A \rightarrow BC)$$

Problems with PCFG

1. Problem lies in the independence assumption.
   We calculate the probability of a parse tree assuming that the rules are independent of each other. However this is not true. How a node expands depends on its location in the parse tree.

2. Lack of Sensitivity to lexical information.
   Lexical infomation plays a major role in determining correct parse in case of PP attachment ambiguities and coordination ambiguities.

Indian Languages :-

The majority of the Indian languages are free word order. By free word order language we mean that the order of words in a sentence can be changed without leading to a grammatically incorrect sentence.

For example:

सबा खाना खाती है । Saba Khana Khati hai.

खाना सबा खाती है । Khana Saba Khati hai.

Both are valid Hindi Sentences meaning Saba eats food. The same is true for Urdu and most other Indian languages. The CFG we used for parsing English is basically positional. It can be used to model language in which a position of the constituents carries useful information, but it fails to model free word order languages.

Extensive and productive use of complex predicates (CPs) is another property that most Indian languages have in common. A complex predicate combines a light verb with a verb, noun or adjective, to produce a new verb. For example,

(a) सबा आयी ।
(Saba Ayi)
Saba Came.

(b) सबा आ गयी ।
(Saba a gayi)
Saba Come went.
Saba arrived.

(c) सबा आ पड़ी ।
Saba a pari
Saba Come fell.
Saba came (Suddenly).

The use of post-position case markers (Vibhakti) and verb complexes consisting of sequences of verbs, e.g., खा रही है are other properties common to Indian languages. The auxiliary verbs in this sequence provide information about tense, aspect and modality.

CHETAN. R
Asst. Professor

Bharati and Sangal described an approach for parsing of Indian languages based on Paninian grammar formalism. It works in two stages.

1. It is responsible for identifying word groups.
2. Assigning a parse structure to the input sentence.

The input to the first stage comes from morphological analysis phases. The output of the first stage is word groups which are sequences of words that act as a unit.

Example:- A verb group consist of a main verb and a sequence of auxiliaries. For the sentence

लड़कियाँ मैदान में हाकी खेल रही हैं।

Ladkiyan maidaan mein hockey khel rahi hain.

The word ladkiyan forms one unit, the words maidaan and mein are grouped together to form a noun group, and the word sequence khel rahi hain forms a verb group.

In the second stage, the parser takes the word groups formed during first stage and identifies
(i) Karaka relations among them and
(ii) Senses of words.

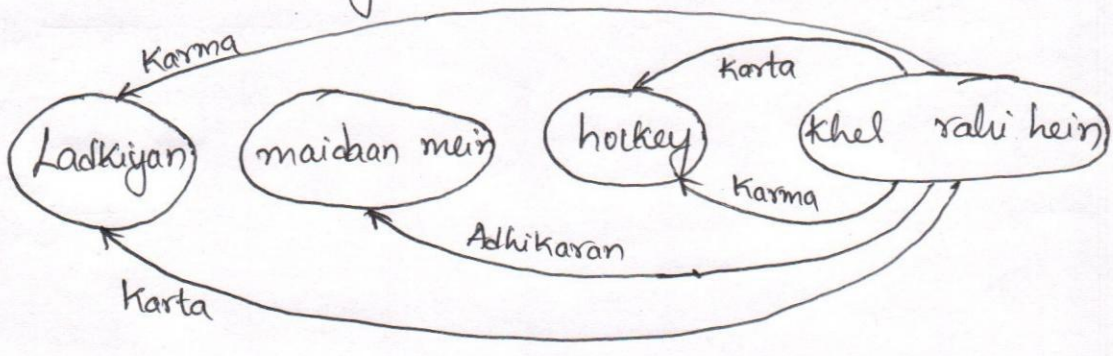
A data structure, called Karaka chart, stores additional information like Karaka-Vibhakti mapping, Karaka necessity and transformation rules for karaka relations, needed in this step.

CHETAN. R
Asst. Professor

Karaka chart is shown below:

Karaka (Case relations)	Vibhakti (case markers or post positions)	Necessity
Karta	ϕ	Mandatory
Karma	Ko or ϕ	Mandatory.
Adhikaran	Mein or par	Optional
Sampradan	Ko or Ke liye	Optional.

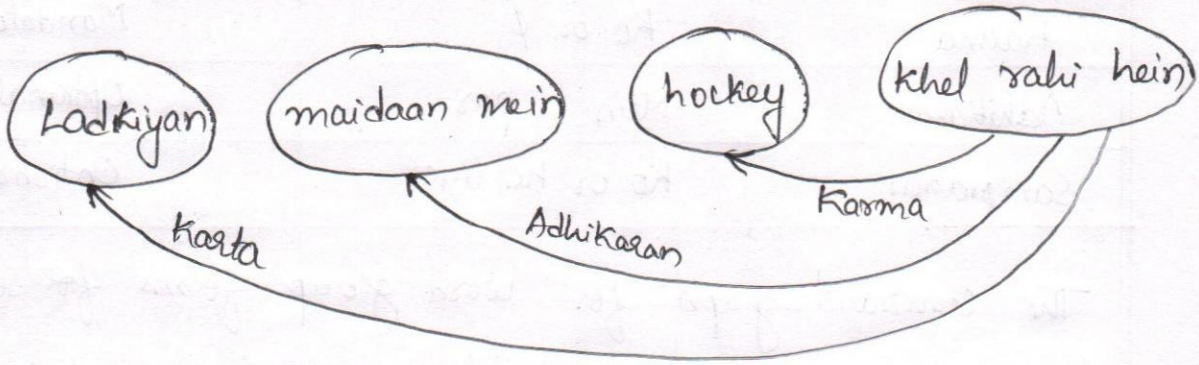
The constraint graph for word group forms for sentence Ladkiyan maidaan mein hockey khel rahi hain. is given below:



Each Sub-graph of the constraint graph that satisfies the following constraints yields a parse of the sentence.

1. It contains all the nodes of the graph.
2. It contains exactly one outgoing edge from a verb group for each of its mandatory Karakas. There edges are labelled by the corresponding Karaka.
3. For each of the optional Karaka in Karaka chart, the subgraph can have at most one outgoing edge labelled by the Karaka from the verb group.
4. For each noun group, the subgraph should have exactly one incoming edge.

CHETAN. R
Asst. Professor



3

Extracting Relations from Text: From Word Sequences to Dependency Paths

Razvan C. Bunescu and Raymond J. Mooney

3.1 Introduction

Extracting semantic relationships between entities mentioned in text documents is an important task in natural language processing. The various types of relationships that are discovered between mentions of entities can provide useful structured information to a text mining system [1]. Traditionally, the task specifies a predefined set of entity types and relation types that are deemed to be relevant to a potential user and that are likely to occur in a particular text collection. For example, information extraction from newspaper articles is usually concerned with identifying mentions of people, organizations, locations, and extracting useful relations between them. Relevant relation types range from social relationships, to roles that people hold inside an organization, to relations between organizations, to physical locations of people and organizations. Scientific publications in the biomedical domain offer a type of narrative that is very different from the newspaper discourse. A significant effort is currently spent on automatically extracting relevant pieces of information from Medline, an online collection of biomedical abstracts. Proteins, genes, and cells are examples of relevant entities in this task, whereas subcellular localizations and protein-protein interactions are two of the relation types that have received significant attention recently. The inherent difficulty of the relation extraction task is further compounded in the biomedical domain by the relative scarcity of tools able to analyze the corresponding type of narrative. Most existing natural language processing tools, such as tokenizers, sentence segmenters, part-of-speech (POS) taggers, shallow or full parsers are trained on newspaper corpora, and consequently they incur a loss in accuracy when applied to biomedical literature. Therefore, information extraction systems developed for biological corpora need to be robust to POS or parsing errors, or to give reasonable performance using shallower but more reliable information, such as chunking instead of full parsing.

In this chapter, we present two recent approaches to relation extraction that differ in terms of the kind of linguistic information they use:

1. In the first method (Section 3.2), each potential relation is represented implicitly as a vector of features, where each feature corresponds to a *word sequence* anchored at the two entities forming the relationship. A relation extraction system

is trained based on the subsequence kernel from [2]. This kernel is further generalized so that words can be replaced with word classes, thus enabling the use of information coming from POS tagging, named entity recognition, chunking, or Wordnet [3].
2. In the second approach (Section 3.3), the representation is centered on the shortest *dependency path* between the two entities in the dependency graph of the sentence. Because syntactic analysis is essential in this method, its applicability is limited to domains where syntactic parsing gives reasonable accuracy.

Entity recognition, a prerequisite for relation extraction, is usually cast as a sequence tagging problem, in which words are tagged as being either outside any entity, or inside a particular type of entity. Most approaches to entity tagging are therefore based on probabilistic models for labeling sequences, such as Hidden Markov Models [4], Maximum Entropy Markov Models [5], or Conditional Random Fields [6], and obtain a reasonably high accuracy. In the two information extraction methods presented in this chapter, we assume that the entity recognition task was done and focus only on the relation extraction part.

3.2 Subsequence Kernels for Relation Extraction

One of the first approaches to extracting interactions between proteins from biomedical abstracts is that of Blaschke *et al.*, described in [7, 8]. Their system is based on a set of manually developed rules, where each rule (or frame) is a sequence of words (or POS tags) and two protein-name tokens. Between every two adjacent words is a number indicating the maximum number of intervening words allowed when matching the rule to a sentence. An example rule is "*interaction of (3) <P> (3) with (3) <P>*", where '<P>' is used to denote a protein name. A sentence matches the rule if and only if it satisfies the word constraints in the given order and respects the respective word gaps.

In [9] the authors described a new method ELCS (Extraction using Longest Common Subsequences) that automatically learns such rules. ELCS' rule representation is similar to that in [7, 8], except that it currently does not use POS tags, but allows disjunctions of words. An example rule learned by this system is "- *(7) interaction (0) [between | of] (5) <P> (9) <P> (17)* ." Words in square brackets separated by '|' indicate disjunctive lexical constraints, i.e., one of the given words must match the sentence at that position. The numbers in parentheses between adjacent constraints indicate the maximum number of unconstrained words allowed between the two.

3.2.1 Capturing Relation Patterns with a String Kernel

Both Blaschke and ELCS do relation extraction based on a limited set of matching rules, where a rule is simply a sparse (gappy) subsequence of words or POS tags anchored on the two protein-name tokens. Therefore, the two methods share a common limitation: either through manual selection (Blaschke), or as a result of a greedy learning procedure (ELCS), they end up using only a subset of all possible anchored sparse subsequences. Ideally, all such anchored sparse subsequences would be used as features, with weights reflecting their relative accuracy. However,

explicitly creating for each sentence a vector with a position for each such feature is infeasible, due to the high dimensionality of the feature space. Here, we exploit dual learning algorithms that process examples only via computing their dot-products, such as in Support Vector Machines (SVMs) [10, 11]. An SVM learner tries to find a hyperplane that separates positive from negative examples and at the same time maximizes the separation (margin) between them. This type of max-margin separator has been shown both theoretically and empirically to resist overfitting and to provide good generalization performance on unseen examples.

Computing the dot-product (i.e., the kernel) between the features vectors associated with two relation examples amounts to calculating the number of common anchored subsequences between the two sentences. This is done efficiently by modifying the dynamic programming algorithm used in the string kernel from [2] to account only for common sparse subsequences constrained to contain the two protein-name tokens. The feature space is further prunned down by utilizing the following property of natural language statements: when a sentence asserts a relationship between two entity mentions, it generally does this using one of the following four patterns:

• **[FB] F**ore–**B**etween: words before and between the two entity mentions are simultaneously used to express the relationship. Examples: 'interaction of $\langle P_1 \rangle$ with $\langle P_2 \rangle$,' 'activation of $\langle P_1 \rangle$ by $\langle P_2 \rangle$.'

• **[B] B**etween: only words between the two entities are essential for asserting the relationship. Examples: '$\langle P_1 \rangle$ interacts with $\langle P_2 \rangle$,' '$\langle P_1 \rangle$ is activated by $\langle P_2 \rangle$.'

• **[BA] B**etween–**A**fter: words between and after the two entity mentions are simultaneously used to express the relationship. Examples: '$\langle P_1 \rangle$ – $\langle P_2 \rangle$ complex,' '$\langle P_1 \rangle$ and $\langle P_2 \rangle$ interact.'

• **[M] M**odifier: the two entity mentions have no words between them. Examples: *U.S. troops* (a ROLE:STAFF relation), *Serbian general* (ROLE:CITIZEN).

While the first three patterns are sufficient to capture most cases of interactions between proteins, the last pattern is needed to account for various relationships expressed through noun-noun or adjective-noun compounds in the newspaper corpora.

Another observation is that all these patterns use at most four words to express the relationship (not counting the two entity names). Consequently, when computing the relation kernel, we restrict the counting of common anchored subsequences only to those having one of the four types described above, with a maximum word-length of four. This type of feature selection leads not only to a faster kernel computation, but also to less overfitting, which results in increased accuracy.

The patterns enumerated above are completely lexicalized and consequently their performance is limited by data sparsity. This can be alleviated by categorizing words into classes with varying degrees of generality, and then allowing patterns to use both words and their classes. Examples of word classes are POS tags and generalizations over POS tags such as Noun, Active Verb, or Passive Verb. The entity type can also be used if the word is part of a known named entity. Also, if the sentence is segmented into syntactic chunks such as noun phrases (NP) or verb phrases (VP), the system may choose to consider only the head word from each chunk, together with the type of the chunk as another word class. Content words such as nouns and verbs can also be related to their synsets via WordNet. Patterns then will consist of sparse subsequences of words, POS tags, generalized POS tags, entity and chunk types, or WordNet synsets. For example, 'Noun of $\langle P_1 \rangle$ by $\langle P_2 \rangle$' is an FB pattern based on words and general POS tags.

3.2.2 A Generalized Subsequence Kernel

Let $\Sigma_1, \Sigma_2, ..., \Sigma_k$ be some disjoint feature spaces. Following the example in Section 3.2.1, Σ_1 could be the set of words, Σ_2 the set of POS tags, etc. Let $\Sigma_\times = \Sigma_1 \times \Sigma_2 \times ... \times \Sigma_k$ be the set of all possible feature vectors, where a feature vector would be associated with each position in a sentence. Given two feature vectors $x, y \in \Sigma_\times$, let $c(x, y)$ denote the number of common features between x and y. The next notation follows that introduced in [2]. Thus, let s, t be two sequences over the finite set $\Sigma_\times$, and let $|s|$ denote the length of $s = s_1...s_{|s|}$. The sequence $s[i{:}j]$ is the contiguous subsequence $s_i...s_j$ of s. Let $\mathbf{i} = (i_1, ..., i_{|\mathbf{i}|})$ be a sequence of $|\mathbf{i}|$ indices in s, in ascending order. We define the length $l(\mathbf{i})$ *of the index sequence* $\mathbf{i}$ *in* s as $i_{|\mathbf{i}|} - i_1 + 1$. Similarly, $\mathbf{j}$ is a sequence of $|\mathbf{j}|$ indices in t.

Let $\Sigma_\cup = \Sigma_1 \cup \Sigma_2 \cup ... \cup \Sigma_k$ be the set of all possible features. We say that the sequence $u \in \Sigma_\cup^*$ is a (sparse) subsequence of s if there is a sequence of $|u|$ indices $\mathbf{i}$ such that $u_k \in s_{i_k}$, for all $k = 1, ..., |u|$. Equivalently, we write $u \prec s[\mathbf{i}]$ as a shorthand for the component-wise '$\in$' relationship between u and $s[\mathbf{i}]$.

Finally, let $K_n(s, t, \lambda)$ (Equation 3.1) be the number of weighted sparse subsequences u of length n common to s and t (i.e., $u \prec s[\mathbf{i}]$, $u \prec t[\mathbf{j}]$), where the weight of u is $\lambda^{l(\mathbf{i})+l(\mathbf{j})}$, for some $\lambda \leq 1$.

$$K_n(s, t, \lambda) = \sum_{u \in \Sigma_\cup^n} \sum_{\mathbf{i}: u \prec s[\mathbf{i}]} \sum_{\mathbf{j}: u \prec t[\mathbf{j}]} \lambda^{l(\mathbf{i})+l(\mathbf{j})} \tag{3.1}$$

Let $\mathbf{i}$ and $\mathbf{j}$ be two index sequences of length n. By definition, for every k between 1 and n, $c(s_{i_k}, t_{j_k})$ returns the number of common features between s and t at positions i_k and j_k. If $c(s_{i_k}, t_{j_k}) = 0$ for some k, there are no common feature sequences of length n between $s[\mathbf{i}]$ and $t[\mathbf{j}]$. On the other hand, if $c(s_{i_k}, t_{j_k})$ is greater than 1, this means that there is more than one common feature that can be used at position k to obtain a common feature sequence of length n. Consequently, the number of common feature sequences of length n between $s[\mathbf{i}]$ and $t[\mathbf{j}]$, i.e., the size of the set $\{u \in \Sigma_\cup^n | u \prec s[\mathbf{i}], u \prec t[\mathbf{j}]\}$, is given by $\prod_{k=1}^{n} c(s_{i_k}, t_{j_k})$. Therefore, $K_n(s, t, \lambda)$ can be rewritten as in Equation 3.2:

$$K_n(s, t, \lambda) = \sum_{\mathbf{i}: |\mathbf{i}|=n} \sum_{\mathbf{j}: |\mathbf{j}|=n} \prod_{k=1}^{n} c(s_{i_k}, t_{j_k}) \lambda^{l(\mathbf{i})+l(\mathbf{j})} \tag{3.2}$$

We use λ as a decaying factor that penalizes longer subsequences. For sparse subsequences, this means that wider gaps will be penalized more, which is exactly the desired behavior for our patterns. Through them, we try to capture head-modifier dependencies that are important for relation extraction; for lack of reliable dependency information, the larger the word gap is between two words, the less confident we are in the existence of a head-modifier relationship between them.

To enable an efficient computation of K_n, we use the auxiliary function $K_n^{'}$ with a definition similar to K_n, the only difference being that it counts the length from the beginning of the particular subsequence u to the end of the strings s and t, as illustrated in Equation 3.3:

$$K_n^{'}(s, t, \lambda) = \sum_{u \in \Sigma_\cup^n} \sum_{\mathbf{i}: u \prec s[\mathbf{i}]} \sum_{\mathbf{j}: u \prec t[\mathbf{j}]} \lambda^{|s|+|t|-i_1-j_1+2} \tag{3.3}$$

An equivalent formula for $K_n^{'}(s,t,\lambda)$ is obtained by changing the exponent of λ from Equation 3.2 to $|s|+|t|-i_1-j_1+2$.

Based on all definitions above, K_n is computed in $O(kn|s||t|)$ time, by modifying the recursive computation from [2] with the new factor $c(x,y)$, as shown in Figure 3.1. As in [2], the complexity of computing $K_i^{'}(s,t)$ is reduced to $O(|s||t|)$ by first evaluating another auxiliary factor $K_i^{''}(s,t)$. In Figure 3.1, the sequence sx is the result of appending x to s (with ty defined in a similar way). To avoid clutter, the parameter λ is not shown in the argument list of K and K', unless it is instantiated to a specific constant.

$$
\begin{aligned}
K_0^{'}(s,t) &= 1,\ for\ all\ s,t \\
K_i^{'}(s,t) &= 0,\ if\ \min(|s|,|t|) < i \\
K_i^{''}(s,\emptyset) &= 0,\ for\ all\ i,s \\
K_i^{''}(sx,ty) &= \lambda K_i^{''}(sx,t) + \lambda^2 K_{i-1}^{'}(s,t)\cdot c(x,y) \\
K_i^{'}(sx,t) &= \lambda K_i^{'}(s,t) + K_i^{''}(sx,t) \\
K_n(s,t) &= 0,\ if\ \min(|s|,|t|) < n \\
K_n(sx,t) &= K_n(s,t) + \sum_j \lambda^2 K_{n-1}^{'}(s,t[1:j-1])\cdot c(x,t[j])
\end{aligned}
$$

Fig. 3.1. Computation of subsequence kernel.

3.2.3 Computing the Relation Kernel

As described at the beginning of Section 3.2, the input consists of a set of sentences, where each sentence contains exactly two entities (protein names in the case of interaction extraction). In Figure 3.2 we show the segments that will be used for computing the relation kernel between two example sentences s and t. In sentence s, for instance, x_1 and x_2 are the two entities, s_f is the sentence segment before x_1, s_b is the segment between x_1 and x_2, and s_a is the sentence segment after x_2. For convenience, we also include the auxiliary segment $s_b^{'} = x_1 s_b x_2$, whose span is computed as $l(s_b^{'}) = l(s_b) + 2$ (in all length computations, we consider x_1 and x_2 as contributing one unit only).

The relation kernel computes the number of common patterns between two sentences s and t, where the set of patterns is restricted to the four types introduced in Section 3.2.1. Therefore, the kernel $rK(s,t)$ is expressed as the sum of four subkernels: $fbK(s,t)$ counting the number of common fore–between patterns, $bK(s,t)$ for between patterns, $baK(s,t)$ for between–after patterns, and $mK(s,t)$ for modifier patterns, as in Figure 3.3. The symbol $\mathbb{1}$ is used there as a shorthand for the indicator function, which is 1 if the argument is true, and 0 otherwise.

The first three sub-kernels include in their computation the counting of common subsequences between $s_b^{'}$ and $t_b^{'}$. In order to speed up the computation, all these

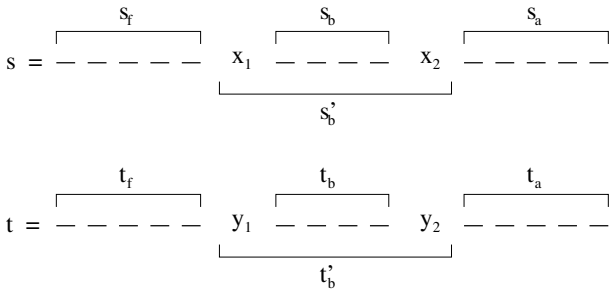


Fig. 3.2. Sentence segments.

$$
\begin{aligned}
rK(s,t) &= fbK(s,t) + bK(s,t) + baK(s,t) + mK(s,t) \\
bK_i(s,t) &= K_i(s_b, t_b, 1) \cdot c(x_1, y_1) \cdot c(x_2, y_2) \cdot \lambda^{l(s_b^{'})+l(t_b^{'})} \\
fbK(s,t) &= \sum_{i,j} bK_i(s,t) \cdot K_j^{'}(s_f, t_f), \quad 1 \le \mathrm{i},\ 1 \le \mathrm{j},\ \mathrm{i}+\mathrm{j} < \mathrm{fb}_{\max} \\
bK(s,t) &= \sum_{i} bK_i(s,t), \quad 1 \le \mathrm{i} \le \mathrm{b}_{\max} \\
baK(s,t) &= \sum_{i,j} bK_i(s,t) \cdot K_j^{'}(s_a^{-}, t_a^{-}), \quad 1 \le \mathrm{i},\ 1 \le \mathrm{j},\ \mathrm{i}+\mathrm{j} < \mathrm{ba}_{\max} \\
mK(s,t) &= \mathbb{1}(s_b = \emptyset) \cdot \mathbb{1}(t_b = \emptyset) \cdot c(x_1, y_1) \cdot c(x_2, y_2) \cdot \lambda^{2+2},
\end{aligned}
$$

Fig. 3.3. Computation of relation kernel.

common counts are calculated separately in bK_i, which is defined as the number of common subsequences of length i between $s_b^{'}$ and $t_b^{'}$, anchored at x_1/x_2 and y_1/y_2 respectively (i.e., constrained to start at x_1 in $s_b^{'}$ and y_1 in $t_b^{'}$, and to end at x_2 in $s_b^{'}$ and y_2 in $t_b^{'}$). Then fbK simply counts the number of subsequences that match j positions before the first entity and i positions between the entities, constrained to have length less than a constant fb_{max}. To obtain a similar formula for baK we simply use the reversed (mirror) version of segments s_a and t_a (e.g., s_a^- and t_a^-). In Section 3.2.1 we observed that all three subsequence patterns use at most 4 words to express a relation, therefore the constants fb_{max}, b_{max} and ba_{max} are set to 4. Kernels K and $K^{'}$ are computed using the procedure described in Section 3.2.2.

3.3 A Dependency-Path Kernel for Relation Extraction

The pattern examples from Section 3.2.1 show the two entity mentions, together with the set of words that are relevant for their relationship. A closer analysis of

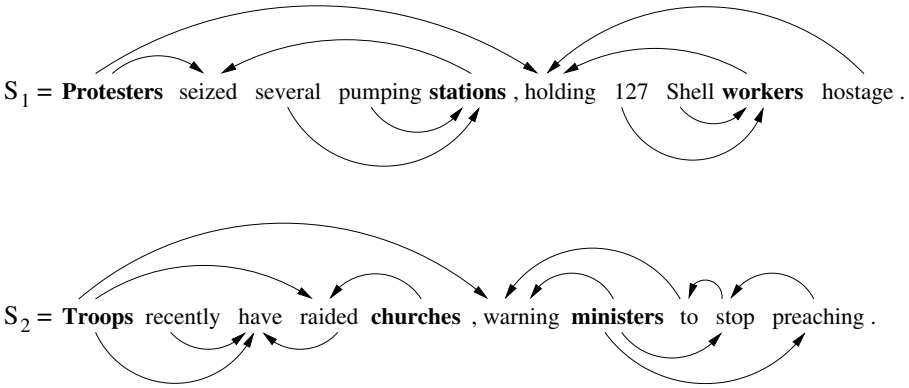


Fig. 3.4. Sentences as dependency graphs.

these examples reveals that all relevant words form a shortest path between the two entities in a graph structure where edges correspond to relations between a word (head) and its dependents. For example, Figure 3.4 shows the full dependency graphs for two sentences from the ACE (Automated Content Extraction) newspaper corpus [12], in which words are represented as nodes and word-word dependencies are represented as directed edges. A subset of these word-word dependencies capture the predicate-argument relations present in the sentence. Arguments are connected to their target predicates either directly through an arc pointing to the predicate ('troops → raided'), or indirectly through a preposition or infinitive particle ('warning ← to ← stop'). Other types of word-word dependencies account for modifier-head relationships present in adjective-noun compounds ('several → stations'), noun-noun compounds ('pumping → stations'), or adverb-verb constructions ('recently → raided').

Word-word dependencies are typically categorized in two classes as follows:

- [**Local Dependencies**] These correspond to local predicate-argument (or head-modifier) constructions such as 'troops → raided', or 'pumping → stations' in Figure 3.4.
- [**Non-local Dependencies**] Long-distance dependencies arise due to various linguistic constructions such as coordination, extraction, raising and control. In Figure 3.4, among non-local dependencies are 'troops → warning', or 'ministers → preaching'.

A Context Free Grammar (CFG) parser can be used to extract local dependencies, which for each sentence form a dependency tree. Mildly context sensitive formalisms such as Combinatory Categorial Grammar (CCG) [13] model word-word dependencies more directly and can be used to extract both local and long-distance dependencies, giving rise to a directed acyclic graph, as illustrated in Figure 3.4.

3.3.1 The Shortest Path Hypothesis

If e_1 and e_2 are two entities mentioned in the same sentence such that they are observed to be in a relationship R, then the contribution of the sentence dependency

Table 3.1. Shortest Path representation of relations.

Relation Instance	Shortest Path in Undirected Dependency Graph
S_1:protesters AT stations	**protesters** → seized ← **stations**
S_1:workers AT stations	**workers** → holding ← protesters → seized ← **stations**
S_2:troops AT churches	**troops** → raided ← **churches**
S_2:ministers AT churches	**ministers** → warning ← troops → raided ← **churches**

graph to establishing the relationship $R(e_1, e_2)$ is almost exclusively concentrated in the shortest path between e_1 and e_2 in the undirected version of the dependency graph.

If entities e_1 and e_2 are arguments of the same predicate, then the shortest path between them will pass through the predicate, which may be connected directly to the two entities, or indirectly through prepositions. If e_1 and e_2 belong to different predicate-argument structures that share a common argument, then the shortest path will pass through this argument. This is the case with the shortest path between 'stations' and 'workers' in Figure 3.4, passing through 'protesters,' which is an argument common to both predicates 'holding' and 'seized'. In Table 3.1, we show the paths corresponding to the four relation instances encoded in the ACE corpus for the two sentences from Figure 3.4. All these paths support the LOCATED relationship. For the first path, it is reasonable to infer that if a PERSON entity (e.g., 'protesters') is doing some action (e.g., 'seized') to a FACILITY entity (e.g., 'station'), then the PERSON entity is LOCATED at that FACILITY entity. The second path captures the fact that the same PERSON entity (e.g., 'protesters') is doing two actions (e.g., 'holding' and 'seized') , one action to a PERSON entity (e.g., 'workers'), and the other action to a FACILITY entity (e.g., 'station'). A reasonable inference in this case is that the 'workers' are LOCATED at the 'station'.

In Figure 3.5, we show three more examples of the LOCATED (AT) relationship as dependency paths created from one or two predicate-argument structures. The second example is an interesting case, as it illustrates how annotation decisions are accommodated in our approach. Using a reasoning similar with that from the previous paragraph, it is reasonable to infer that 'troops' are LOCATED in 'vans,' and that 'vans' are LOCATED in 'city'. However, because 'vans' is not an ACE markable, it cannot participate in an annotated relationship. Therefore, 'troops' is annotated as being LOCATED in 'city,' which makes sense due to the transitivity of the relation LOCATED. In our approach, this leads to shortest paths that pass through two or more predicate-argument structures.

The last relation example is a case where there exist multiple shortest paths in the dependency graph between the same two entities – there are actually two different paths, with each path replicated into three similar paths due to coordination. Our current approach considers only one of the shortest paths, nevertheless it seems reasonable to investigate using all of them as multiple sources of evidence for relation extraction.

There may be cases where e_1 and e_2 belong to predicate-argument structures that have no argument in common. However, because the dependency graph is always connected, we are guaranteed to find a shortest path between the two entities. In general, we shall find a shortest sequence of predicate-argument structures with

target predicates $P_1, P_2, ..., P_n$ such that e_1 is an argument of P_1, e_2 is an argument of P_n, and any two consecutive predicates P_i and P_{i+1} share a common argument (where by "argument" we mean both arguments and complements).

(1) He had no regrets for **his** actions in **Brcko**.

his → actions ← in ← **Brcko**

(2) U.S. **troops** today acted for the first time to capture an alleged Bosnian war criminal, rushing from unmarked vans parked in the northern Serb-dominated **city** of Bijeljina.

troops → rushing ← from ← vans → parked ← in ← **city**

(3) Jelisic created an atmosphere of terror at the **camp** by killing, abusing and threatening the **detainees**.

detainees → killing ← Jelisic → created ← at ← **camp**
detainees → abusing ← Jelisic → created ← at ← **camp**
detainees → threatning ← Jelisic → created ← at ← **camp**
detainees → killing → by → created ← at ← **camp**
detainees → abusing → by → created ← at ← **camp**
detainees → threatening → by → created ← at ← **camp**

Fig. 3.5. Relation examples.

3.3.2 Learning with Dependency Paths

The shortest path between two entities in a dependency graph offers a very condensed representation of the information needed to assess their relationship. A dependency path is represented as a sequence of words interspersed with arrows that indicate the orientation of each dependency, as illustrated in Table 3.1. These paths, however, are completely lexicalized and consequently their performance will be limited by data sparsity. The solution is to allow paths to use both words and their word classes, similar with the approach taken for the subsequence patterns in Section 3.2.1.

The set of features can then be defined as a Cartesian product over words and word classes, as illustrated in Figure 3.6 for the dependency path between 'protesters' and 'station' in sentence S_1. In this representation, sparse or contiguous subsequences of nodes along the lexicalized dependency path (i.e., path fragments) are included as features simply by replacing the rest of the nodes with their corresponding generalizations.

Examples of features generated by Figure 3.6 are "protesters → seized ← stations," "Noun → Verb ← Noun," "PERSON → seized ← FACILITY," or "PERSON → Verb ← FACILITY." The total number of features generated by this dependency path is $4 \times 1 \times 3 \times 1 \times 4$.

$$\begin{bmatrix} \text{protesters} \\ \text{NNS} \\ \text{Noun} \\ \text{PERSON} \end{bmatrix} \times [\rightarrow] \times \begin{bmatrix} \text{seized} \\ \text{VBD} \\ \text{Verb} \end{bmatrix} \times [\leftarrow] \times \begin{bmatrix} \text{stations} \\ \text{NNS} \\ \text{Noun} \\ \text{FACILITY} \end{bmatrix}$$

Fig. 3.6. Feature generation from dependency path.

For verbs and nouns (and their respective word classes) occurring along a dependency path we also use an additional suffix '(-)' to indicate a negative polarity item. In the case of verbs, this suffix is used when the verb (or an attached auxiliary) is modified by a negative polarity adverb such as 'not' or 'never.' Nouns get the negative suffix whenever they are modified by negative determiners such as 'no,' 'neither' or 'nor.' For example, the phrase "He never went to Paris" is associated with the dependency path "He $\rightarrow$ went(-) $\leftarrow$ to $\leftarrow$ Paris."

As in Section 3.2, we use kernel SVMs in order to avoid working explicitly with high-dimensional dependency path feature vectors. Computing the dot-product (i.e., kernel) between two relation examples amounts to calculating the number of common features (i.e., paths) between the two examples. If $\boldsymbol{x} = x_1 x_2 ... x_m$ and $\boldsymbol{y} = y_1 y_2 ... y_n$ are two relation examples, where x_i denotes the set of word classes corresponding to position i (as in Figure 3.6), then the number of common features between $\boldsymbol{x}$ and $\boldsymbol{y}$ is computed as in Equation 3.4.

$$K(\mathbf{x}, \mathbf{y}) = \mathbb{1}(m = n) \cdot \prod_{i=1}^{n} c(x_i, y_i) \tag{3.4}$$

where $c(x_i, y_i) = |x_i \cap y_i|$ is the number of common word classes between x_i and y_i.

This is a simple kernel, whose computation takes $O(n)$ time. If the two paths have different lengths, they correspond to different ways of expressing a relationship – for instance, they may pass through a different number of predicate argument structures. Consequently, the kernel is defined to be 0 in this case. Otherwise, it is the product of the number of common word classes at each position in the two paths. As an example, let us consider two instances of the LOCATED relationship, and their corresponding dependency paths:

1. '**his** actions in **Brcko**' (**his** $\rightarrow$ actions $\leftarrow$ in $\leftarrow$ **Brcko**).
2. '**his** arrival in **Beijing**' (**his** $\rightarrow$ arrival $\leftarrow$ in $\leftarrow$ **Beijing**).

Their representation as a sequence of sets of word classes is given by:

1. $\boldsymbol{x} = [x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7]$, where $x_1 = \{$his, PRP, PERSON$\}$, $x_2 = \{\rightarrow\}$, $x_3 = \{$actions, NNS, Noun$\}$, $x_4 = \{\leftarrow\}$, $x_5 = \{$in, IN$\}$, $x_6 = \{\leftarrow\}$, $x_7 = \{$Brcko, NNP, Noun, LOCATION$\}$
2. $\boldsymbol{y} = [y_1\ y_2\ y_3\ y_4\ y_5\ y_6\ y_7]$, where $y_1 = \{$his, PRP, PERSON$\}$, $y_2 = \{\rightarrow\}$, $y_3 = \{$arrival, NN, Noun$\}$, $y_4 = \{\leftarrow\}$, $y_5 = \{$in, IN$\}$, $y_6 = \{\leftarrow\}$, $y_7 = \{$Beijing, NNP, Noun, LOCATION$\}$

Based on the formula from Equation 3.4, the kernel is computed as $K(\boldsymbol{x}, \boldsymbol{y}) = 3 \times 1 \times 1 \times 1 \times 2 \times 1 \times 3 = 18$.

3.4 Experimental Evaluation

The two relation kernels described above are evaluated on the task of extracting relations from two corpora with different types of narrative, which are described in more detail in the following sections. In both cases, we assume that the entities and their labels are known. All preprocessing steps – sentence segmentation, tokenization, POS tagging, and chunking – were performed using the OpenNLP[1] package. If a sentence contains n entities ($n \geq 2$), it is replicated into $\binom{n}{2}$ sentences, each containing only two entities. If the two entities are known to be in a relationship, then the replicated sentence is added to the set of corresponding positive sentences, otherwise it is added to the set of negative sentences. During testing, a sentence having n entities ($n \geq 2$) is again replicated into $\binom{n}{2}$ sentences in a similar way.

The dependency graph that is input to the shortest path dependecy kernel is obtained from two different parsers:

- The CCG parser introduced in [14][2] outputs a list of functor-argument dependencies, from which head-modifier dependencies are obtained using a straightforward procedure (for more details, see [15]).
- Head-modifier dependencies can be easily extracted from the full parse output of Collins' CFG parser [16], in which every non-terminal node is annotated with head information.

The relation kernels are used in conjunction with SVM learning in order to find a decision hyperplane that best separates the positive examples from negative examples. We modified the LibSVM[3] package by plugging in the kernels described above. The factor λ in the subsequence kernel is set to 0.75. The performance is measured using *precision* (percentage of correctly extracted relations out of the total number of relations extracted), *recall* (percentage of correctly extracted relations out of the total number of relations annotated in the corpus), and *F-measure* (the harmonic mean of *precision* and *recall*).

3.4.1 Interaction Extraction from AIMed

We did comparative experiments on the AIMed corpus, which has been previously used for training the protein interaction extraction systems in [9]. It consists of 225 Medline abstracts, of which 200 are known to describe interactions between human proteins, while the other 25 do not refer to any interaction. There are 4084 protein references and around 1000 tagged interactions in this dataset.

The following systems are evaluated on the task of retrieving protein interactions from AIMed (assuming gold standard proteins):

• **[Manual]**: We report the performance of the rule-based system of [7, 8].

• **[ELCS]**: We report the 10-fold cross-validated results from [9] as a Precision-Recall (PR) graph.

• **[SSK]**: The subseqeuence kernel is trained and tested on the same splits as ELCS. In order to have a fair comparison with the other two systems, which use only lexical information, we do not use any word classes here.

[1] URL: http://opennlp.sourceforge.net

[2] URL:http://www.ircs.upenn.edu/~juliahr/Parser/

[3] URL:http://www.csie.ntu.edu.tw/~cjlin/libsvm/

• **[SPK]**: This is the shortest path dependency kernel, using the head-modifier dependencies extracted by Collins' syntactic parser. The kernel is trained and tested on the same 10 splits as ELCS and SSK.

The Precision-Recall curves that show the trade-off between these metrics are obtained by varying a threshold on the minimum acceptable extraction confidence, based on the probability estimates from LibSVM. The results, summarized in Figure 3.7, show that the subsequence kernel outperforms the other three systems, with a substantial gain. The syntactic parser, which is originally trained on a newspaper corpus, builds less accurate dependency structures for the biomedical text. This is reflected in a significantly reduced accuracy for the dependency kernel.

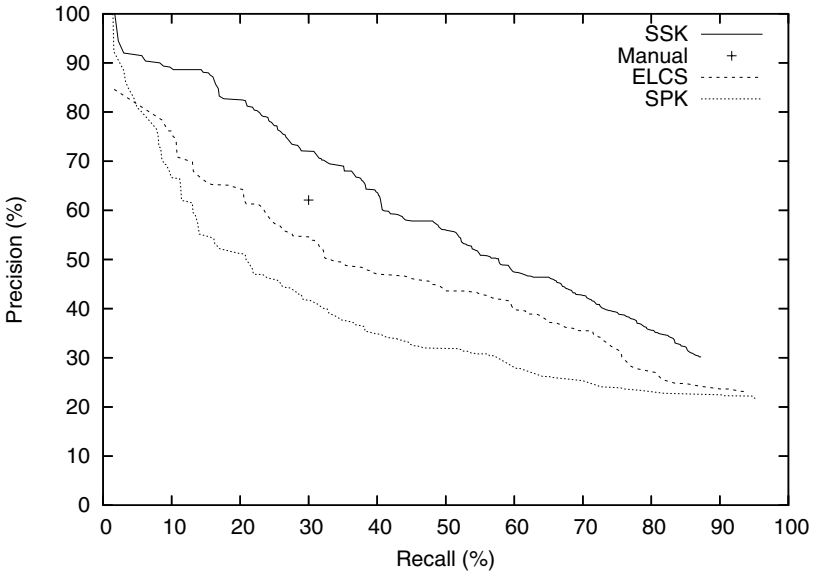


Fig. 3.7. Precision-Recall curves for protein interaction extractors.

3.4.2 Relation Extraction from ACE

The two kernels are also evaluated on the task of extracting top-level relations from the ACE corpus [12], the version used for the September 2002 evaluation. The training part of this dataset consists of 422 documents, with a separate set of 97 documents reserved for testing. This version of the ACE corpus contains three types of annotations: coreference, named entities and relations. There are five types of entities – Person, Organization, Facility, Location, and Geo-Political Entity – which can participate in five general, top-level relations: Role, Part, Located, Near, and Social. In total, there are 7,646 intra-sentential relations, of which 6,156 are in the training data and 1,490 in the test data.

A recent approach to extracting relations is described in [17]. The authors use a generalized version of the tree kernel from [18] to compute a kernel over relation examples, where a relation example consists of the smallest dependency tree containing the two entities of the relation. Precision and recall values are reported for the task of extracting the five top-level relations in the ACE corpus under two different scenarios:

– **[S1]** This is the classic setting: one multi-class SVM is learned to discriminate among the five top-level classes, plus one more class for the no-relation cases.

– **[S2]** One binary SVM is trained for *relation detection*, meaning that all positive relation instances are combined into one class. The thresholded output of this binary classifier is used as training data for a second multi-class SVM, trained for *relation classification*.

The subsequence kernel (SSK) is trained under the first scenario, to recognize the same five top-level relation types. While for protein interaction extraction only the lexicalized version of the kernel was used, here we utilize more features, corresponding to the following feature spaces: Σ_1 is the word vocabulary, Σ_2 is the set of POS tags, Σ_3 is the set of generic POS tags, and Σ_4 contains the five entity types. Chunking information is used as follows: all (sparse) subsequences are created exclusively from the chunk heads, where a head is defined as the last word in a chunk. The same criterion is used for computing the length of a subsequence – all words other than head words are ignored. This is based on the observation that in general words other than the chunk head do not contribute to establishing a relationship between two entities outside of that chunk. One exception is when both entities in the example sentence are contained in the same chunk. This happens very often due to noun-noun ('U.S. troops') or adjective-noun ('Serbian general') compounds. In these cases, the chunk is allowed to contribute both entity heads.

The shortest-path dependency kernel (SPK) is trained under both scenarios. The dependencies are extracted using either Hockenmaier's CCG parser (SPK-CCG) [14], or Collins' CFG parser (SPK-CFG) [16].

Table 3.2 summarizes the performance of the two relation kernels on the ACE corpus. For comparison, we also show the results presented in [17] for their best performing kernel K4 (a sum between a bag-of-words kernel and a tree dependency kernel) under both scenarios.

Table 3.2. Extraction Performance on ACE.

(Scenario) Method	Precision	Recall	F-measure
(S1) K4	70.3	26.3	38.0
(S1) SSK	73.9	35.2	47.7
(S1) SPK-CCG	67.5	37.2	48.0
(S1) SPK-CFG	**71.1**	**39.2**	**50.5**
(S2) K4	67.1	35.0	45.8
(S2) SPK-CCG	63.7	41.4	50.2
(S2) SPK-CFG	**65.5**	**43.8**	**52.5**

The shortest-path dependency kernels outperform the dependency kernel from [17] in both scenarios, with a more substantial gain for SP-CFG. An error analysis revealed that Collins' parser was better at capturing local dependencies, hence the increased accuracy of SP-CFG. Another advantage of shortest-path dependency kernels is that their training and testing are very fast – this is due to representing the sentence as a chain of dependencies on which a fast kernel can be computed. All of the four SP kernels from Table 3.2 take between 2 and 3 hours to train and test on a 2.6GHz Pentium IV machine.

As expected, the newspaper articles from ACE are less prone to parsing errors than the biomedical articles from AIMed. Consequently, the extracted dependency structures are more accurate, leading to an improved accuracy for the dependency kernel.

To avoid numerical problems, the dependency paths are constrained to pass through at most 10 words (as observed in the training data) by setting the kernel to 0 for longer paths. The alternative solution of normalizing the kernel leads to a slight decrease in accuracy. The fact that longer paths have larger kernel scores in the unnormalized version does not pose a problem because, by definition, paths of different lengths correspond to disjoint sets of features. Consequently, the SVM algorithm will induce lower weights for features occurring in longer paths, resulting in a linear separator that works irrespective of the size of the dependency paths.

3.5 Future Work

There are cases when words that do not belong to the shortest dependency path do influence the extraction decision. In Section 3.3.2, we showed how negative polarity items are integrated in the model through annotations of words along the dependency paths. Modality is another phenomenon that is influencing relation extraction, and we plan to incorporate it using the same annotation approach.

The two relation extraction methods are very similar: the subsequence patterns in one kernel correspond to dependency paths in the second kernel. More exactly, pairs of words from a subsequence pattern correspond to pairs of consecutive words (i.e., edges) on the dependency path. The lack of dependency information in the subsequence kernel leads to allowing gaps between words, with the corresponding exponential penalty factor λ. Given the observed similarity between the two methods, it seems reasonable to use them both in an integrated model. This model would use high-confidence head-modifier dependencies, falling back on pairs of words with gaps, when the dependency information is unreliable.

3.6 Conclusion

Mining knowledge from text documents can benefit from using the structured information that comes from entity recognition and relation extraction. However, accurately extracting relationships between relevant entities is dependent on the granularity and reliability of the required linguistic analysis. In this chapter, we presented two relation extraction kernels that differ in terms of the amount of linguistic information they use. Experimental evaluations on two corpora with different types of discourse show that they compare favorably to previous extraction approaches.

3.7 Acknowledgment

This work was supported by grants IIS-0117308 and IIS-0325116 from the NSF. We would like to thank Arun Ramani and Edward Marcotte for their help in preparing the AIMed corpus.

References

1. R. J. Mooney, R. C. Bunescu, Mining knowledge from text using information extraction, SIGKDD Explorations (special issue on Text Mining and Natural Language Processing) 7 (1) (2005) 3–10.
2. H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, C. Watkins, Text classification using string kernels, Journal of Machine Learning Research 2 (2002) 419–444.
3. C. D. Fellbaum, WordNet: An Electronic Lexical Database, MIT Press, Cambridge, MA, 1998.
4. L. R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proceedings of the IEEE 77 (2) (1989) 257–286.
5. A. McCallum, D. Freitag, F. Pereira, Maximum entropy Markov models for information extraction and segmentation, in: Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000), Stanford, CA, 2000.
6. J. Lafferty, A. McCallum, F. Pereira, Conditional random fields: Probabilistic models for segmenting and labeling sequence data, in: Proceedings of 18th International Conference on Machine Learning (ICML-2001), Williamstown, MA, 2001, pp. 282–289.
7. C. Blaschke, A. Valencia, Can bibliographic pointers for known biological data be found automatically? protein interactions as a case study, Comparative and Functional Genomics 2 (2001) 196–206.
8. C. Blaschke, A. Valencia, The frame-based module of the Suiseki information extraction system, IEEE Intelligent Systems 17 (2002) 14–20.
9. R. Bunescu, R. Ge, R. J. Kate, E. M. Marcotte, R. J. Mooney, A. K. Ramani, Y. W. Wong, Comparative experiments on learning information extractors for proteins and their interactions, Artificial Intelligence in Medicine (special issue on Summarization and Information Extraction from Medical Documents) 33 (2) (2005) 139–155.
10. V. N. Vapnik, Statistical Learning Theory, John Wiley & Sons, New York, 1998.
11. N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, Cambridge University Press, 2000.
12. National Institute of Standards and Technology, ACE – Automatic Content Extraction, http://www.nist.gov/speech/tests/ace (2000).
13. M. Steedman, The Syntactic Process, MIT Press, Cambridge, MA, 2000.
14. J. Hockenmaier, M. Steedman, Generative models for statistical parsing with combinatory categorial grammar, in: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002), Philadelphia, PA, 2002, pp. 335–342.
15. R. C. Bunescu, R. J. Mooney, A shortest path dependency kernel for relation extraction, in: Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP-05), Vancouver, BC, 2005, pp. 724–731.

16. M. J. Collins, Three generative, lexicalised models for statistical parsing, in: Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL-97), 1997, pp. 16–23.
17. A. Culotta, J. Sorensen, Dependency tree kernels for relation extraction, in: Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04), Barcelona, Spain, 2004, pp. 423–429.
18. D. Zelenko, C. Aone, A. Richardella, Kernel methods for relation extraction, Journal of Machine Learning Research 3 (2003) 1083–1106.

Book Chapter in *Natural Language Processing and Text Mining*. Editors: Anne Kao and Steve Poteet. November 2006. Springer.

1 Mining Diagnostic Text Reports by Learning to Annotate Knowledge Roles

Eni Mustafaraj, Martin Hoof, and Bernd Freisleben

1.1 Introduction

Several tasks approached by using text mining techniques, like text categorization, document clustering, or information retrieval, operate on the document level, making use of the so called bag-of-words model. Other tasks, like document summarization, information extraction, or question answering, have to operate on the sentence level, in order to fulfill their specific requirements. While both groups of text mining tasks are typically affected by the problem of data sparsity, this is more accentuated for the latter group of tasks. Thus, while the tasks of the first group can be tackled by statistical and machine learning methods based on a bag-of-words approach alone, the tasks of the second group need natural language processing (NLP) at the sentence or paragraph level in order to produce more informative features.

Another issue common to all previously mentioned tasks is the availability of labeled data for training. Usually, for documents in real world text mining projects, training data do not exist or are expensive to acquire. In order to still satisfy the text mining goals while making use of a small contingent of labeled data, several approaches in machine learning have been developed and tested: different types of active learning Jones et al. [2003], bootstrapping Ghani and Jones [2002], or a combination of labeled and unlabeled data Blum and Mitchell [1998]. Thus, the issue of the lack of labeled data turns into the issue of selecting an appropriate machine learning approach.

The nature of the text mining task as well as the quantity and quality of available text data are other issues that need to be considered. While some text mining approaches can cope with data noise by leveraging the redundancy and the large quantity of available documents (for example, information retrieval on the Web), for other tasks (typically those restricted within a domain) the collection of documents might not possess such qualities. Therefore, more care is required for preparing such documents for the text mining task.

The previous observations suggest that performing a text mining task on new and unknown data requires handling all of the above mentioned issues, by combining and adopting different research approaches. In this paper, we present an approach to extract knowledge from text documents containing diagnostic problem solving situations in a technical domain (i.e. electrical engineering). In the proposed approach, we have combined techniques from several areas, including NLP, knowledge engineering, and machine learning to implement a learning framework for annotating cases with knowledge roles. The ultimate goal of the approach is to discover interesting problem solving situations (hereafter simply referred to as cases) that can be used by an experience management system to support new engineers during their working activities. However, the performed annotations facilitate the retrieval of cases on demand, allow collecting empirical domain knowledge, and can be formalized with the help of an ontology to also permit reasoning. The experimental results presented in the paper are based on a collection of 500 Microsoft Word documents written in German, amounting to about one million words. Several processing steps were required to achieve the goal of case annotation. In particular, we had to (a) transform the documents in an XML format, (b) extract paragraphs belonging to cases, (c) perform part-of-speech tagging, (d) perform syntactical parsing, (e) transform the results into XML representation for manual annotation, (f) construct features for the learning algorithm, and (g) implement an active learning strategy. Experimental results demonstrate the feasibility of the learning approach and a high quality of the obtained annotations.

The paper is organized as follows. In Section 1.2 we describe our domain of interest, the related collection of documents, and how knowledge roles can be used to annotate text. In Section 1.3 we consider work in natural language processing, especially frame semantics and semantic role labeling, emphasizing parallels to our task and identifying how resources and tools from these domains can be applied to perform annotation. Section 1.4 describes in detail all the preparatory steps for the process of learning to annotate cases. Section 1.5 evaluates the results of learning. Section 1.6 concludes the paper and outlines areas of future work.

1.2 Domain Knowledge And Knowledge Roles

1.2.1 Domain Knowledge

Our domain of interest is predictive maintenance in the field of power engineering, more specifically, the maintenance of insulation systems of high voltage rotating electrical machines. Since in many domains it is prohibitive to allow faults that could result in a breakdown of the system, components of the system are periodically or continuously monitored to look for changes in the expected behavior, in order to undertake predictive maintenance actions

when necessary. Usually, the findings related to the predictive maintenance process are documented in several forms: the measured values in a relational database; the evaluations of measurements/tests in diagnostic reports written in natural language; or the recognized symptoms in photographs. The focus of the work described here are the textual diagnostic reports.

In the domain of predictive maintenance, two parties are involved: the service provider (the company that has the know-how to perform diagnostic procedures and recommend predictive maintenance actions) and the customer (the operator of the machine). As part of their business agreement, the service provider submits to the customer an *official diagnostic report*. Such a report follows a predefined structure template and is written in syntactically correct and parsimonious language. In our case, the language is German.

A report is organized into many sections: summary, reason for the inspection, data of the inspected machine, list of performed tests and measurements, evaluations of measurement and test results, overall assessment and recommendations, as well as several attachments with graphical plots of numerical measurements or photographs of damaged parts.

From a diagnostic point of view, the most important information is found in the evaluations of the measurements and tests performed. As a demonstration, consider the two excerpts in Figure 1.1 (originating from English documents for non-German speaking customers).

At $1.9U_N (= 30kV)$, an insulation breakdown occurred on the upper bar of the slot $N^\circ 18$, at the slot exit on the NDE side. The breakdown indicates that the bar insulation is seriously weakened. This may be caused by intense discharges due to a malfunction of the slot anti-corona protection.	The measured bypass currents are in a relatively high range indicating a certain surface conductivity. This is due to the fact that the motor was stored in cold area before it was moved to the high voltage laboratory where the temperature and humidity was much higher so that a certain degree of condensation could occur on the surface of the winding.

Fig. 1.1. Excerpts from two evaluations of isolation current measurements.

As it is often the case with diagnosis, while the quantities that are measured or the components that are inspected are the same, the findings depend on a series of contextual factors, and the reasons for these findings could be quite unique (as the examples of Figure 1.1 demonstrate). Usually, human experts need many years of field experience to gain a degree of expertise that allows them to handle any situation. The goal of our project is to mine the text documents for relevant pieces of knowledge acquired during diagnostic problem solving situations.

1.2.2 Domain Concepts

In some text mining applications, such as text categorization or information retrieval, the goal is often to discover terms specific to the domain that could

be used as indices for organizing or retrieving information. Indeed, the excerpts of Figure 1.1 contain several of such domain-specific terms: *insulation*, *discharge*, *slot anti-corona protection*, *conductivity*, or *winding*. Still, using these terms as indices or keywords for representing the documents does not contribute to the purpose of our intended application, which is to find knowledge that supports diagnostic problem solving. To exemplify, consider the sentences in Figure 1.2:

1) The calculated **insulating resistance** values lay in the safe operating area.
2) Compared to the last examination, lower values for the **insulating resistance** were ascertained, due to dirtiness at the surface.

Fig. 1.2. Two sentences with the same domain concept shown in boldface.

In both sentences, the domain concept *insulating resistance* is found, but from a diagnostic point of view only the second sentence is interesting, because it describes a possible cause for lower values. Thus, more than domain concepts are needed to capture the knowledge expressed in the documents. Our solution to this problem is to label the text with semantic annotations expressed in terms of knowledge roles, which are introduced in the following subsection.

1.2.3 Knowledge Roles

Knowledge roles are a concept introduced in CommonKADS Schreiber et al. [2000], a knowledge engineering methodology for implementing knowledge-based systems. More specifically, knowledge roles are abstract names that refer to the role a domain concept plays when reasoning about a knowledge task. Such tasks are, for example, diagnosis, assessment, monitoring, or planning. Although these tasks are found in many domains, their description in CommonKADS is domain-independent. Thus, when describing a diagnosis task, knowledge roles like `finding`, `symptom`, `fault`, `parameter`, or `hypothesis` would be used.

Indeed, if we consider again the sentences in Figure 1.2, it is reasonable to represent the second sentence with knowledge roles as shown in Figure 1.3:

Knowledge Role	**Text Phrase**
Observed Object:	insulating resistance
Symptom:	lower values
Cause:	dirtiness at the surface

Fig. 1.3. Knowledge roles for sentence 2 of Figure 1.2.

Such a representation can have several advantages. Given a certain value of an *Observed Object*, a list of *Symptoms* that should be checked during the diagnosis could be retrieved. Or, given a certain *Symptom*, possible *Causes* for it could be listed, and so forth.

Understandably, we are interested in performing the text annotation with knowledge roles automatically. To achieve this goal, we draw on research in natural language understanding as described in Section 1.3.

It might be argued that one could simply use a combination of keywords to retrieve the information. For example, for sentences like that in Figure 1.2, one might write a query as below:

[low | small | high | large] && [value] && [insulating resistance]

for retrieving symptoms. Or one can search for:

[due to] | [caused by] | [as a result of] ...

to retrieve sentences containing causes. While this approach may be appealing and in some occasions even successful, there are several reasons why it could not be applied in our application:

- A large number of words (adjectives, nouns, adverbs, or verbs) can be used to describe changes (considered as symptoms in our domain), and no one can know beforehand which of them is used in the text.
- While verbs are very important for capturing the meaning of a sentence, they also abound in numbers. For example, to express an observation, any of the following verbs can be used: *observe*, *detect*, *show*, *exhibit*, *recognize*, *determine*, *result in*, *indicate*, etc. Furthermore, adverbs and negations can change their meaning and therefore need to be considered. Thus, instead of using verbs as keywords, we use them to bootstrap the annotating process, and incorporate them within semantic frames, like the frame *Observation* for the group above.
- Often, meaning emerges from the relation between different words, instead of the words separately, and this is exactly what we encountered in the diagnostic cases.

The knowledge roles used for annotating cases are abstract constructs in knowledge engineering, defined independently of any natural language constructs. Thus, a contribution of this work lies in trying to bridge the gap between knowledge roles and the natural language constructs whose meaning they capture. For this purpose, frame semantics, as described in the next section, is an ideal place to start.

1.3 Frame Semantics and Semantic Role Labeling

1.3.1 Frame Semantics

In frame semantics theory Fillmore [1976], a frame is a "script-like conceptual structure that describes a particular type of situation, object, or event and the participants involved in it" Ruppenhofer et al. [2005]. Based on this theory, the Berkeley FrameNet Project[1] is creating an online lexical resource for the English language by annotating text from the 100 million words British National Corpus.

The structure of a frame contains lexical units (pairs of a word with its meaning), frame elements (semantic roles played by different syntactic dependents), as well as annotated sentences for all lexical units that evoke the frame. An example of a frame with its related components is shown in Figure 1.4.

Annotation of text with frames and roles in FrameNet has been performed manually by trained linguists. An effort to handle this task automatically is being carried out by research in semantic role labeling, as described in the next subsection.

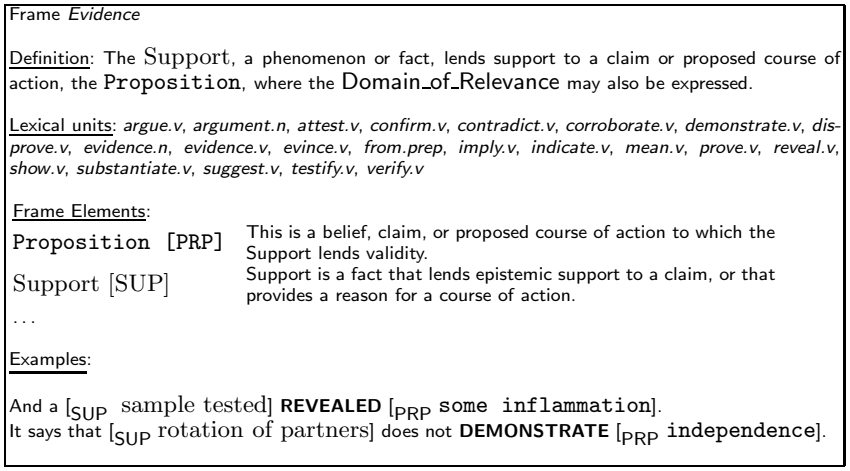
1.3.2 Semantic Role Labeling

Automatic labeling of semantic roles was introduced in Gildea and Jurafsky [2002]. In this work, after acknowledging the success of information extraction systems that try to fill in domain-specific frame-and-slot templates (see Section 1.3.4), the need for semantic frames that can capture the meaning of text independently of the domain was expressed. The authors envision that the semantic interpretation of text in terms of frames and roles would contribute to many applications, like question answering, information extraction, semantic dialogue systems, as well as statistical machine translation or automatic text summarization, and finally also to text mining.

After this initial work, research on semantic role labeling (SRL) has grown steadily, and in the years 2004 and 2005 Carreras and Màrquez [2004, 2005] a shared task at the CoNLL[2] was defined, in which several research institutions compared their systems. In the meantime, besides FrameNet, another corpus with manually annotated semantic roles has been prepared, PropNet Palmer and Gildea [2005], which differs from FrameNet in the fact that it has general semantic roles not related to semantic frames. PropNet is also the corpus used for training and evaluation of research systems on the SRL shared task. A similar corpus to FrameNet for the German language is been created by the Salsa project Erk et al. [2003a], and a discussion on the differences and similarities among these three projects is found in Ellsworth et al. [2004].

SRL is approached as a learning task. For a given target verb in a sentence, the syntactic constituents expressing semantic roles associated to this

[1] http://framenet.icsi.berkeley.edu/
[2] Conference of Natural Language Learning


Frame *Evidence*

Definition: The Support, a phenomenon or fact, lends support to a claim or proposed course of action, the Proposition, where the Domain_of_Relevance may also be expressed.

Lexical units: *argue.v*, *argument.n*, *attest.v*, *confirm.v*, *contradict.v*, *corroborate.v*, *demonstrate.v*, *disprove.v*, *evidence.n*, *evidence.v*, *evince.v*, *from.prep*, *imply.v*, *indicate.v*, *mean.v*, *prove.v*, *reveal.v*, *show.v*, *substantiate.v*, *suggest.v*, *testify.v*, *verify.v*

Frame Elements:

Proposition [PRP]	This is a belief, claim, or proposed course of action to which the Support lends validity.
Support [SUP]	Support is a fact that lends epistemic support to a claim, or that provides a reason for a course of action.
...	

Examples:

And a [SUP sample tested] **REVEALED** [PRP some inflammation].
It says that [SUP rotation of partners] does not **DEMONSTRATE** [PRP independence].

Fig. 1.4. Information on the frame *Evidence* from FrameNet.

verb need to be identified and labeled with the right roles. SRL systems usually divide sentences word-by-word or phrase-by-phrase and for each of these instances calculate many features creating a feature vector. The feature vectors are then fed to supervised classifiers, such as support vector machines, maximum entropy, or memory-based learners. While adapting such classifiers to perform better on this task could bring some improvement, better results can be achieved by constructing informative features for learning. A thorough discussion of different features used for SRL can be found in Gildea and Jurafsky [2002], Pradhan et al. [2005].

1.3.3 Frames and Roles for Annotating Cases

On the one hand, in knowledge engineering there are knowledge tasks and knowledge roles to represent knowledge; on the other hand, in natural language understanding there are semantic frames and semantic roles to represent meaning. When knowledge related to a knowledge task (like diagnosis) is represented by natural language, it is reasonable to expect that some knowledge roles will map to some semantic roles. The question is how to find these mappings, and more importantly, how to label text with these roles?

A knowledge task like diagnosis or monitoring is not equivalent to a semantic frame. The former are more complex and abstract, and can usually be divided into several components, which in turn can be regarded equivalent to semantic frames. By analyzing the textual episodes of diagnostic evaluations, we noticed that they typically contain a list of observations, explanations based on evidence, and suggestions to perform some activities. Thus, we consulted FrameNet for frames like Observation, Change, Evidence, or Activity. Indeed, these frames are all present in FrameNet, for example, Activity is

present in 10 subframes, or different meanings of Change are captured in 21 frames. The frame Evidence was shown in Figure 1.4, and besides the two roles of Proposition and Support, it has also roles for Degree, Depictive, Domain_of_Relevance, Manner, Means, and Result. When one carefully reads the definition of the roles Proposition and Support and looks at the examples (Figure 1.4), one can conclude that Proposition is similar to Cause and Support to Symptom in a diagnosis task.

The problem is to determine which frames to look for, given that there are currently more than six hundred frames in FrameNet. The key are the lexical units related to each frame, usually verbs. Starting with the verbs, one gets to the frames and then to the associated roles. This is also the approach we follow. We initially look for the most frequent verbs in our corpus, and by consulting several sources (since the verbs are in German), such as im Walde [2003], VerbNet[3], and FrameNet, we connect every verb with a frame, and try to map between semantic roles in a frame and knowledge roles we are interested in. One could also use the roles of FrameNet, but they are linguistically biased, and as such are not understandable by domain users that will annotate training instances for learning (a domain user would directly know to annotate *Cause*, but finds *Proposition* somehow confusing.)

In this work, FrameNet was only used as a lexical resource for consultation, that is, to find out which frames are evoked by certain lexical units, and what the related semantic roles are. Since the language of our corpus is German, we cannot make any statements about how useful the FrameNet frames could be to a learning system based on English annotated data corresponding to the defined frames.

Finally, it should be discussed why such an approach to annotating text cases with frames and roles could be beneficial to text mining. For the purpose of this discussion, consider some facts from the introduced domain corpus. During the evaluation of the learning approach, we manually annotated a subcorpus of unique sentences describing one specific measurement (high-voltage isolation current). In the 585 annotated sentences, the frame Evidence was found 152 times, 84 times evoked by the verb *zurückführen* (trace back to), 40 times by the verb *hindeuten* (point to), and 28 times by 9 other verbs. Analyzing the text annotated with the role *Cause* in the sentences with *zurückführen*, 27 different phrases expressing causes of anomalies pointed to by the symptoms were found. A few of these expressions appeared frequently, some of them occasionally, some others rarely. In Table 1.1, some of these expressions are shown.

If for every sentence with the frame *Evidence* the text annotated with *Symptom* and *Cause* is extracted, this text can then be processed further with other text mining techniques for deriving domain knowledge, which is not available in any of the analyzed texts alone. For example, one could get answers to questions like: which are the most frequent symptoms and what

[3] http://www.cis.upenn.edu/~bsnyder3/cgi-bin/search.cgi

Table 1.1. Some phrases annotated with the role *Cause*.

German Phrase	English Translation	Frequency
Verschmutzungseinflüsse	influences of pollution	10
leitende Verschmutzungen	conducting pollutions	8
Ionisation in Klemmenbereich	ionization in the terminal area	3
äussere Entladungen	external discharges	1

causes can explain them; what problems (i.e. causes) do appear frequently in a specific type of machine, etc. Thus, such an annotation with frames and roles preprocesses text by generating very informative data for text mining, and it can also be used in the original form for information retrieval. Still, such an approach makes sense in those cases when text contains descriptions of repetitive tasks, which are then expressed by a small number of underlying semantic frames. Since data and text mining try to extract knowledge from data of the same nature in the same domain, we find that annotation of text with knowledge roles could be a valuable approach.

Before explaining in detail the process of learning to automatically annotate text with knowledge roles (based on the SRL task) in Section 4, we briefly discuss the related field of information extraction.

1.3.4 Information Extraction

Information extraction (IE), often regarded as a restricted form of natural language understanding, predates research in text mining, although today, IE is seen as one of the techniques contributing to text mining Weiss et al. [2004]. Actually, the purpose of IE is very similar to what we are trying to achieve with role annotation. In IE it is usually known in advance what information is needed, and part of text is extracted to fill in slots of a predefined template. An example, found in Mooney and Bunescu [2005], is the job posting template, where, from job posting announcements in Usenet, text to fill slots like: title, state, city, language, platform, etc. is extracted and stored in a database for simpler querying and retrieval.

Usually, methods used by IE have been based on shallow NLP techniques, trying to extract from a corpus different types of syntactic rules that match syntactic roles to semantic categories, as for example in Rillof and Schelzenbach [1998].

With the advances in NLP and machine learning research, IE methods have also become more sophisticated. Actually, SRL can also be seen as a technology for performing information extraction, in those cases when text is syntactically and semantically more demanding and expressive. All these technologies are intended to be used for extracting knowledge from text, despite their differences in implementation or scope.

1.4 Learning to Annotate Cases with Knowledge Roles

To perform the task of learning to annotate cases with knowledge roles, we implemented a software framework, as shown in Figure 1.5. Only the preparation of documents (described in Section 1.4.1) is performed outside of this framework. In the remainder of the section, every component of the framework is presented in detail.

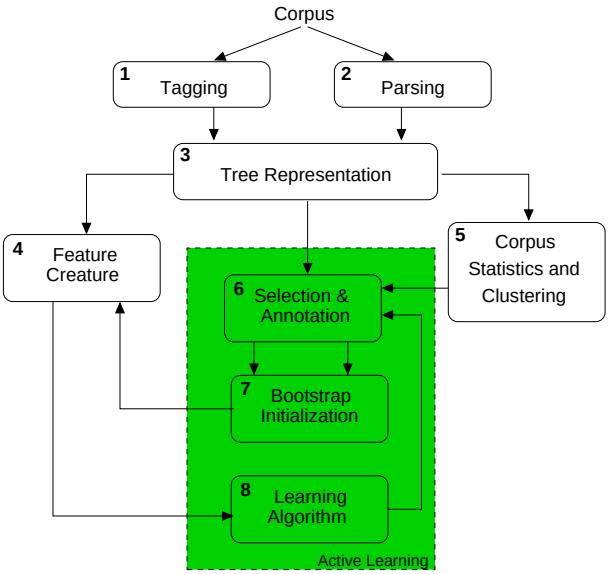


Fig. 1.5. The Learning Framework Architecture.

1.4.1 Document Preparation

In Section 1.2.1 it was mentioned that our documents are official diagnostic reports hierarchically structured in several sections and subsections, written by using MS® Word. Actually, extracting text from such documents, while preserving the content structure, is a difficult task. In completing it we were fortunate twice. First, with MS® Office 2003 the XML based format WordML was introduced that permits storing MS® Word documents directly in XML. Second, the documents were originally created using a MS® Word document template, so that the majority of them had the same structure. Still, many problems needed to be handled. MS® Word mixes formatting instructions with content very heavily and this is reflected also in its XML format. In addition, information about spelling, versioning, hidden template elements, and so on are also stored. Thus, one needs to explore the XML output of

the documents to find out how to distinguish text and content structure from unimportant information. Such a process will always be a heuristic one, depending on the nature of the documents. We wrote a program that reads the XML document tree, and for each section with a specified label (from the document template) it extracts the pure text and stores it in a new XML document, as the excerpt in Figure 1.6 shows.

```
<section title="Measurements">
  <subsection title="Stator_Winding">
    <measurement title="Visual_Control">
      <submeasurement title="Overhang_Support">
        <evaluation>
          Die Wickelkopfabsttzung AS und NS befand sich in einem ...
        </evaluation>
        <action>Keine</action>
      </submeasurement>
...
```

Fig. 1.6. Excerpt of the XML representation of the documents.

Based on such an XML representation, we create subcorpora of text containing measurement evaluations of the same type, stored as paragraphs of one to many sentences.

1.4.2 Tagging

The part-of-speech (POS) tagger (TreeTagger[4]) that we used Schmid [1995] is a probabilistic tagger with parameter files for tagging several languages: German, English, French, or Italian. For some small problems we encountered, the author of the tool was very cooperative in providing fixes. Nevertheless, our primary interest in using the tagger was not the POS tagging itself (the parser, as is it shown in Section 1.4.3 performs tagging and parsing), but getting stem information (since the German language has a very rich morphology) and dividing the paragraphs in sentences (since the sentence is the unit of operation for the next processing steps).

The tag set used for tagging German is slightly different from that of English [5]. Figure 1.7 shows the output of the tagger for a short sentence [6].

As indicated in Figure 1.7, to create sentences it suffices to find the lines containing: `". \$. ."` (one sentence contains all the words between two such lines). In general, this is a very good heuristic, but its accuracy depends on the nature of the text. For example, while the tagger correctly

[4] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger
[5] http://www.ims.uni-stuttgart.de/projekte/corplex/TagSets/stts-table.html
[6] Translation: A generally good external winding condition is present.

Es	PPER	es
liegt	VVFIN	liegen
insgesamt	ADV	insgesamt
ein	ART	ein
guter	ADJA	gut
äusserer	ADJA	äuβer
Wicklungszustand	NN	<unknown>
vor	PTKVZ	vor
.	$.	.

Fig. 1.7. A German sentence tagged with POS-tags by TreeTagger.

tagged abbreviations found in its list of abbreviations (and the list of abbreviations can be customized by adding abbreviations common to the domain of the text), it got confused when the same abbreviations were found inside parentheses, as the examples in Figure 1.8 for the word 'ca.' (circa) show.

If such phenomena occur often, they become a problem for the further correct processing of sentences, although one becomes aware of such problems only in the course of the work. A possible solution in such cases is to use heuristics to replace erroneous tags with correct ones for the types of identified errors.

an	APR	an
ca.	ADV	ca.
50	CARD	50
%	NN	%

(	$(	(
ca	NE	<unknown>
.	$.	.
20	CARD	20

Fig. 1.8. Correct and erroneous tagging for the word 'ca.'

The more problematic issue is that of words marked with the stem <unknown>. Actually, their POS is usually correctly induced, but we are specifically interested in the stem information. The two reasons for an <unknown> label are a) the word has been misspelled and b) the word is domain specific, and as such not seen during the training of the tagger. On the positive side, selecting the words with the <unknown> label directly creates the list of domain specific words, useful in creating a domain lexicon.

A handy solution for correcting spelling errors is to use a string similarity function, available in many programming language libraries. For example, the Python language has the function "get_close_matches" in its "difflib" library. An advantage of such a function is having as a parameter the degree of similarity between strings. By setting this value very high (between 0 and 1) one is sure to get really similar matches if any at all.

Before trying to solve the problem of providing stems for words with the <unknown> label, one should determine whether the stemming information substantially contributes to the further processing of text. Since we could not know that in advance, we manually provided stems for all words labeled as

<unknown>. Then, during the learning process we performed a set of experiments, where: a) no stem information at all was used and b) all words had stem information (tagger + manually created list of stems). Table 1.2 summarizes the recall and precision of the learning task in each experiment.

Table 1.2. Results of experiments for the contribution of stem information on learning.

Experiment	Recall	Precision
a) no stems (only words)	90.38	92.32
b) only stems	91.29	93.40

These results show approximately 1% improvement in recall and precision when stems instead of original words are used. We can say that at least for the learning task of annotating text with knowledge roles stem information is not necessarily important, but this could also be due to the fact that a large number of other features (see Section 1.4.5) besides words are used for learning.

Still, the reason for having a list of stems was not in avoiding more data due to word inflections, but in capturing the word composition, a phenomenon typical for the German language. For example, all the words in the first row of Table 1.3 are compound words that belong to the same semantic category identified by their last word 'wert' (value), i.e. they all denote values of different measured quantities, and as such have a similar meaning. This similarity cannot be induced if one compares the words in the original form, something possible by comparing the word representations of the second row.

Table 1.3. Original words (first row), words composed of stems (second row).

Ableitstrom*werte*, Gesamtstrom*werte*, Isolationswiderstands*werte*, Isolationsstrom*werte*, Kapazitäts*werte*, Ladestrom*werte*, Strom*wert*en, Verlustfaktoranfangs*wert*, etc.
Ableit-Strom-Wert, Gesamt-Strom-Wert, Isolation-Widerstand-Wert, Isolation-Strom-Wert, Kapazität-Wert, Lade-Strom-Wert, Strom-Wert, Verlustfaktor-Anfang-Wert, etc.

Unfortunately, there are only a few tools available for morphological analysis of German words. We tried Morphy Lezius [2000], which is publicly available, but it was not able to analyze any of our domain-specific words. Therefore, we had to perform this task by hand.

1.4.3 Parsing

Syntactical parsing is one of the most important steps in the learning framework, since the produced parse trees serve as input for the creation of features used for learning. Since we are interested in getting qualitative parsing results, we experimented with three different parsers: the Stanford parser (Klein 2005), the BitPar parser Schmid [2004], Schiehlen [2004], and the Sleepy parser Dubey [2003]. What these parsers have in common is that they all are based on unlexicalized probabilistic context free grammars (PCFG) Manning and Schütze [1999], trained on the same corpus of German, Negra[7] (or its superset Tiger[8]), and their source code is publicly available. Still, they do differ in the degree they model some structural aspects of the German language, their annotation schemas, and the information included in the output. Figure 1.9 shows the output of the same sentence[9] parsed by each parser, and in the following, we discuss each of them.

Stanford Parser - The Stanford parser is an ambitious project that tackles the task of generating parse trees from unlabeled data independently of the language. For the moment, the parser is distributed with parameter files for parsing English, German, and Chinese. We tested the parser on our data and noticed that the POS tags were often erroneously induced (in the sentence with only 8 words of Figure 1.9 there are 3 such errors—CARD tags for 2 nouns and 1 adjective), which then resulted in erroneous parse trees. But, in those cases when the tagging was performed correctly, the parse trees were also correct. Still, the parser could not parse long sentences, perhaps due to the fact that it was trained in the part of the Negra corpus with sentences having up to 10 words. Trying the parser with long English sentences instead, produced excellent results. We concluded that at this phase of implementation, the Stanford parser could not be used with our corpus of German sentences that contain an average of up to 18 words per sentence.

BitPar Parser - This parser is composed of two parts, the parser itself Schmid [2004] and the parameter files (chart rules, lexicon, etc.) from Schiehlen [2004]. Published experimental results claim robust performance, due to the use of sophisticated annotation and transformation schemata for modeling grammars. Another advantage of the parser is that its lexicon can be extended very easily with triples of domain-dependent words, their tags, their frequency counts in corpus, thus avoiding the tagging errors typical for unlexicalised parsers. These tagging errors damage the parse results, as can be seen from the results of the Stanford parser. Our critique for the described BitPar is that it usually produces trees with more nodes than

[7] http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/
[8] http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERCorpus/
[9] English translation: "On NS were detected circa 5 torn wedge's safety bands."

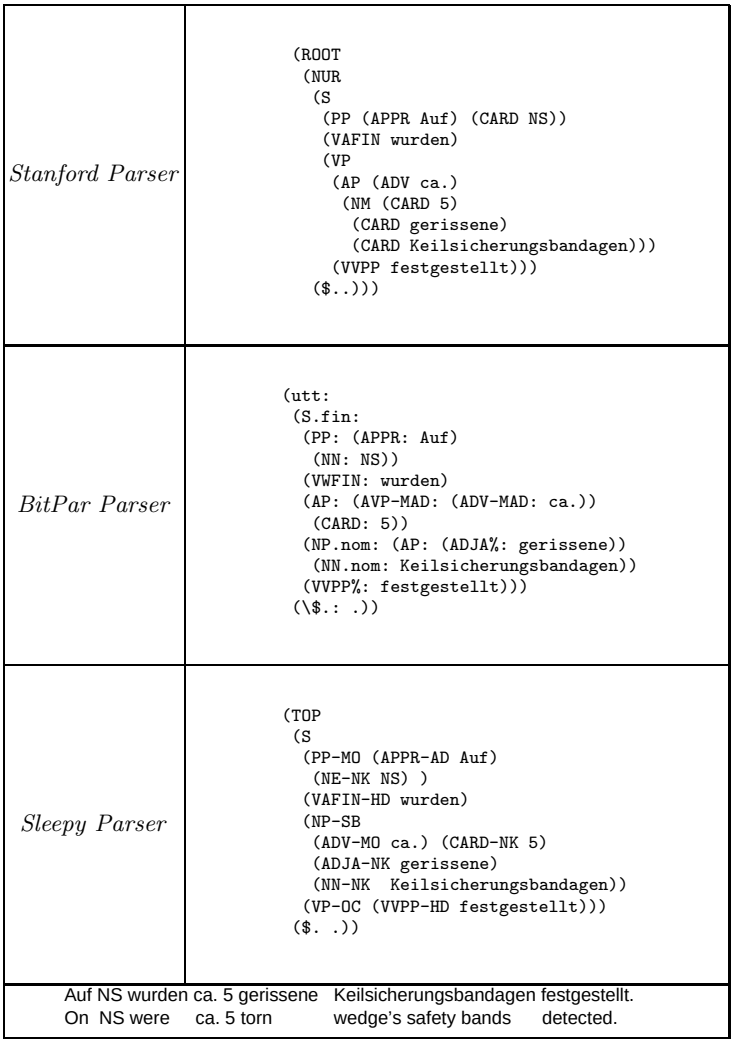


Fig. 1.9. Parsing output of the same sentence from the three parsers

the other parsers and the annotation of nodes contains specialized linguistic information, not very appropriate for creating features for learning.

Sleepy Parser - This parser has been specifically tuned for the German language, and while it is a statistical parser like the others, it uses different annotation schemas and incorporates grammatical functions (SB–subject, OC–clausal object, MO–modifier, HD–head, etc.) or long-distance dependencies between terms. In constrast to the two other parsers, it also has a highly tuned suffix analyzer for guessing POS tags Dubey [2005], which contributes to more accurate tagging results than the other parsers, al-

though some domain-dependent words are not always correctly tagged. Erroneous parsing is also encountered for very long sentences.

Chossing a Parser

All the tested parsers make errors during parsing. In the end, the criteria upon which we based our choice of the parser were: speed and output information. Sleepy was the fastest and had the most informative output (it prints the log value expressing the likelihood of parsing, and it labels the majority of nodes with their grammatical function). Actually, choosing a parser upon these criteria instead of the accuracy of parsing could be regarded as inappropriate. Our justification is that a metric to measure the accuracy of parsing on new data does not exist. These parsers have all been trained on the same corpus, and at least the two German parsers tuned up to the point where their results are almost the same. Thus, *a priori* their expected accuracy in a new corpus should be equal, and accuracy is not a criterion for choosing one over the other. Given the difficulty of evaluating the accuracy of the parse trees and their presumed similarity, we based the choice of parser on the qualities that contributed most to our task, namely speed and informative output.

1.4.4 Tree Representation

The bracketed parse tree and the stem information of tagging serve as input for the step of creating a tree data structure. The tree is composed of terminals (leaf nodes) and non-terminals (internal nodes), all of them known as constituents of the tree. For export purposes as well as for performing exploration or annotation of the corpus, the tree data structures are stored in XML format, according to a schema defined in the TigerSearch[10] tool. The created tree, when visualized in TigerSearch, looks like the one shown in Figure 1.10[11]. The terminals are labeled with their POS tags and also contain the corresponding words and stems; the inside nodes are labeled with their phrase types (NP, PP, etc.); and the branches have labels, too, corresponding to the grammatical functions of the nodes. The XML representation of a portion of the tree is shown in Figure 1.11.

1.4.5 Feature Creation

Features are created from the parse tree of a sentence. A feature vector is created for every constituent of the tree, containing some features unique to the constituent, some features common to all constituents of the sentence, and some others calculated with respect to the target constituent (the predicate verb).

[10] http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/

[11] English translation: "... irregularities, which point to a not anymore continuous steering of voltage in the area of the winding head."

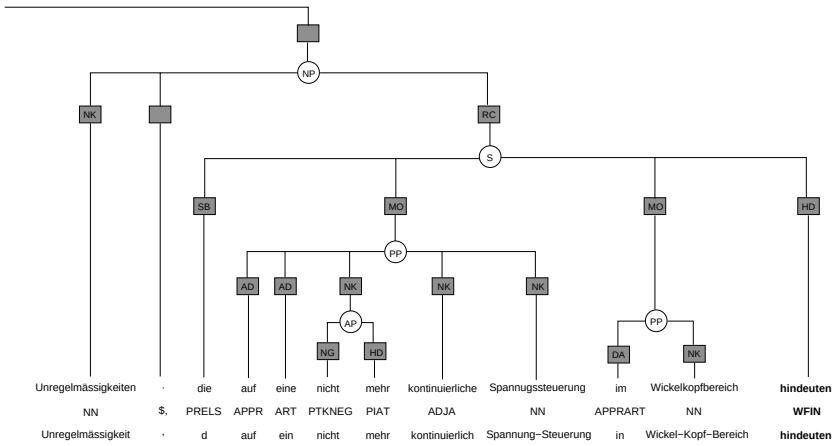


Fig. 1.10. Representation of a parsed tree in the TigerSearch tool. Due to space reasons, only a branch of the tree is shown.

```
...
  <t lemma="Spannung-Steuerung" word="Spannungssteuerung" pos="NN"
       id="sentences._108_28" />
  <t lemma="in" word="im" pos="APPRART"
       id="sentences._108_29" />
  <t lemma="Wickel-Kopf-Bereich" word="Wickelkopfbereich" pos="NN"
       id="sentences._108_30" />
  <t lemma="hindeuten" word="hindeuten" pos="VVFIN" id="sentences._108_31" />
 </terminals>
 <nonterminals>
  <nt id="sentences._108_500" cat="PP">
   <edge idref="sentences._108_3" label="NK" />
   <edge idref="sentences._108_2" label="DA" />
   <edge idref="sentences._108_1" label="DA" />
  </nt>
 ...
```

Fig. 1.11. XML representation of a portion of the parse tree from Figure 1.10.

A detailed linguistic description of possible features used by different research systems for the SRL task is found in Pradhan et al. [2005]. In this subsection, we only list the features used in our system and give example values for the leaf node `Spannungssteuerung` of the parse tree in Figure 1.10.

Phrase type **NN**
Grammatical function **NK**
Terminal (is the constituent a terminal or non-terminal node?) **1**
Path (path from the target verb to the constituent, denoting u(up) and d(down) for the direction) **uSdPPd**
Grammatical path (like Path, but instead of node labels, branches labels are considered) **uHDdMOdNK**
Path length (number of branches from target to constituent) **3**
Partial path (path to the lowest common ancestor between target and constituent) **uPPuS**
Relative Position (position of the constituent relative to the target) **left**
Parent phrase type (phrase type of the parent node of the constituent) **PP**
Target (lemma of the target word) **hindeuten**
Target POS (part-of-speech of the target) **VVFIN**
Passive (is the target verb passive or active?) **0**
Preposition (the preposition if the constituent is a PP) **none**
Head Word (for rules on head words refer to Collins [1999]) **Spannung-Steuerung**
Left sibling phrase type **ADJA**
Left sibling lemma **kontinuierlich**
Right sibling phrase type **none**
Right sibling lemma **none**
Firstword, Firstword POS, Lastword, Lastword POS (in this case, the constituent has only one word, thus, these features get the same values: Spannung-Steuerung and NN. For non-terminal constituents like PP or NP, first word and last word will be different.)
Frame (the frame evoked by the target verb) **Evidence**
Role (this is the class label that the classifier will learn to predict. It will be one of the roles related to the frame or none, for an example refer to Figure 1.12.) **none**

If a sentence has several clauses where each verb evokes a frame, the feature vectors are calculated for each evoked frame separately and all the vectors participate in the learning.

1.4.6 Annotation

To perform the manual annotation, we used the Salsa annotation tool (publicly available) Erk et al. [2003b]. The Salsa annotation tool reads the XML representation of a parse tree and displays it as shown in Figure 1.12. The user has the opportunity to add frames and roles as well as to attach them to a desired target verb. In the example of Figure 1.12 (the same sentence of Figure 1.10), the target verb *hindeuten* (point to) evokes the frame Evidence, and three of its roles have been assigned to constituents of the tree. Such an assignment can be easily performed per point-and-click. After this process, an element <frames> is added to the XML representation of the sentence, containing information about the frame. Excerpts of the XML code are shown in Figure 1.13.

1.4.7 Active Learning

Research in IE has indicated that using an active learning approach for acquiring labels from a human annotator has advantages over other approaches of selecting instances for labeling Jones et al. [2003]. In our learning framework, we have also implemented an active learning approach. The possibilities for designing an active learning strategy are manifold; the one we have implemented uses a committee-based classification scheme that is steered by corpus statistics. The strategy consists of the following steps:

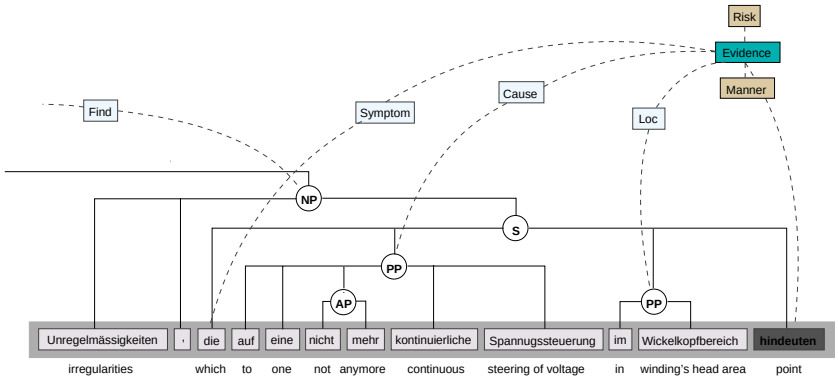


Fig. 1.12. Annotation with roles with the Salsa tool.

```
<frames>
 <frame name="Evidence" id="sentences._108__f1">
  <target><fenode idref="sentences._108_31"/></target>
   <fe name="Symptom" id="sentences._108_f1_e1">
    <fenode idref="sentences._108_22"/>
   </fe>
   <fe name="Cause" id="sentences._108__f1_e2">
    <fenode idref="sentences._108_509"/>
   </fe>
    <fe name="Loc" id="sentences._108__f1_e5">
    <fenode idref="sentences._108_510"/>
   </fe>
...
```

Fig. 1.13. XML Representation of an annotated frame.

a) Divide the corpus in clusters of sentences with the same target verb. If a cluster has fewer sentences than a given threshold, group sentences with verbs evoking the same frame into the same cluster.
b) Within each cluster, group the sentences (or clauses) with the same parse sub-tree together.
c) Select sentences from the largest groups of the largest clusters and present them to the user for annotation.
d) Bootstrap initialization: apply the labels assigned by the user to groups of sentences with the same parse sub-tree.
e) Train all the classifiers of the committee on the labeled instances; apply each trained classifier to the unlabeled sentences.
f) Get a pool of instances where the classifiers of the committee disagree and present to the user the instances belonging to sentences from the next largest clusters not yet manually labeled.
g) Repeat steps d)–f) a few times until a desired accuracy of classification is achieved.

In the following, the rationale behind choosing these steps is explained.

Steps a), b), c): In these steps, statistics about the syntactical structure of the corpus are created, with the intention of capturing its underlying distribution, so that representative instances for labeling can be selected.

Step d): This step has been regarded as applicable to our corpus, due to the nature of the text. Our corpus contains repetitive descriptions of the same diagnostic measurements on electrical machines, and often, even the used language has a repetitive nature. Actually, this does not mean that the same words are repeated (although often standard formulations are used, especially in those cases when nothing of value was observed). Rather, the kind of sentences used to describe the task has the same syntactical structure. As an example, consider the sentences shown in Figure 1.14.

[$_{PP}$ Im Nutaustrittsbereich] wurden [$_{NP}$ stärkere Glimmentladungsspuren] festgestellt.
In the area of slot exit stronger signs of corona discharges were detected.

[PP Bei den Endkeilen] wurde [NP ein ausreichender Verkeildruck] festgestellt.
At the terminals' end a sufficient wedging pressure was detected.

[$_{PP}$ An der Schleifringbolzenisolation] wurden [$_{NP}$ mechanische Beschädigungen] festgestellt.
On the insulation of slip rings mechanical damages were detected.

[$_{PP}$ Im Wickelkopfbereich] wurden [$_{NP}$ grossflächige Decklackablätterungen] festgestellt.
In the winding head area extensive chippings of the top coating were detected.

Fig. 1.14. Examples of sentences with the same structure.

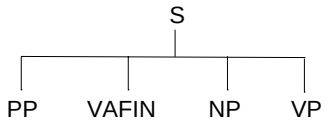


Fig. 1.15. Parse tree of the sentences in Figure 1.14.

What all these sentences have in common is the passive form of the verb *feststellen* (wurden festgestellt), and due to the subcategorization of this verb, the parse tree on the level of phrases is identical for all sentences, as indicated by 1.15. Furthermore, for the frame Observation evoked by the verb, the assigned roles are in all cases: NP—Finding, PP—Observed_Object. Thus, to bootstrap initialization, we assign the same roles to sentences with the same sub-tree as the manually labeled sentences.

Step e): The committee of classifiers consists of a maximum entropy (MaxEnt) classifier from Mallet McCallum [2002], a Winnow classifier from SNoW Carlson et al. [2004], and a memory-based learner (MBL) from TiMBL Daelemans et al. [2004]. For the MBL, we selected k=5 as the number of the nearest neighbours. The classification is performed as follows: if at least two classifiers

agree on a label, the label is accepted. If there is disagreement, the cluster of labels from the five nearest neighbours is examined. If the cluster is not homogenous (i.e. it contains different labels), the instance is included in the set of instances to be presented to the user for manual labeling.

Step f): If one selects new sentences for manual annotation only based on the output of the committee-based classifier, the risk of selecting outlier sentences is high Tang et al. [2002]. Thus, from the instances' set created by the classifier, we select those belonging to large clusters not manually labeled yet.

1.5 Evaluations

To evaluate this active learning approach on the task of annotating text with knowledge roles, we performed a series of experiments that are described in the following. It was explained in Section 1.4.1 that based on the XML structure of the documents we created subcorpora with text belonging to different types of diagnostic tests. After such subcorpora have been processed to create sentences, only unique sentences are retained for further processing (repetitive, standard sentences do not bring any new information, they only disturb the learning and therefore are discarded). Then, lists of verbs were created, and by consulting the sources mentioned in Section 1.3.3, verbs were grouped with one of the frames: Observation, Evidence, Activity, and Change. Other verbs that did not belong to any of these frames were not considered for role labeling.

1.5.1 Learning Performance on the Benchmark Datasets

With the aim of exploring the corpus to identify roles for the frames and by using our learning framework, we annotated two different subcorpora and then manually controlled them, to create benchmark datasets for evaluation. Some statistics for the manually annotated subcorpora are summarized in Table 1.4. Then, to evaluate the efficiency of the classification, we performed 10-fold cross-validations on each set, obtaining the results shown in Table 1.5, where recall, precision, and the $F_{\beta=1}$ measure are the standard metrics of information retrieval.

Table 1.4. Statistics for the benchmark datasets.

Subcorpus	Cases No.	Sentences No.	Unique Sentences No.	Annotated Roles No.
Isolation Current	491	1134	585	1862
Wedging System	453	775	602	1751

Table 1.5. Learning results for the benchmark datasets.

Subcorpus	Recall	Precision	$F_{\beta=1}$ measure
Isolation Current	0.913	0.934	0.92
Wedging System	0.838	0.882	0.86

We analyzed some of the classification errors and found that they were due to parsing anomalies, which had forced us in several occasions to split a role among several constituents.

1.5.2 Active Learning versus Uniform Random Selection

In order to evaluate the advantages of active learning, we compared it to the uniform random selection of sentences for manual annotations. Some results for both approaches are summarized in Table 1.6 and Table 1.7. Recall, precision, and $F_{\beta=1}$ measure were calculated after each iteration, in which 10 new sentences manually labeled were added to the training set. The results of active learning ($F_{\beta=1}$ measure) are 5–10 points better than those of random learning. For this experiment, the step d) of the active learning strategy was not applied, since it is very specific to our corpus.

Table 1.6. Random Learning Results.

Sentences No.	Recall	Precision	$F_{\beta=1}$ measure
10	0.508	0.678	0.581
20	0.601	0.801	0.687
30	0.708	0.832	0.765
40	0.749	0.832	0.788

Table 1.7. Active Learning Results.

Sentences No.	Recall	Precision	$F_{\beta=1}$ measure
10	0.616	0.802	0.697
20	0.717	0.896	0.797
30	0.743	0.907	0.817
40	0.803	0.906	0.851

1.5.3 Bootstrapping Based on Other Sets

During the annotation of the two benchmark datasets, we noticed that the two subcorpora, although different in nature (set_1: *Isolation Current* contains evaluations of numerical measurements performed on the three phases of the machine, set_2: *Wedging System* describes visual inspections on the wedging components of the machine) had very often the frame Observation or Change in common, while the frame Evidence appeared almost only in the first set, and the frame Activity almost always in the second. Thus, we tested whether text annotated with the same roles in one set could bootstrap the learning in the second, and the results are summarized in Table 1.8.

Table 1.8. Results for bootstrapping based on other labeled sets..

Training File	Testing File	Recall	Precision
Isolation Current	Wedging System	0.765	0.859
Wedging System	Isolation Current	0.642	0.737

We consider these results as very promising, since they give a hint at the possibility of using previously annotated text from other subcorpora to bootstrap the learning process, something that would alleviate the process of acquiring manual annotations for new text.

1.6 Conclusions

In this paper, we have presented an approach for extracting knowledge from text documents containing descriptions of knowledge tasks in a technical domain. Knowledge extraction in our approach is based on the annotation of text with knowledge roles (a concept originating in knowledge engineering), which we map to semantic roles found in frame semantics. The framework implemented for this purpose is based on deep NLP and active learning. Experiments have demonstrated a robust learning performance, and the obtained annotations were of high quality. Since our framework is inspired by and founded upon research in semantic role labeling (SRL), the results indicate that SRL could become a highly valuable processing step for text mining tasks.

In future work, we will consider the advantages of representing annotated text by means of knowledge roles and the related frames. Besides the previously explained uses for semantic retrieval of cases and the extraction of empirical domain knowledge facts, such a representation could also permit looking for potentially interesting relations in text and can be exploited to populate application and domain ontologies with lexical items.

1.7 Acknowledgements

The Insulation Competence Center of ALSTOM Ltd. Switzerland kindly permitted the use of the text documents for research purposes. Katrin Erk, Sebastian Pado, Amit Dubey, Sabine Schulte im Walde, Michael Schiehlen, and Helmut Schmid provided their linguistic tools and were an invaluable source of information and support. We are grateful to all of them.

References

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proc. of the Workshop on Computational Learning Theory, COLT '98, Madison, WI*, pages 92–100, 1998.

A. J. Carlson, C. M. Cumby, N. D. Rizzolo, J. L. Rosen, and D. Roth. SNoW: Sparse Network of Winnow. 2004. URL `http://l2r.cs.uiuc.edu/~cogcomp/software.php`.

X. Carreras and L. Màrquez. Introduction to the coNLL shared task: Semantic role labeling. In *Proc. of 8th Conference of Natural Language Learning*, pages 89–97, Boston, MA, 2004.

X. Carreras and L. Màrquez. Introduction to the coNLL-2005 shared task: Semantic role labeling. In *Proc. of 9th Conference of Natural Language Learning*, pages 152–165, Ann Arbor, MI, June 2005.

M. Collins. *Head-Driven Statistical Models for Natural Language Parsing.* PhD thesis, University of Pennsylvania, 1999.

W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. TiMBL: Tilburg Memory Based Learner. 2004. URL `http://ilk.uvt.nl/downloads/pub/papers/ilk0402.pdf`.

A. Dubey. *Statistical Parsing for German.* PhD thesis, University of Saarland, Germany, 2003.

A. Dubey. What to do when lexicalization fails: Parsing German with suffix analysis and smoothing. In *Proc. of 43rd Annual Meeting of ACL, Ann Arbor, MI*, pages 314–321, 2005.

M. Ellsworth, K. Erk, P. Kingsbury, and S. Padó. PropBank, SALSA, and FrameNet: How design determines product. In *Proc. of the LREC 2004 Workshop on Building Lexical Resources from Semantically Annotated Corpora, Lisbon, Portugal*, 2004.

K. Erk, A. Kowalski, and S. Padó. The Salsa annotation tool-demo description. In *Proc. of the 6th Lorraine-Saarland Workshop, Nancy, France*, pages 111–113, 2003a.

K. Erk, A. Kowalski, S. Padó, and M. Pinkal. Towards a resource for lexical semantics: A large German corpus with extensive semantic annotation. In *Proc. of 41st Annual Meeting of ACL, Saporo, Japan*, pages 537–544, 2003b.

C. J. Fillmore. Frame semantics and the nature of language. In *Annals of the New York Academy of Sciences: Conf. on the Origin and Development of Language and Speech*, volume 280, pages 20–32, 1976.

R. Ghani and R. Jones. A comparison of efficacy of bootstrapping of algorithms for information extraction. In *Proc. of LREC 2002 Workshop on Linguistic Knowledge Acquisition, Las Palmas, Spain*, 2002.

D. Gildea and D. Jurafsky. Automatic labeling of semantic roles. In *Computational Linguistics*, volume 23, pages 245–288, 2002.

S. Schulte im Walde. *Experiments on the Automatic Induction of German Semantic Verb Classes*. PhD thesis, Universität Stuttgart, Germany, 2003.

R. Jones, R. Ghani, T. Mitchell, and E. Riloff. Active learning for information extraction with multiple view features sets. In *Proc. of Adaptive Text Extraction and Mining, EMCL/PKDD-03, Cavtat-Dubrovnik, Croatia*, pages 26–34, 2003.

W. Lezius. Morphy - German morphology, part-of-speech tagging and applications. In *Proc. of 9th Euralex International Congress, Stuttgart, Germany*, pages 619–623, 2000.

C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA, 1999.

A. K. McCallum. MALLET: A machine learning for language toolkit, 2002. URL `http://mallet.cs.umass.edu`.

R. J. Mooney and R. Bunescu. Mining knowledge from text using information extraction. *SIGKDD Explor. Newsl.*, 7(1):3–10, 2005.

M. Palmer and D. Gildea. The proposition bank: An annotated corpus of semantic roles. In *Computational Linguistics*, volume 31, pages 71–106, 2005.

S. Pradhan, K. Hacioglu, V. Kruglery, W. Ward, J. H. Martin, and D. Jurafsky. Support vector learning for semantic argument classification. *Machine Learning Journal, Kluver Academic Publishers*, 59:1–29, 2005.

E. Rillof and M. Schelzenbach. An empirical approach to conceptual frame acquisition. In *Proc. of 6th Workshop on Very Large Corpora, Montreal, Canada*, pages 49–56, 1998.

J. Ruppenhofer, M. Ellsworth, M. R. L. Petruck, and C. R. Johnson. *FrameNet: Theory and Practice*. 2005. URL `http://framenet.icsi.berkeley.edu/book/book.html`.

M. Schiehlen. Annotation strategies for probabilistic parsing in German. In *Proc. of CoLing'04, Geneva, Switzerland*, 2004.

H. Schmid. Efficient parsing of highly ambiguous context-free grammars with bit vectors. In *Proc. of CoLing'04, Geneva, Switzerland*, 2004.

H. Schmid. Improvement in part-of-speech tagging with an application to German. In *Proc. of the ACL SIGDAT-Workshop, Dublin, Ireland*, pages 47–50, 1995.

G. Schreiber, H. Akkermans, A. Anjewierden, R. deHoog, N. Shadbolt, W. VandeVelde, and B. Wielinga. *Knowledge Engineering and Manage-*

ment: The CommonKADS Methodology. The MIT Press, Cambridge, MA, 2000.

M. Tang, X. Luo, and S. Roukos. Active learning for statistical natural language parsing. In *Proc. of the ACL 40th Anniversary Meeting, Philadelphia, PA*, pages 120–127, 2002.

S. Weiss, N. Indurkhya, T. Zhang, and F. Damerau. *Text Mining: Predictive Methods for Analyzing Unstructured Information*. Springer, New York, NY, 2004.

5

A Case Study in Natural Language Based Web Search

Giovanni Marchisio, Navdeep Dhillon, Jisheng Liang, Carsten Tusk, Krzysztof Koperski, Thien Nguyen, Dan White, and Lubos Pochman

5.1 Introduction

Is there a public for natural language based search? This study, based on our experience with a Web portal, attempts to address criticisms on the lack of scalability and usability of natural language approaches to search. Our solution is based on InFact®, a natural language search engine that combines the speed of keyword search with the power of natural language processing. InFact performs clause level indexing, and offers a full spectrum of functionality that ranges from Boolean keyword operators to linguistic pattern matching in real time, which include recognition of syntactic roles, such as subject/object and semantic categories, such as people and places. A user of our search can navigate and retrieve information based on an understanding of actions, roles and relationships. In developing InFact, we ported the functionality of a deep text analysis platform to a modern search engine architecture. Our distributed indexing and search services are designed to scale to large document collections and large numbers of users. We tested the operational viability of InFact as a search platform by powering a live search on the Web. Site statistics and user logs demonstrate that a statistically significant segment of the user population is relying on natural language search functionality. Going forward, we will focus on promoting this functionality to an even greater percentage of users through a series of creative interfaces.

Information retrieval on the Web today makes little use of Natural Language Processing (NLP) techniques [1, 3, 11, 15, 18]. The perceived value of improved understanding is greatly outweighed by the practical difficulty of storing complex linguistic annotations in a scalable indexing and search framework. In addition, any champion of natural language techniques must overcome significant hurdles in user interface design, as greater search power often comes at a price of more work in formulating a query and navigating the results. All of these obstacles are compounded by the expected resistance to any technological innovation that has the potential to change or erode established models for advertising and search optimization, which are based on pricing of individual keywords or noun phrases, rather than relationships or more complex linguistic constructs.

Nevertheless, with the increasing amount of high value content made available on the Web and increased user sophistication, we have reasons to believe that a segment

of the user population will eventually welcome tools that understand a lot more than present day keyword search does. Better understanding and increased search power depend on better parameterization of text content in a search engine index. The most universal storage employed today to capture text content is an inverted index. In a typical Web search engine, an inverted index may register presence or frequency or keywords, along with font size or style, and relative location in a Web page. Obviously this model is only a rough approximation to the complexity of human language and has the potential to be superseded by future generation of indexing standards.

InFact relies on a new approach to text parameterization that captures many linguistic attributes ignored by standard inverted indices. Examples are syntactic categories (parts of speech), syntactical roles (such as subject, objects, verbs, prepositional constraints, modifiers, etc.) and semantic categories (such as people, places, monetary amounts, etc.). Correspondingly, at query time, there are explicit or implicit search operators that can match, join or filter results based on this rich assortment of tags to satisfy very precise search requirements.

The goal of our experiment was to demonstrate that, once scalability barriers are overcome, a statistically significant percentage of Web users can be converted from keyword search to natural language based search. InFact has been the search behind the GlobalSecurity.org site (www.globalsecurity.org) for the past six months. According to the Alexa site (www.alexa.com), GlobalSecurity.org has a respectable overall traffic rank (no. 6,751 as of Feb 14, 2006). Users of the site can perform keyword searches, navigate results by action themes, or enter explicit semantic queries. An analysis of query logs demonstrate that all these non-standard information discovery processes based on NLP have become increasingly popular over the first six months of operation.

The remainder of this chapter is organized as follows. Section 5.2 presents an overview of our system, with special emphasis on the linguistic analyses and new search logic. Section 5.3 describes the architecture and deployment of a typical InFact system. Section 5.4 is a study of user patterns and site statistics.

5.2 InFact System Overview

InFact consists of an indexing and a search module. With reference to Figure 5.1, indexing pertains to the processing flow on the bottom of the diagram. InFact models text as a complex multivariate object using a unique combination of deep parsing, linguistic normalization and efficient storage. The storage schema addresses the fundamental difficulty of reducing information contained in parse trees into generalized data structures that can be queried dynamically. In addition, InFact handles the problem of linguistic variation by mapping complex linguistic structures into semantic and syntactic equivalents. This representation supports dynamic relationship and event search, information extraction and pattern matching from large document collections in real time.

5.2.1 Indexing

With reference to Figure 5.1, InFact's Indexing Service performs in order: 1) document processing, 2) clause processing, and 3) linguistic normalization.

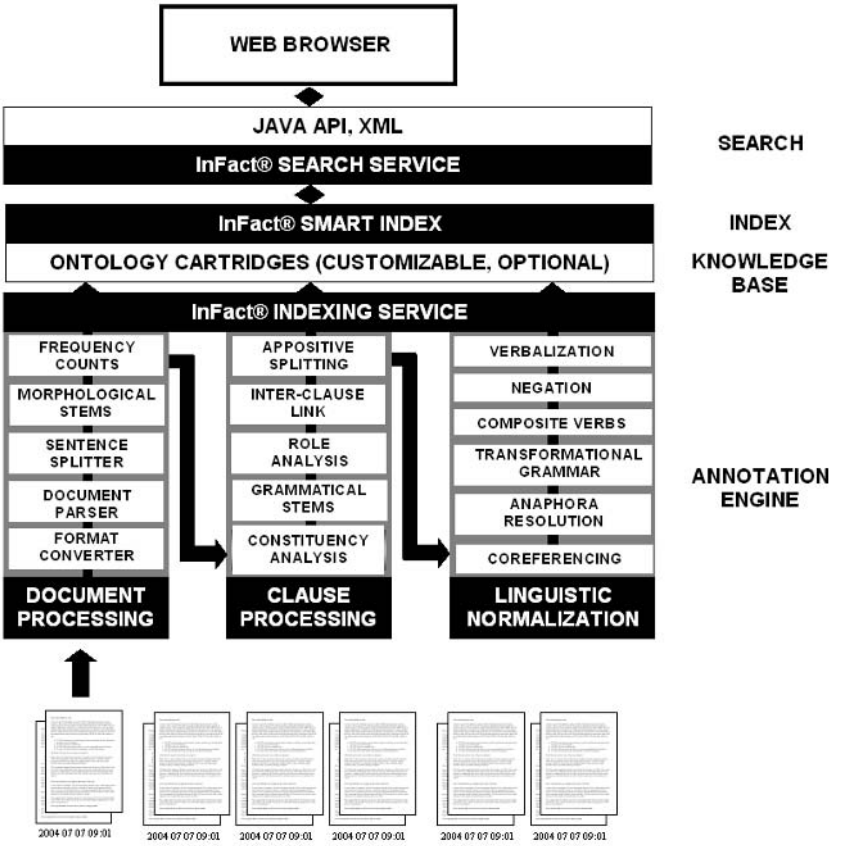


Fig. 5.1. Functional overview of InFact.

Document Processing

The first step in document processing is format conversion, which we handle through our native format converters, or optionally via search export conversion software from Stellant™ (www.stellent.com), which can convert 370 different input file types. Our customized document parsers can process disparate styles and recognized zones within each document. Customized document parsers address the issue that a Web page may not be the basic unit of content, but it may consist of separate sections with an associated set of relationships and metadata. For instance a blog post may contain blocks of text with different dates and topics. The challenge is to automatically recognize variations from a common style template, and segment information in the index to match zones in the source documents, so the relevant section can be displayed in response to a query. Next we apply logic for sentence splitting in preparation for clause processing. Challenges here include the ability to unambiguously recognize sentence delimiters, and recognize regions such as lists or tables that

are unsuitable for deep parsing. Last, we extract morphological stems and compute frequency counts, which are then entered in the index.

Clause Processing

The indexing service takes the output of the sentence splitter and feeds it to a deep linguistic parser. A sentence may consist of multiple clauses. Unlike traditional models that store only term frequency distributions, InFact performs clause level indexing and captures syntactic category and roles for each term, and grammatical constructs, relationships, and inter-clause links that enable it to understand events. One strong differentiator of our approach to information extraction [4, 5, 7, 8, 14, 19] is that we create these indices automatically, without using predefined extraction rules, and we capture all information, not just predefined patterns. Our parser performs a full constituency and dependency analysis, extracting part-of-speech (POS) tags and grammatical roles for all tokens in every clause. In the process, tokens undergo grammatical stemming and an optional, additional level of tagging. For instance, when performing grammatical stemming on verb forms, we normalize to the infinitive, but we may retain temporal tags (e.g., past, present, future), aspect tags (e.g., progressive, perfect), mood/modality tags (e.g., possibility, subjunctive, irrealis, negated, conditional, causal) for later use in search.

Next we capture inter-clause links, through: 1) explicit tagging of conjunctions or pronouns that provide the link between the syntactic structures for two adjacent clauses in the same sentence; and 2) pointing to the list of annotated keywords in the antecedent and following sentence. Note that the second mechanism ensures good recall in those instances where the parser fails to produce a full parse tree for long and convoluted sentences, or information about an event is spread across adjacent sentences. In addition, appositive clauses are recognized, split into separate clauses and cross-referenced to the parent clause.

For instance, the sentence: "Appointed commander of the Continental Army in 1775, George Washington molded a fighting force that eventually won independence from Great Britain" consists of three clauses, each containing a governing verb (appoint, mold, and win). InFact decomposes it into a primary clause ("George Washington molded a fighting force") and two secondary clauses, which are related to the primary clause by an appositive construct ("Appointed commander of the Continental Army in 1775") and a pronoun ("that eventually won independence from Great Britain"), respectively. Each term in each clause is assigned a syntactic category or POS tag (e.g., noun, adjective, etc.) and a grammatical role tag (e.g., subject, object, etc.). InFact then utilizes these linguistic tags to extract relationships that are normalized and stored in an index, as outlined in the next two sections.

Linguistic Normalization

We apply normalization rules at the syntactic, semantic, or even pragmatic level. Our approach to coreferencing and anaphora resolution make use of syntactic agreement and/or binding theory constraints, as well as modeling of referential distance, syntactic position, and head noun [6, 10, 12, 13, 16, 17]. Binding theory places syntactic restrictions on the possible coreference relationships between pronouns and

their antecedents [2]. For instance, when performing pronoun coreferencing, syntactic agreement based on person, gender and number limits our search for a noun phrase linked to a pronoun to a few candidates in the text. In addition, consistency restrictions limit our search to a precise text span (the previous sentence, the preceding text in the current sentence, or the previous and current sentence) depending upon whether the pronoun is personal, possessive, reflective, and what is its person. In the sentence "John works by himself," "himself" must refer to John, whereas in "John bought him a new car," "him" must refer to some other individual mentioned in a previous sentence. In the sentence, ""You have not been sending money," John said in a recent call to his wife from Germany," binding theory constraints limit pronoun resolution to first and second persons within a quotation (e.g., you), and the candidate antecedent to a noun outside the quotation, which fits the grammatical role of object of a verb or argument of a preposition (e.g., wife). Our coreferencing and anaphora resolution models also benefit from preferential weighting based on dependency attributes. The candidate antecedents that appear closer to a pronoun in the text are scored higher (weighting by referential distance). Subject is favored over object, except for accusative pronouns (weighting by syntactic position). A head noun is favored over its modifiers (weighting by head label). In addition, as part of the normalization process, we apply a transformational grammar to map multiple surface structures into an equivalent deep structure. A common example is the normalization of a dependency structure involving a passive verb form into the active, and recognition of the deep subject of such clause. At the more pragmatic level, we apply rules to normalize composite verb expressions, capture explicit and implicit negations, or to verbalize noun or adjectives in cases where they convey action sense in preference to the governing verb of a clause. For instance, the sentences "Bill did not visit Jane," which contains an explicit negation, and "Bill failed to visit Jane," where the negation is rendered by a composite verb expression, are mapped to the same structure.

5.2.2 Storage

The output of a deep parser is a complex augmented tree structure that usually does not lend itself to a tractable indexing schema for cross-document search. Therefore, we have developed a set of rules for converting an augmented tree representation into a scalable data storage structure.

In a dependency tree, every word in the sentence is a modifier of exactly one other word (called its head), except the head word of the sentence, which does not have a head. We use a list of tuples to specify a dependency tree with the following format:

(Label Modifier Root POS Head-label Role Antecedent [Attributes])

where: **Label** is a unique numeric ID; **Modifier** is a term in the sentence; **Root** is the root form (or category) of the modifier; **POS** is its lexical category; **Head-label** is the ID of the term that modifier modifies; **Role** specifies the type of dependency relationship between head and modifier, such as subject, complement, etc; **Antecedent** is the antecedent of the modifier; **Attributes** is the list of semantic attributes that may be associated with the modifier, e.g., person's name, location, time, number, date, etc.

For instance, the parse tree for our Washington example above is shown in Table 5.1.

Table 5.1. The parse tree representation of a sentence.

Label	Modifier	Root	POS	Head Label	Role	Antecedent	Attributes
1	Appointed	Appoint	V				
2	commander		N	1	Obj		Person/title
3	of		Prep	2	Mod		
4	the		Det	5	Det		
5	Continental Army		N	3	Pcomp		Organization/name
6	in		Prep	1	Mod		
7	1775		N	6	Pcomp		Numeric/date
8	George Washington		N	9	Subj		Person/name
9	molded	mold	V				
10	a		Det	12	Det		
11	fighting		A	12	Mod		
12	force		N	9	Obj		
13	that		N	15	Subj	12	
14	eventually		A	15	Mod		
15	won	win	V				
16	independence		N	15	Obj		
17	from		Prep	16	Mod		
18	Great Britain		N	17	Pcomp		Location/country

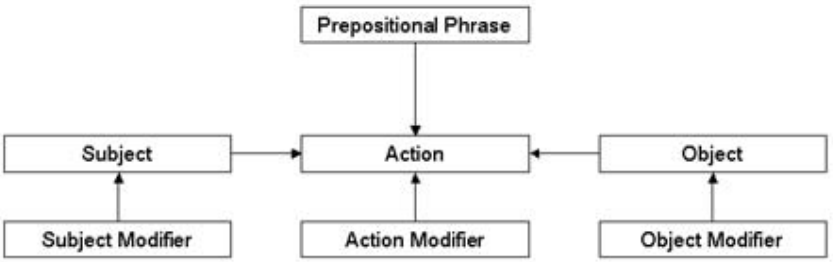


Fig. 5.2. The Subject-Action-Object indexing structure.

The basic idea behind or approach to indexing involves collapsing selected nodes in the parse tree to reduce the overall complexity of the dependency structures.

We model our storage structures after the general notion of subject-action-object triplets, as shown in Figure 5.2. Interlinked subject-action-object triples and their respective modifiers can express most types of syntactic relations between various entities within a sentence.

The index abstraction is presented in Table 5.2, where the additional column *"Dist"* denotes degrees of separations (or distance) between primary *Subject, Verb, Object* and each *Modifier*, and *"Neg"* keeps track of negated actions.

Table 5.2. The index abstraction of a sentence.

Subject	Subject-Modifier	Object	Object-Modifier	Verb	Verb-Modifier	Prep	Pcomp	Dist	Neg
		Washington	George	appoint				1	F
		commander		appoint				1	F
		Army	Continental	appoint				3	F
				appoint		in	1775	2	F
Washington	George	force	fighting	mold				2	F
force	fighting	independence		win				1	F
				win		from	Great Britain	3	F
				win	eventually			1	F

InFact stores the normalized triplets into dedicated index structures that

- are optimized for efficient keyword search
- are optimized for efficient cross-document retrieval of arbitrary classes of relationships or events (see examples in the next section)
- store document metadata and additional ancillary linguistic variables for filtering of search results by metadata constraints (e.g., author, date range), or by linguistic attributes (e.g., retrieve negated actions, search subject modifier field in addition to primary subject in a relationship search)
- (optionally) superimposes annotations and taxonomical dependencies from a custom ontology or knowledge base.

With regard to the last feature, for instance, we may superimpose a *[Country]* entity label on a noun phrase, which is the subject of the verb *"to attack."* The index supports multiple ontologies and entangled multiparent taxonomies.

InFact stores "soft events" instead of fitting textual information into a rigid relational schema that may result in information loss. "Soft events" are data structures that can be recombined to form events and relationships. "Soft events" are pre-indexed to facilitate thematic retrieval by action, subject, and object type. For instance, a sentence like "The president of France visited the capital of Tunisia" contains evidence of 1) a presidential visit to a country's capital and 2) diplomatic relationships between two countries. Our storage strategy maintains both interpretations. In other words, we allow more than one subject or object to be associated with the governing verb of a sentence. The tuples stored in the database are therefore "soft events," as they may encode alternative patterns and relationships found in each sentence. Typically, only one pattern is chosen at search time, in response to a specific user request (i.e., request #1: gather all instances of a president visiting a country; request #2: gather all instances of interactions between any two countries).

5.2.3 Search

Unlike keyword search engines, InFact employs a highly expressive query language (IQL or InFact Query Language) that combines the power of grammatical roles with the flexibility of Boolean operators, and allows users to search for actions, entities, relationships, and events. InFact represents the basic relationship between two entities with an expression of the kind:

Subject Entity > Action > Object Entity,

The arrows in the query refer to the directionality of the action, which could be either uni-directional (as above) or bi-directional. For example,

Entity 1 <> Action <> Entity 2

will retrieve all relationships involving *Entity 1* and *Entity 2*, regardless of their roles as subject or object of the action. Wildcards can be used for any grammatical role. For instance, the query "$*$ > *eat* > *cake*" will retrieve a list of anybody or anything that eats a cake; and a query like "*John* > $*$ > *Jane*" will retrieve a list of all uni-directional relationships between John and Jane. InFact also supports the notion of entity types. For instance, in addition to entering an explicit country name like "Argentina" as Entity 1 or Entity 2 in a relationship query, a user can enter a wildcard for any country name by using the syntax *[Country]*. InFact comes with a generic ontology that includes *[Location]*, *[Person]*, *[Organization]*, *[Numeric]* as the four main branches. Entity types can be organized hierarchically in a taxonomy. IQL renders hierarchical dependencies by means of taxonomy paths. For instance, in *[Entity/Location/Country]* and *[Entity/Location/City]* both *[Country]* and *[City]* nodes have a common parent *[Location]*. Taxonomy path can encode "is-a" relations (as in the above examples), or any other relations defined in a particular ontology (e.g., "part-of" relation). When querying, we can use a taxonomy node in a relationship search, e.g., *[Location]*, and the query will automatically include all subpaths in the taxonomic hierarchy, including *[City]*, *[Location]*, or narrow the search by expanding the path to *[Location/City]*.

With the InFact query language, we can search for:

- Any relationships involving an entity of interest

For example, the query "*George Bush* <> * <> *" will retrieve any events involving "*George Bush*" as subject or object

- Relationships between two entities or entity types

For example, the query "*China* <> * <> *Afghan**" will retrieve all relationships between the two countries. Note in this case a wildcard is used in "*Afghan**" to handle different spelling variations of Afghanistan. The query "*Bin Laden* <>*<> *[Organization]*" will retrieve any relationships involving "Bin Laden" and an organization.

- Events involving one or more entities or types

For example, the query "*Pope* > *visit* > *[country]*" will return all instances of the Pope visiting a country. In another example, "*[Organization/name]* > *acquire* > *[Organization/name]*" will return all events involving a named company buying another named company.

- Events involving a certain action type

"Action types" are groups of semantically linked actions. For example, query "*[Person] > [Communication] > [Person]*" will retrieve all events involving communication between two people.

InFact's query syntax supports Boolean operators (i.e., AND, OR, NOT). For example, the query:

Clinton NOT Hillary > visit OR travel to > [Location]

is likely to retrieve the travels of *Bill Clinton*, but not *Hillary Clinton*.

We can further constrain actions with modifiers, which can be explicit entities or entity types, e.g., *Paris* or *[location]*. For example, the query

[Organization/Name] > buy > [Organization/Name]ˆ[money]

will only return results where a document mentions a specific monetary amount along with a corporate acquisition. Similarly, the query

Bush <> meet<> Clinton ˆ[location]

will return results restricted to actions that occur in an explicit geographical location.

We can also filter search results by specifying document-level constraints, including:

- Document metadata tags – lists of returned actions, relationships or events are restricted to documents that contain the specified metadata values.
- Boolean keyword expressions – lists of returned actions, relationships or events are restricted to documents that contain the specified Boolean keyword expressions.

For instance, a query like:

[Organization/Name] > buy > [Organization/Name]ˆ[money]; energy NOT oil

will return documents that mentions a corporate acquisition with a specific monetary amount, and also contain the keyword "energy" but do not contain the keyword "oil."

InFact also provides a context operator for inter-clause linking. Suppose for instance, that we want to retrieve all events where a plane crash kills a certain number of passengers. The event could be spread over adjacent sentences, as in: *"The plane crashed shortly after take-off. As many as 224 people were killed."*

In this case, a query like:

** > kill > [numeric] ∼plane crash*

will retrieve all plane crash events, regardless of whether they are contained in a single or multiple, adjacent sentences.

InFact can also support synonyms and query expansion via custom ontologies. In this case, InFact will automatically recognize the equivalence of entities or actions that belong to the same ontology node.

The InFact Query Language rests on a flexible Java Search API. The Java Search API allows us to programmatically concatenate search operators, package and present them to the end user through a simpler interface.

5.3 Architecture and Deployment

We designed both indexing and search as parallel distributed services. Figure 5.3 shows a typical deployment scenario, with an indexing service on the left and a search service on the right. A typical node in each of the diagrams would is a dual processor (e.g., 2.8+GHz Xeon 1U) machine with 4GB of RAM and two 120GB drives.

The Indexing Service (left) processes documents in parallel. Index workers access source documents from external web servers. Multiple index workers can run on each node. Each index worker performs all the "Annotation Engine" analyses described in Figure 5.1. An index manager orchestrates the indexing process across many index workers. The results of all analyses are stored in temporary indices in the index workers. At configurable intervals, the index manager orchestrates the merging of all temporary indices into the partition index components.

A partition index hosts the actual disk based indices used for searching. The contents of a document corpus are broken up into one or more subsets that are each stored in a partition index. The system supports multiple partition indices: the exact number will depend on corpus size, number of queries per second and desired response time. Indices are queried in parallel and are heavily IO bound. Partition indices are attached to the leaf nodes of the Search Service on the right.

In addition to storing results in a temporary index, index workers can also store the raw results of parsing in a Database Management System (DBMS). The database is used almost exclusively to restore a partition index in the event of index corruption. Data storage requirements on the DBMS range between 0.5 and 6x corpus size depending on which recovery options for the InFact system are enabled. Once a document has been indexed and merged into a partition index it is available for searching.

In a typical search deployment, queries are sent from a client application; the client application may be a Web browser or a custom application built using the Search API. Requests arrive over HTTP and are passed through a Web Server to the Search Service layer and on to the top searcher of a searcher tree. Searchers are responsible for searching one or more partition index. Multiple searchers are supported and can be stacked in a hierarchical tree configuration to enable searching large data sets. The top level searcher routes ontology related requests to one or more ontology searchers, which can run on a single node. Search requests are passed to child searchers, which then pass the request down to one or more partition indices. The partition index performs the actual search against the index, and the result passes up the tree until it arrives back at the client for display to the user.

If a particular segment of data located in a partition index is very popular and becomes a search bottleneck, it may be cloned; the parent searcher will load balance across two or three partition indices. In addition, if ontology searches become a bottleneck, more ontology searchers may be added. If a searcher becomes a bottleneck, more searchers can be added. The search service and Web server tier may be replicated, as well, if a load balancer is used.

The example in Figure 5.3 is an example of a large-scale deployment. In the GlobalSecurity.org portal, we currently need only four nodes to support a user community of 100,000 against a corpus of several GB of international news articles, which are updated on a daily basis.

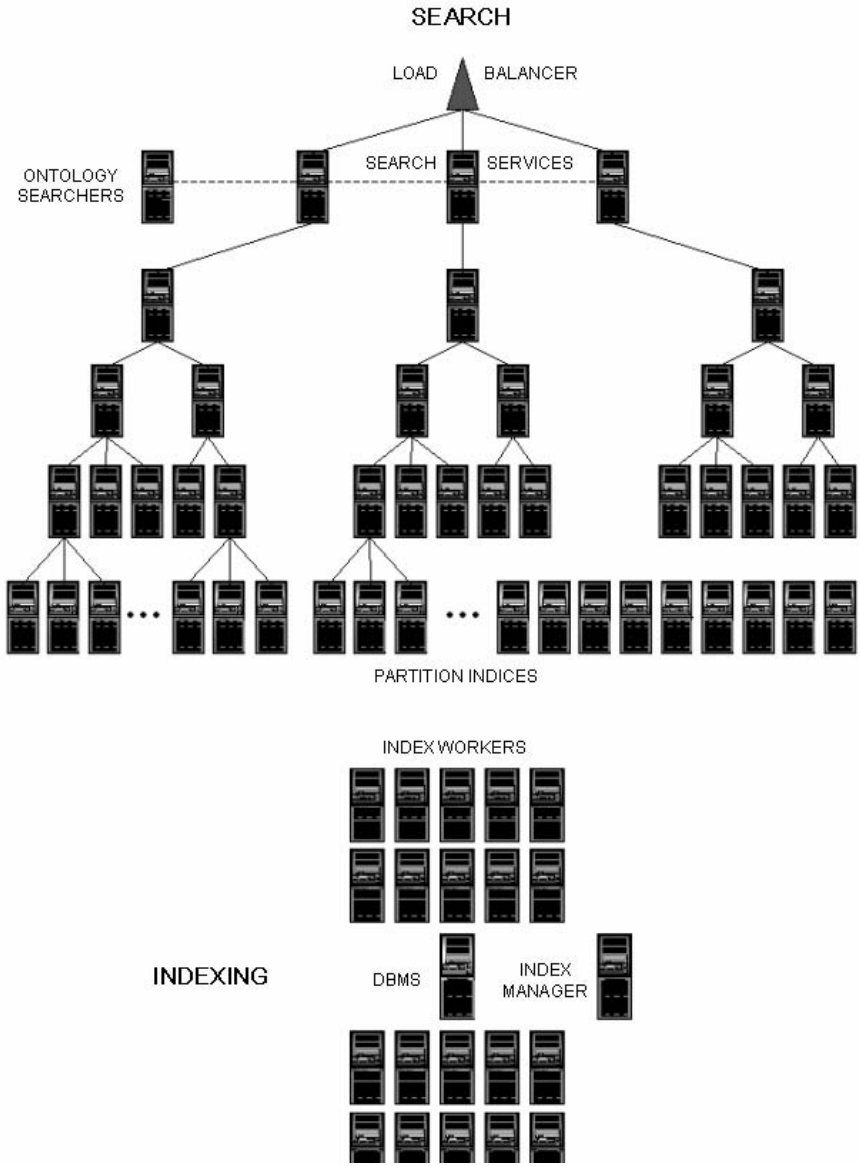


Fig. 5.3. Architectural overview of InFact.

5.4 The GlobalSecurity.org Experience

5.4.1 Site Background

InFact started powering the GlobalSecurity.org Web site on June 22, 2005. Based in Alexandria, VA, and "launched in 2000, GlobalSecurity.org is the most compre-

hensive and authoritative online destination for those in need of both reliable background information and breaking news ... GlobalSecurity.org's unique positioning enables it to reach both a targeted and large diversified audience. The content of the website is updated hourly, as events around the world develop, providing in-depth coverage of complicated issues. The breadth and depth of information on the site ensures a loyal repeat audience. This is supplemented by GlobalSecurity.org's unique visibility in the mass media, which drives additional growth" [9]. The director of GlobalSecurity.org, John Pike, regularly provides commentary and analysis on space and security issues to PBS, CNN, MSNBC, Fox, ABC, CBS, NBC, BBC, NPR, and numerous print and online publications. In powering this site, InFact serves the information search needs of a well-established user community of 100,000, consisting of news reporters, concerned citizens, subject matter experts, senior leaders, and junior staff and interns.

5.4.2 Operational Considerations

When preparing the GlobalSecurity.org deployment, one of our prime concerns was the response time of the system. For this reason, we kept the data size of the partition indices small enough so that most operations occur in memory and disk access is minimal. We split the GlobalSecurity.org data across two index chunks, each containing roughly 14 GB of data in each partition index. Another concern was having sufficient capacity to handle the user load. To account for future user traffic, we specified the deployment for 2-3 times the maximum expected load of about 11,000 queries per day. This left us with two cloned partition indices per index chunk. In addition, we wanted a hot back up of the entire site, in case of any hardware failures, and to support us each time we are rolling out new features.

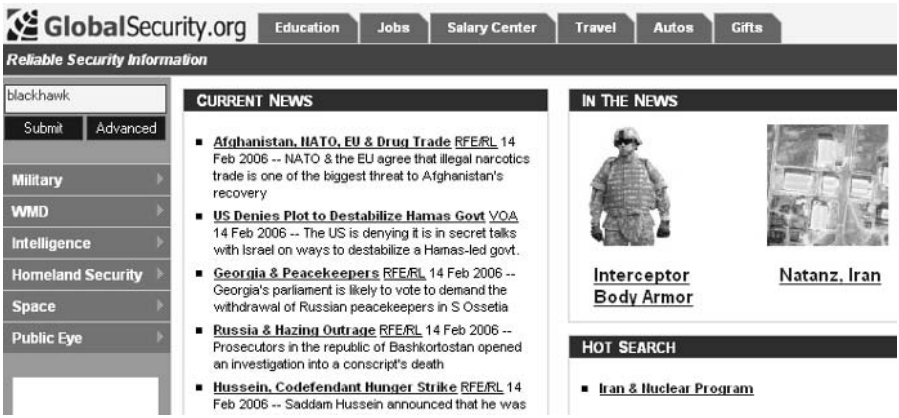


Fig. 5.4. The GlobalSecurity.org home page.

Another area of concern was the distribution of query types. Our system has significantly varying average response time and throughput (measured in queries/minute) depending on the type of queries being executed. We assumed that users

would take some time to migrate from keyword queries to fact queries. Therefore, we selected a very conservative ratio of 50/50 fact-to-keyword query types with a view to adding more hardware if needed. After automatically generating millions of query files, we heavily loaded the system with the queries to simulate heavy traffic using JMeter, a multi-threaded client web user simulation application from the Apache Jakarta organization. Based on these simulations, we deployed with only four nodes.

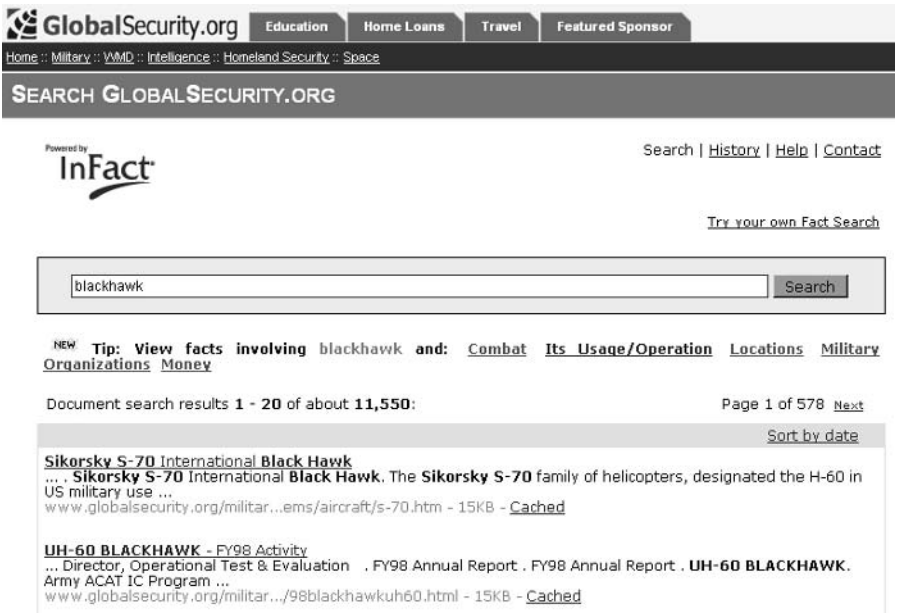


Fig. 5.5. Keyword search result and automatic tip generation with InFact in response to the keyword query "blackhawk."

5.4.3 Usability Considerations

In deploying InFact on the GlobalSecurity.org site, our goal was to serve the information needs of a wide community of users, the majority of which are accustomed to straightforward keyword search. Therefore, on this site, by default, InFact acts as a keyword search engine. However, we also started experimenting with ways to progressively migrate users away from keyword search and towards natural language search or "fact search." With reference to Figure 5.4, users approaching the site can enter InFact queries from the search box in the upper left, or click on the Hot Search link. The latter executes a predefined fact search, which is particularly popular over an extended time period (days or even weeks). The Hot Search is controlled by GlobalSecurity.org staff, and is outside the control of the general user. However, once in the InFact search page (Figure 5.5), the user can execute fact searches explicitly by using the IQL syntax. The IQL syntax is fully documented in the InFact Help page.

Alternatively, by clicking on the "Try your own Fact Search" link on the upper right of the InFact Search page, the user is introduced to a Custom Query Generator (Figure 5.6), which produces the query of Figure 5.7.

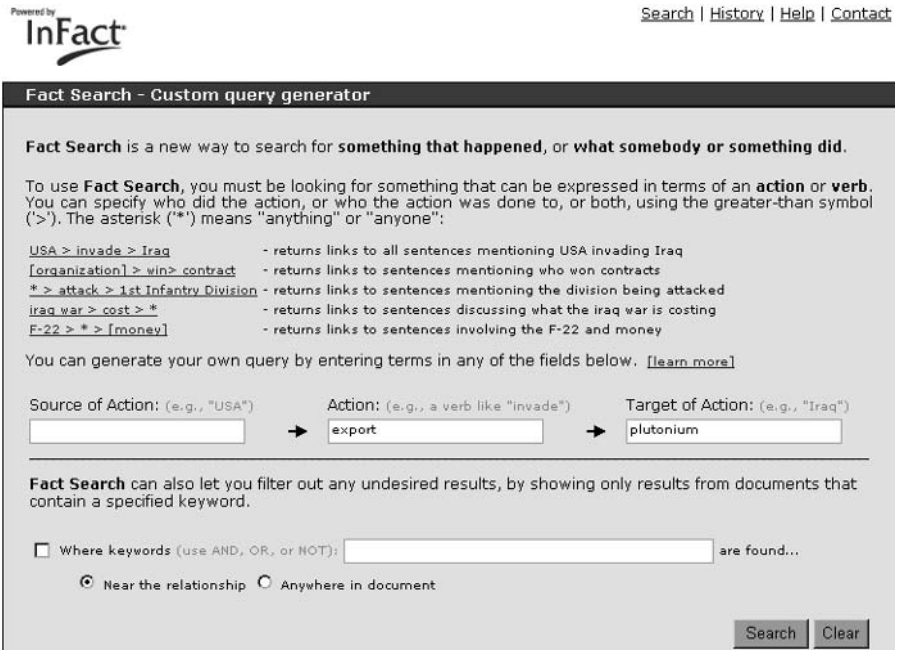


Fig. 5.6. Fact search with the InFact Custom Query Generator: the user is looking for facts that involve the export of plutonium.

The most interesting device we employed is guided fact navigation in response to a keyword entry. We call this process "tip generation." In this scenario, we capture keywords entered by a user and try to understand whether these are names of people, places, organization, military units, vehicles, etc. When executing a keyword search, the InFact system can recommend several fact searches which may be of interest to a user based on the keywords entered. These recommendations are presented to the user as a guided navigation menu consisting of links. In the example of Figure 5.5, the user is performing a keyword search for "blackhawk." The user sees a series of links presented at the top of the result set. They read: "Tip: View facts involving blackhawk and: Combat, Its Usage/Operation, Locations, Military Organizations, Money." Each of these links when clicked in turn executes a fact search. For instance, clicking on Military Organizations will generate the list of facts or relationships of Figure 5.8, which gives an overview of all military units that have used the blackhawk helicopter; clicking on Money will generate the list of facts or relationships of Figure 5.9, which gives an overview of procurement and maintenance costs, as well as government spending for this vehicle. The relationships are returned in a table display where each row is an event, and columns identify the three basic semantic

Fig. 5.7. Fact search with the InFact Custom Query Generator: InFact translates the query of Figure 5.6 into the InFact Query Language (IQL) and returns a list of results. IQL operators are fully documented in the Help page.

roles of source (or subject), action (or verb), and target (or object). Note that relationships, by default, are sorted by relevance to a query, but can also be resorted by date, action frequency, or alphabetically by source, action or target. Each of the relationships or facts in the table is in turn hyperlinked to the exact location in the source document where it was found, so the user can quickly validate the findings and explore its context (Figure 5.10).

Usage logs were the primary driver for this customization effort. The personnel at GlobalSecurity.org were very helpful and provided us with many months of user traffic Web logs. We wrote some simple scripts to analyze the logs. For example, we studied the 500 most popular key word searches performed on the site ranked in order of popularity. Next, we began looking for entity types that would be helpful to the most number of users. We found a lot of user interest in weapons, terrorists, and US officials, amongst other things. We then set about creating ontologies for each of these areas. New custom ontologies can easily be mapped into the internal InFact ontology XML format.

5.4.4 Analysis of Query Logs

We wish to quantify the relative popularity of natural language (Fact) search versus keyword search. In addition, we wish to compare the relative success of alternative strategies we adopted to overcome usability issues. This study of log data reflect

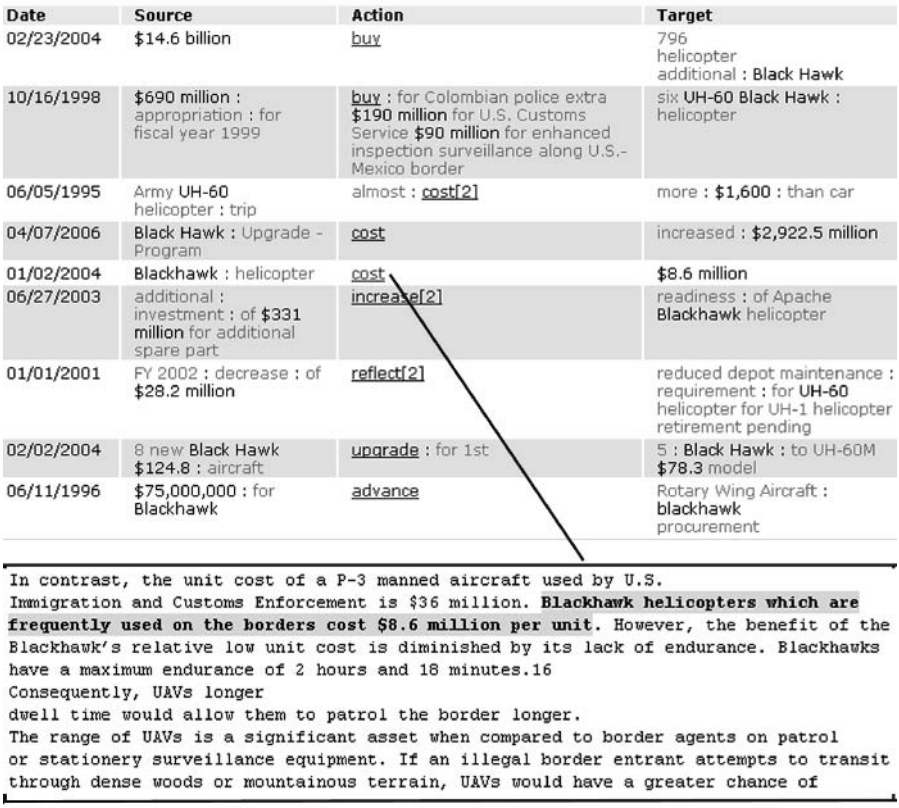
Chinook All : BLACK HAWK	include	160th Special Operations Aviation Regiment 101st Airborne Division 10th Mountain Division 82nd Airborne Division
Marine Corps	inspect : across airfield	rigging : on Blackhawk
U.S. Army	install	UH-60 : in future ALQ-211 : on CH-47
U.S. Army	introduce : for example	fly-by-wire : capability : to Army Apache Black Hawk fleet
air force	investigate	Black Hawk fratricide : incident
Troops : from 2nd Battalion 505th Parachute Infantry Regiment 82nd Airborne Division	kick off	Operation Desert Lion : with air assault from Chinook Black Hawk helicopter
Black Hawk : helicopter	land : As part of Operation Falcon Sweep	Shakaria Soldier : of 2nd Battalion 502nd Infantry Regiment
40 : CH-47 UH-60 AH-1	lift	1st Brigade : into
211th Aviation Group	maintain	UH-60 Blackhawk AH-64 Apache
air force : contractor	maintain	Blackhawk : helicopter Army : Apache

Fig. 5.8. Tip Navigation with InFact: facts involving the "blackhawk" helicopter and military organizations.

Date	Source	Action	Target
02/23/2004	$14.6 billion	buy	796 helicopter additional : Black Hawk
10/16/1998	$690 million : appropriation : for fiscal year 1999	buy : for Colombian police extra $190 million for U.S. Customs Service $90 million for enhanced inspection surveillance along U.S.-Mexico border	six UH-60 Black Hawk : helicopter
06/05/1995	Army UH-60 helicopter : trip	almost : cost[2]	more : $1,600 : than car
04/07/2006	Black Hawk : Upgrade - Program	cost	increased : $2,922.5 million
01/02/2004	Blackhawk : helicopter	cost	$8.6 million
06/27/2003	additional : investment : of $331 million for additional spare part	increase[2]	readiness : of Apache Blackhawk helicopter
01/01/2001	FY 2002 : decrease : of $28.2 million	reflect[2]	reduced depot maintenance : requirement : for UH-60 helicopter for UH-1 helicopter retirement pending
02/02/2004	8 new Black Hawk $124.8 : aircraft	upgrade : for 1st	5 : Black Hawk : to UH-60M $78.3 model
06/11/1996	$75,000,000 : for Blackhawk	advance	Rotary Wing Aircraft : blackhawk procurement

Fig. 5.9. Tip Navigation with InFact: facts involving the "blackhawk" helicopter and money.

four ways users submit a natural language query to InFact: 1) they click on the Hot Search link; 2) they click on a keyword tip; 3) they click on an example in the Query Generator or Help page; 4) they attempt to type an explicit relationship or fact search using the IQL syntax.



Date	Source	Action	Target
02/23/2004	$14.6 billion	buy	796 helicopter additional : Black Hawk
10/16/1998	$690 million : appropriation : for fiscal year 1999	buy : for Colombian police extra $190 million for U.S. Customs Service $90 million for enhanced inspection surveillance along U.S.-Mexico border	six UH-60 Black Hawk : helicopter
06/05/1995	Army UH-60 helicopter : trip	almost : cost[2]	more : $1,600 : than car
04/07/2006	Black Hawk : Upgrade - Program	cost	increased : $2,922.5 million
01/02/2004	Blackhawk : helicopter	cost	$8.6 million
06/27/2003	additional : investment : of $331 million for additional spare part	increase[2]	readiness : of Apache Blackhawk helicopter
01/01/2001	FY 2002 : decrease : of $28.2 million	reflect[2]	reduced depot maintenance : requirement : for UH-60 helicopter for UH-1 helicopter retirement pending
02/02/2004	8 new Black Hawk $124.8 : aircraft	upgrade : for 1st	5 : Black Hawk : to UH-60M $78.3 model
06/11/1996	$75,000,000 : for Blackhawk	advance	Rotary Wing Aircraft : blackhawk procurement

In contrast, the unit cost of a P-3 manned aircraft used by U.S. Immigration and Customs Enforcement is $36 million. **Blackhawk helicopters which are frequently used on the borders cost $8.6 million per unit.** However, the benefit of the Blackhawk's relative low unit cost is diminished by its lack of endurance. Blackhawks have a maximum endurance of 2 hours and 18 minutes.16
Consequently, UAVs longer
dwell time would allow them to patrol the border longer.
The range of UAVs is a significant asset when compared to border agents on patrol or stationery surveillance equipment. If an illegal border entrant attempts to transit through dense woods or mountainous terrain, UAVs would have a greater chance of

Fig. 5.10. Tip Navigation with InFact: each fact is hyperlinked to the exact location where it was found in the source document.

At the time of this writing, an average of 36% of advanced search users click on the hot search link "Iran and Nuclear program," which executes a predefined search like *"Iran* $> * \sim$ *nuclear."* However, it is difficult to assess what the user experience is like because in 80% of cases the user performs non-search-related tasks, and therefore we don't know how long they spent looking at the results. Note that users clicking on this link may not realize that they are going to a search engine page, since the link title is ambiguous. The results of this search are quite good, and still relevant. The hot search is an important entry point into our search site, as 36% of all the fact search queries executed came from this source. It seems likely that adding more of these hot search links or otherwise accentuating them on the page would significantly increase user exposure to natural language based queries.

Our analysis of query logs shows that keyword tips are the most effective way to migrate users to Fact Search. Users who click on tips frequently follow up with

queries of their own. Tip clickers also write better queries, probably because, after seeing the column display, they have a much better sense of how queries can be composed. Keyword tip clickers typically find the results engaging enough to spend an average of 1.5 minutes studying the results: 37% of users go back and click on more than one tip. Even better, 87% follow up by clicking on the "Try your own Fact Search" link and try their own query. All of the queries attempted are queries; 90% produce results; our follow up analysis suggests that for two thirds of these queries the results are relevant to the users search goals. In other words, users who click on the tips are extremely likely not only to try their own fact search, but also to pay enough attention to the format to write both valid and useful queries.

Examples in the Help File or Query Generator are largely ineffective at getting users to try Fact Search. Because the results returned by the examples usually do not necessarily relate to what the user wishes to search on, the column display is more of a distraction than an enticement to try Fact Search. However, those who go on to try Fact Search, after clicking on an example, have a better chance of writing good queries. Example link clickers are less likely to experiment with Fact Search or invest time learning how it works. Seventy-two percent of users end their session after clicking on one or more examples, not even returning to perform the keyword search that presumably brought them to the site in the first place. Of the 28% who did not leave the site after clicking an example, two thirds went on to try a Fact Search. Only 6% of users click on examples after having tried a Fact Search query on their own. Analysis of this user group suggests that examples have a place in the UI, but are not sufficiently compelling to motivate users to try Fact Search alone. However, this evidence does lend support to the hypothesis that users who see the column display are more likely to create valid queries: 60% of the users who click on examples and go on to write their own queries write valid queries and get results, which is still a much higher percentage than for users who blindly try to create queries.

About 75% of users who try Fact Search directly by using the IQL syntax, and without seeing the column display first fail to get results. Forty-five percent of users write invalid queries where nouns are inserted in the action field (the most common error). Another common error is specifying too much information or attaching prepositions to noun phrases. We can detect some of these errors automatically, and we plan to provide automatic guidance to users going forward. About 20% of query creators get impressive results. Most successful users get their queries right on the first shot, and, in general, seem unwilling to invest much time experimenting. Successful users are most likely expert analysts. In reproducing their searches and inspecting their results, we estimate that they have a positive impression of Fact search. In 75% of cases the results of Fact Search take direct the user quickly to the relevant parts of relevant documents, providing a deeper overview and faster navigation of content. However, in 25% of cases, expert users also write queries that return no results. Reasons for this include specifying too much information or including modifiers or prepositional terms in the verb field such as: "cyber attack," "led by," and "go to." In many cases users would be successful by just entering the verb. In some cases, users get lots of fact search results, but lack the experience to refine their query, so they simply go back to keyword search. We should try to communicate how queries can be modified further if there are too many results, perhaps by adding an ontology tag, or a context operator to the query syntax. For instance, the query *"Bush > meet > [person]"* could yield a large number of irrelevant results, if

a user is only interested in a list of diplomatic meetings. The query can be refined as *"Bush >meet > [person/name]."* In this case, the addition of an ontology tag restricts the number of meetings to those that are likely to involve named political personalities of some relevance. If the user is primarily interested in meeting that involve talks on nuclear arms control, the query can be further refined as *"Bush > meet > [person/name] ∼ nuclear arms control."* Similarly, the query *"[country] > produce > uranium"* can be turned into the query *"[country] > produce >[numeric] uranium"* if a user is after quantities of uranium that are being produced around the world. In general, we observe that users accustomed to keyword search believe that specifying more terms translates into more accurate results. In moving these users to Fact Search we must encourage them to start as simple as possible, since the IQL can express in two words what would take 20 lines using Boolean language.

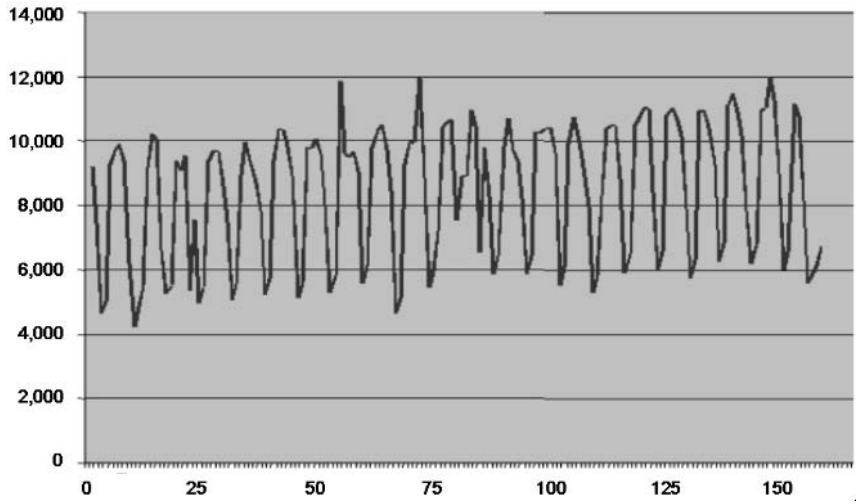


Fig. 5.11. Queries/day vs day of operation (June 22, 2005, to November 30, 2005).

Finally, Figure 5.11 shows overall query volumes (keyword search and Fact Search) as a function of day from the first day of operation (June 22 to November 30, 2005). The cyclic nature of the graph derives from the fact that most user access the site during the working week. Figure 5.12, which displays query volumes vs week of operation, clearly shows a positive trend: overall traffic to the site has increased by almost 40% ever since we introduced InFact search. The most interesting metrics relate to the percentage of users that derive value from Fact Search. The most effective mechanism to promote natural language search, as we have seen, are the tips. Figure 5.13 shows a 60% increase in the number of users that click on the tips automatically generated by InFact's advanced linguistic analysis over our entire period of operation. The overall percentage has increased from 4% to 10%. Our analysis also suggests that the best way to teach users how to write good queries is to first expose them to the summary result displays that ensues from a natural language query. The sooner users become aware of the type of results that a natural

language query can yield, the higher the chances that they learn how to use the new search functions correctly. This reinforces the idea that the result display may be a driver of Fact Search.

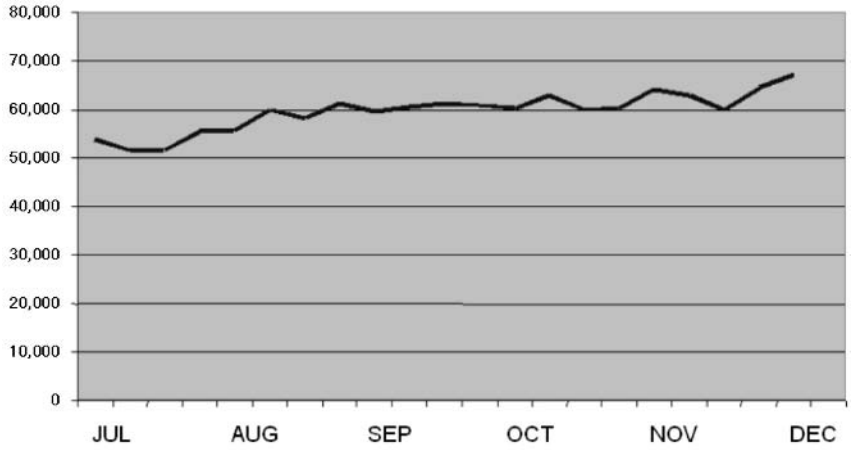


Fig. 5.12. Queries/week vs week of operation (June to November, 2005).

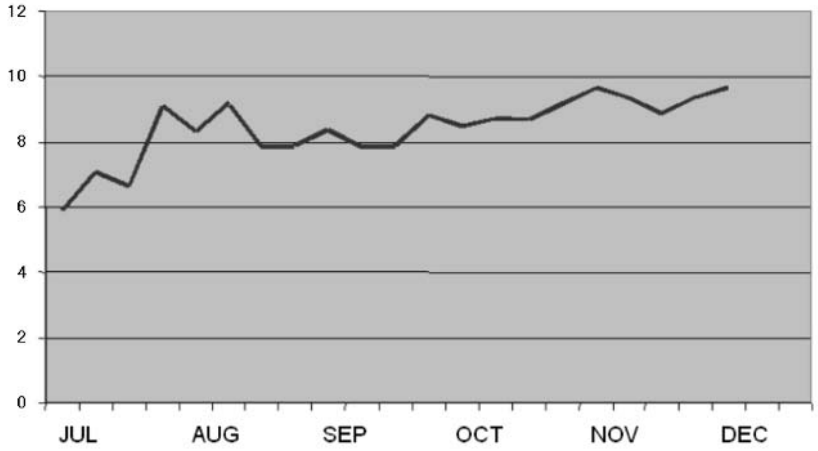


Fig. 5.13. Percentage of tips clicked (June to November, 2005).

5.5 Conclusion

We deployed a natural language based search to a community of Web users, and measured its popularity relative to conventional keyword search. Our work addressed criticisms of NLP approaches to search to the effect that they are not scalable and are too complex to be usable by average end-users. Our approach rests on a sophisticated index parameterization of text content, that captures syntactic and semantic roles, in addition to keyword counts, and enables interactive search and retrieval of events patterns based on a combination of keyword distributions and natural language attributes. Our distributed indexing and search services are designed to scale to large document collections and large numbers of users. We successfully deployed on a Web site that serves a community of 100,000 users. An analysis of query logs shows that, during the first six months of operation, traffic has increased by almost 40%. Even more significantly, we are encountering some success in promoting natural language searches. Our study demonstrates that the percentage of users that avail themselves of guided fact navigation based on natural language understanding has increased from 4% to 10% during the first six months of operation. Going forward, we will focus on increasing this percentage with a more innovative UI.

5.6 Acknowledgments

This work was partially supported by Dr. Joseph Psotka of the US Army Research Institute under contract No. W74V8H-05-C-0016. We are also indebted to John Pike, director of GlobalSecurity.org and his staff for providing us with many months of user traffic Web logs prior to going live.

References

1. D. Appelt and D. Israel. Introduction to information extraction technology. IJCAI-99 tutorial. http://www.ai.sri.com/∼appelt/ie-tutorial/ijcai99.pdf.
2. D. Appelt and D. Israel. Semantic approaches to binding theory. In *Proceedings of the Workshop on Semantic Approaches to Binding Theory. ESSLLI*, 2003.
3. A. Arampatzis, T. van der Weide, P. van Bommel, and C. Koster. Linguistically-motivated information retrieval. In M. Dekker, editor, *Encyclopedia of Library and Information Science*, Springer Verlag, volume 69, pages 201–222. 2000.
4. C. F. Baker, C. J. Fillmore, and J. B. Lowe. The Berkeley FrameNet project. In C. Boitet and P. Whitelock, editors, *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*, pages 86–90, San Francisco, California, 1998. Morgan Kaufmann Publishers.
5. I. Dagan, O. Glickman, and B. Magnini. The pascal recognizing textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop Recognizing Textual Entailment*, 2005.
6. M. Dimitrov. A light-weight approach to coreference resolution for named entities in text. Master's thesis, University of Sofia, 2002.

7. D. Gildea and D. Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288, 2002.
8. D. Gildea and M. Palmer. The necessity of parsing for predicate argument recognition. In *Proceedings of the 40th Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 239–246, 2002.
9. GlobalSecurity.org. http://www.globalsecurity.org/org/overview/history.htm.
10. M. Kameyama. Recognizing referential links: An information extraction perspective. In *Proceedings of the ACL'97/EACL'97 Workshop on Operation Factors in Practical, Robust Anaphora Resolution*, pages 46–53, 1997.
11. A. Kao and S. Poteet. Report on KDD conference 2004 panel discussion can natural language processing help text mining? *SIGKDD Explorations*, 6(2):132–133, 2004.
12. C. Kennedy and B. Boguraev. Anaphora for everyone: Pronominal anaphora resolution without a parser. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING'96)*, pages 113–118, 1996.
13. S. Lappin and H. Leass. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561, 1994.
14. D. Lin and P. Pantel. DIRT - discovery of inference rules from text. In *Knowledge Discovery and Data Mining*, pages 323–328, 2001.
15. C. Manning and H. Schutze. *Foundation of Statistical Natural Language Processing.* The MIT Press, 2000.
16. R. Miltov. Robust pronoun resolution with limited knowledge. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING'98)/ACL'98*, pages 869–875.
17. R. Miltov. Anaphora resolution: The state of the art. Working paper. University of Wolverhamption, 1999.
18. National Institute of Standards and Technology. Automatic content extraction (ACE). http://www.itl.nist.gov/iaui/894.01/tests/ace.
19. M. Surdeanu, S. Harabagiu, J. Williams, and P. Aarseth. Using predicate-argument structures for information extraction. In *41th Annual Meeting of the Association for Computational Linguistics*, pages 8–15, 2003.

6

Evaluating Self-Explanations in iSTART: Word Matching, Latent Semantic Analysis, and Topic Models

Chutima Boonthum, Irwin B. Levinstein, and Danielle S. McNamara

6.1 Introduction

iSTART (Interactive Strategy Trainer for Active Reading and Thinking) is a web-based, automated tutor designed to help students become better readers via multimedia technologies. It provides young adolescent to college-aged students with a program of self-explanation and reading strategy training [19] called Self-Explanation Reading Training, or SERT [17, 21, 24, 25]. The reading strategies include (a) comprehension monitoring, being aware of one's understanding of the text; (b) paraphrasing, or restating the text in different words; (c) elaboration, using prior knowledge or experiences to understand the text (i.e., domain-specific knowledge-based inferences) or common sense, using logic to understand the text (i.e., domain-general knowledge based inferences); (d) predictions, predicting what the text will say next; and (e) bridging, understanding the relation between separate sentences of the text. The overall process is called "self-explanation" because the reader is encouraged to explain difficult text to him- or herself. iSTART consists of three modules: Introduction, Demonstration, and Practice. In the last module, students practice using reading strategies by typing self-explanations of sentences. The system evaluates each self-explanation and then provides appropriate feedback to the student. If the explanation is irrelevant or too short, the student is required to add more information. Otherwise, the feedback is based on the level of overall quality.

The computational challenge here is to provide appropriate feedback to the students concerning their self-explanations. To do so requires capturing some sense of both the meaning and quality of the self-explanation. Interpreting text is critical for intelligent tutoring systems, such as iSTART, that are designed to interact meaningfully with, and adapt to, the users' input. iSTART was initially proposed as using Latent Semantic Analysis (LSA; [13]) to capture the meanings of texts and to assess the students' self-explanation; however, while the LSA algorithms were being built, iSTART used simple word matching algorithms. In the course of integrating the LSA algorithms, we found that a combination of word-matching and LSA provided better results than either separately [18].

Our goal in evaluating the adequacy of the algorithms has been to imitate experts' judgments of the quality of the self-explanations. The current evaluation system predicts the score that a human gives on a 4-point scale, where 0 represents an

evaluation of the explanation as irrelevant or too short; 1, minimally acceptable; 2, better but including primarily the local textual context; and 3, oriented to a more global comprehension. Depending on the text, population, and LSA space used, our results have ranged from 55 to 70 percent agreement with expert evaluations using that scale. We are currently attempting to improve the effectiveness of our algorithms by incorporating Topic Models (TM) either in place of or in conjunction with LSA and by using more than one LSA space from different genres (science, narrative, and general TASA corpus). We present some of the results of these efforts in this chapter.

Our algorithms are constrained by two major requirements, speedy response times and speedy introduction of new texts. Since the trainer operates in real time, the server that calculates the evaluation must respond in 4 to 5 seconds. Furthermore the algorithms must not require any significant preparation of new texts, a requirement precisely contrary to our plans when the project began. In order to accommodate the needs of the teachers whose classes use iSTART, the trainer must be able to use texts that the teachers wish their students to use for practice within a day or two. This time limit precludes us from significantly marking up the text or gathering related texts to incorporate into an LSA corpus.

In addition to the overall 4-point quality score, we are attempting to expand our evaluation to include an assessment of the presence of various reading strategies in the student's explanation so that we can generate more specific feedback. If the system were able to detect whether the explanation uses paraphrasing, bridging, or elaboration we could provide more detailed feedback to the students, as well as an individualized curriculum based on a more complete model of the student. For example, if the system were able to assess that the student only paraphrased sentences while self-explaining, and never used strategies such as making bridging inferences or knowledge-based elaborations, then the student could be provided additional training to generate more inference-based explanations.

This chapter describes how we employ word matching, LSA, and TM in the iSTART feedback systems and the performance of these techniques in producing both overall quality and reading strategy scores.

6.2 iSTART: Feedback Systems

iSTART was intended from the outset to employ LSA to determine appropriate feedback. The initial goal was to develop one or more benchmarks for each of the SERT strategies relative to each of the sentences in the practice texts and to use LSA to measure the similarity of a trainee's explanation to each of the benchmarks. A benchmark is simply a collection of words, in this case, words chosen to represent each of the strategies (e.g., words that represent the current sentence, words that represent a bridge to a prior sentence). However, while work toward this goal was progressing, we also developed a preliminary "word-based" (WB) system to provide feedback in our first version of iSTART [19] so that we could provide a complete curriculum for use in experimental situations. The second version of iSTART has integrated both LSA and WB in the evaluation process; however, the system still provides only overall quality feedback. Our current investigations aim to provide feedback based on identifying specific reading strategies.

6.2.1 Word Matching Feedback Systems

Word matching is a very simple and intuitive way to estimate the nature of a self-explanation. In the first version of iSTART, several hand-coded components were built for each practice text. For example, for each sentence in the text, the "important words" were identified by a human expert and a length criterion for the explanation was manually estimated. Important words were generally content words that were deemed important to the meaning of the sentence and could include words not found in the sentence. For each important word, an association list of synonyms and related terms was created by examining dictionaries and existing protocols as well as by human judgments of what words were likely to occur in a self-explanation of the sentence. In the sentence "All thunderstorms have a similar life history," for example, important words are *thunderstorm, similar, life*, and *history*. An association list for *thunderstorm* would include *storms, moisture, lightning, thunder, cold, tstorm, t-storm, rain, temperature, rainstorms*, and *electric-storm*. In essence, the attempt was made to imitate LSA.

A trainee's explanation was analyzed by matching the words in the explanation against the words in the target sentence and words in the corresponding association lists. This was accomplished in two ways: (1) Literal word matching and (2) Soundex matching.

Literal word matching - Words are compared character by character and if there is a match of the first 75% of the characters in a word in the target sentence (or its association list) then we call this a literal match. This also includes removing suffix -s, -d, -ed, -ing, and -ion at the end of each words. For example, if the trainee's self-explanation contains 'thunderstom' (even with the misspelling), it still counts as a literal match with words in the target sentence since the first nine characters are exactly the same. On the other hand, if it contains 'thunder,' it will not get a match with the target sentence, but rather with a word on the association list.

Soundex matching - This algorithm compensates for misspellings by mapping similar characters to the same soundex symbol [1, 5]. Words are transformed to their soundex code by retaining the first character, dropping the vowels, and then converting other characters into soundex symbols. If the same symbol occurs more than once consecutively, only one occurrence is retained. For example, 'thunderstorm' will be transformed to 't8693698'; 'communication' to 'c8368.' Note that the later example was originally transformed to 'c888368' and two 8s were dropped ('m' and 'n' are both mapped to '8'). If the trainee's self-explanation contains 'thonderstorm' or 'tonderstorm,' both will be matched with 'thunderstorm' and this is called a soundex match. An exact soundex match is required for short words (i.e., those with fewer than six alpha-characters) due to the high number of false alarms when soundex is used. For longer words, a match on the first four soundex symbols suffices. We are considering replacing this rough and ready approach with a spell-checker.

A formula based on the length of the sentence, the length of the explanation, the length criterion mentioned below, the number of matches to the important words, and the number of matches to the association lists produces a rating of 0 (inadequate), 1 (barely adequate), 2 (good), or 3 (very good) for the explanation. The rating of 0 or inadequate is based on a series of filtering criteria that assesses whether the explanation is too short, too similar to the original sentence, or irrelevant. *Length*

is assessed by a ratio of the number of words in the explanation to the number in the target sentence, taking into consideration the length criterion. For example, if the length of the sentence is 10 words and the length priority is 1, then the required length of the self-explanation would be 10 words. If the length of the sentence is 30 words and the length priority is 0.5, then the self-explanation would require a minimum of 15 words. *Relevance* is assessed from the number of matches to important words in the sentence and words in the association lists. *Similarity* is assessed in terms of a ratio of the sentence and explanation lengths and the number of matching important words. If the explanation is close in length to the sentence, with a high percentage of word overlap, the explanation would be deemed too similar to the target sentence. If the explanation failed any of these three criteria (Length, Relevance, and Similarity), the trainee would be given feedback corresponding to the problem and encouraged to revise the self-explanation.

Once the explanation passes the above criteria, then it is evaluated in terms of its overall quality. The three levels of quality that guide feedback to the trainee are based on two factors: 1) the number of words in the explanation that match either the important words or association-list words of the target sentence compared to the number of important words in the sentence and 2) the length of the explanation in comparison with the length of the target sentence. This algorithm will be referred as *WB-ASSO*, which stands for *word-based with association list.*

This first version of iSTART (word-based system) required a great deal of human effort per text, because of the need to identify important words and, especially, to create an association list for each important word. However, because we envisioned a scaled-up system rapidly adaptable to many texts, we needed a system that required relatively little manual effort per text. Therefore, WB-ASSO was replaced. Instead of lists of important and associated words we simply used content words (nouns, verbs, adjectives, adverbs) taken literally from the sentence and the entire text. This algorithm is referred to as *WB-TT*, which stands for *word-based with total text.* The content words were identified using algorithms from Coh-Metrix, an automated tool that yields various measures of cohesion, readability, other characteristics of language [9, 20]. The iSTART system then compares the words in the self-explanation to the content words from the current sentence, prior sentences, and subsequent sentences in the target text, and does a word-based match (both literal and soundex) to determine the number of content words in the self-explanation from each source in the text. While WB-ASSO is based on a richer corpus of words than WB-TT, the replacement was successful because the latter was intended for use together with LSA which incorporates the richness of a corpus of hundreds of documents. In contrast, WB-ASSO was used on its own.

Some hand-coding remained in WB-TT because the length criterion for an explanation was calculated based on the average length of explanations of that sentence collected from a separate pool of participants and on the importance of the sentence according to a manual analysis of the text. Besides being relatively subjective, this process was time consuming because it required an expert in discourse analysis as well as the collection of self-explanation protocols. Consequently, the hand-coded length criterion was replaced with one that could be determined automatically from the number of words and content words in the target sentence (we called this *word-based with total text and automated criteria*, or *WB2-TT*). The change from WB-TT to WB2-TT affected only the screening process of the length and similarity criteria. Its lower-bound and upper-bound lengths are entirely based on the target sentence's

length. The overall quality of each self-explanation (1, 2, or 3) is still computed with the same formula used in WB-TT.

6.2.2 Latent Semantic Analysis (LSA) Feedback Systems

Latent Semantic Analysis (LSA; [13, 14]) uses statistical computations to extract and represent the meaning of words. Meanings are represented in terms of their similarity to other words in a large corpus of documents. LSA begins by finding the frequency of terms used and the number of co-occurrences in each document throughout the corpus and then uses a powerful mathematical transformation to find deeper meanings and relations among words. When measuring the similarity between text-objects, LSA's accuracy improves with the size of the objects. Hence, LSA provides the most benefit in finding similarity between two documents. The method, unfortunately, does not take into account word order; hence, very short documents may not be able to receive the full benefit of LSA.

To construct an LSA corpus matrix, a collection of documents are selected. A document may be a sentence, a paragraph, or larger unit of text. A term-document-frequency (TDF) matrix X is created for those terms that appear in two or more documents. The row entities correspond to the words or terms (hence the W) and the column entities correspond to the documents (hence the D). The matrix is then analyzed using Singular Value Decomposition (SVD; [26]), that is the TDF matrix X is decomposed into the product of three other matrices: (1) vectors of derived orthogonal factor values of the original row entities W, (2) vectors of derived orthogonal factor values of the original column entities D, and (3) scaling values (which is a diagonal matrix) S. The product of these three matrices is the original TDF matrix.

$$\{X\} = \{W\}\{S\}\{D\} \tag{6.1}$$

The dimension (d) of $\{S\}$ significantly affects the effectiveness of the LSA space for any particular application. There is no definite formula for finding an optimal number of dimensions; the dimensionality can be determined by sampling the results of using the matrix $\{W\}\{S\}$ to determine the similarity of previously-evaluated document pairs for different dimensionalities of $\{S\}$. The optimal size is usually in the range of 300-400 dimensions.

The similarity of terms is computed by taking the cosine of the corresponding term vectors. A term vector is the row entity of that term in the matrix W. In iSTART, the documents are sentences from texts and trainees' explanations of those sentences. These documents consist of terms, which are represented by term vectors; hence, the document can be represented as a document vector which is computed as the sum of the term vectors of its terms:

$$D_i = \sum_{t=1}^{n} T_{ti} \tag{6.2}$$

where D_i is the vector for the i^{th} document D, T_{ti} is the term vector for the term t in D_i, and n is number of terms in D. The similarity between two documents (i.e., the cosine between the two document vectors) is computed as

$$Sim(D1, D2) = \frac{\sum_{i=1}^{d}(D1_i \times D2_i)}{\sum_{i=1}^{d}(D1_i)^2 \times \sum_{i=1}^{d}(D2_i)^2} \tag{6.3}$$

Since the first versions of iSTART were intended to improve students' comprehension of science texts, the LSA space was derived from a collection of science texts [11]. This corpus consists of 7,765 documents containing 13,502 terms that were used in two or more documents. By the time the first version of the LSA-based system was created (referred to as *LSA1*), the original goal of identifying particular strategies in an explanation had been replaced with the less ambitious one of rating the explanation as belonging one of three levels [22]. The highest level of explanation, called "*global-focused*," integrates the sentence material in a deep understanding of the text. A "*local-focused*" explanation explores the sentence in the context of its immediate predecessors. Finally, a "*sentence-focused*" explanation goes little beyond paraphrasing. To assess the level of an explanation, it is compared to four benchmarks or bags of words. The rating is based on formulae that use weighted sums of the four LSA cosines between the explanation and each of the four benchmarks.

The four benchmarks include: 1) the words in the title of the passage ("title"), 2) the words in the sentence ("current sentence"), 3) words that appear in prior sentences in the text that are causally related to the sentence ("prior text"), and 4) words that did not appear in the text but were used by two or more subjects who explained the sentence during experiments ("world knowledge"). While the title and current sentence benchmarks are created automatically, the prior-text benchmark depends on a causal analysis of the conceptual structure of the text, relating each sentence to previous sentences. This analysis requires both time and expertise. Furthermore, the world-knowledge benchmark requires the collection of numerous explanations of each text to be used. To evaluate the explanation of a sentence, the explanation is compared to each benchmark, using the similarity function mentioned above. The result is called a cosine value between the self-explanation (SE) and the benchmark. For example, *Sim(SE, Title)* is called the *title LSA cosine*. Discriminant Analysis was used to construct the formulae that categorized the overall quality as being a level 1, 2, or 3 [23]. A score is calculated for each of the levels using these formulae. The highest of the three scores determines the predicted level of the explanation. For example, the overall quality score of the explanation is a 1 if the level-1 score is higher than both the level-2 and level-3 scores.

Further investigation showed that the LSA1 cosines and the factors used in the WB-ASSO approach could be combined in a discriminant analysis that resulted in better predictions of the values assigned to explanations by human experts. However, the combined approach was less than satisfactory. Like WB-ASSO, LSA1 was not suitable for an iSTART program that would be readily adaptable to new practice texts. Therefore, we experimented with formulae that would simplify the data gathering requirements to develop LSA2. Instead of the four benchmarks mentioned above, we discarded the world knowledge benchmark entirely and replaced the benchmark based on causal analysis of prior-text with one that simply consisted of the words in the previous two sentences. We could do this because the texts were taken from science textbooks whose argumentation tends to be highly linear argumentation in science texts; consequently the two immediately prior sentences

worked well as stand-ins for the set of causally related sentences. It should be noted that this approach may not succeed so well with other genres, such as narrative or history texts.

We tested several systems that combined the use of word-matching and LSA2 and the best one is LSA2/WB2-TT. In these combinatory systems, we combine a weighted sum of the factors used in the fully automated word-based systems and LSA2. These combinations allowed us to examine the benefits of using the world knowledge benchmark (in LSA1) when LSA was combined with a fully automated word-based system and we found that world knowledge benchmark could be dropped. Hence, only three benchmarks are used for LSA-based factors: 1) the words in the title of the passage, 2) the words in the sentence, and 3) the words in the two immediately prior sentences. From the word-based values we include 4) the number of content words matched in the target sentence, 5) the number of content words matched in the prior sentences, 6) the number of content words matched in the subsequent sentences, and 7) the number of content words that were not matched in 4, 5, or 6. One further adjustment was made because we noticed that the LSA approach alone was better at predicting higher values correctly, while the word-based approach was better at predicting lower values. Consequently, if the formulae of the combined system predicted a score of 2 or 3, that value is used. However, if the system predicted a 1, a formula from the word-based system is applied. Finally, level 0 was assigned to explanations that had negligible cosine matches with all three LSA benchmarks.

6.2.3 Topic Models (TM) Feedback System

The Topic Models approach (TM; [10, 27]) applies a probabilistic model in finding a relationship between terms and documents in terms of topics. A document is conceived of as having been generated probabilistically from a number of topics and each topic consists of number of terms, each given a probability of selection if that topic is used. By using a TM matrix, we can estimate the probability that a certain topic was used in the creation of a given document. If two documents are similar, the estimates of the topics they probably contain should be similar. TM is very similar to LSA, except that a term-document frequency matrix is factored into two matrices instead of three.

$$\{X_{normalized}\} = \{W\}\{D\} \tag{6.4}$$

The dimension of matrix $\{W\}$ is W x T, where W is the number of words in the corpus and T is number of topics. The number of topics varies, more or less, with the size of corpus; for example, a corpus of 8,000 documents may require only 50 topics while a corpus of 40,000 documents could require about 300 topics. We use the TM Toolbox [28] to generate the $\{W\}$ or TM matrix, using the same science corpus as we used for the LSA matrix. In this construction, the matrix $\{X\}$ is for all terms in the corpus, not just those appearing in two different documents. Although matrix $\{X\}$ is supposed to be normalized, the TM toolbox takes care of this normalization and outputs for each topic, the topic probability, and a list of terms in this topic along with their probabilities in descending order (shown in Table 6.1). This output is easily transformed into the term-topic-probability matrix.

Table 6.1. Results from Topic Models Toolbox: science corpus, 50 topics, seed 1, 500 iteration, default alpha and beta.

TOPIC 2 0.0201963151	TOPIC 38 0.0214418635
earth 0.1373291184	light 0.1238061875
sun 0.0883152826	red 0.0339683946
solar 0.0454833721	color 0.0307797075
atmosphere 0.0418036547	white 0.0262046347
moon 0.0362104843	green 0.0230159476
surface 0.0181062747	radiation 0.0230159476
planet 0.0166343877	wavelengths 0.0230159476
center 0.0148681234	blue 0.0184408748
bodies 0.0147209347	dark 0.0178863206
tides 0.0139849912	visible 0.0170544891
planets 0.0133962364	spectrum 0.0151135492
gravitational 0.0125131042	absorbed 0.0149749106
system 0.0111884060	colors 0.0148362720
appear 0.0110412173	rays 0.0116475849
mass 0.0100108964	eyes 0.0108157535
core 0.0083918207	yellow 0.0105384764
space 0.0083918207	absorption 0.0102611992
times 0.0079502547	eye 0.0095680064
orbit 0.0073614999	pigment 0.0092907293
...	...

To measure the similarity between documents based on TM, the Kullback Liebler distance (KL-distance: [27]) between two documents is recommended, rather than the cosine (which, nevertheless, can be used). A document can be represented by a set of probabilities that this document could contain topic i using the following

$$D_t = \sum_{i=1}^{n} T_{it} \tag{6.5}$$

where D_t is the probability of topic t in the document D, T_{it} is the probability of topic t of the term i in the document D, and n is number of terms appearing in the document D. The KL-distance between two documents (the similarity) is computed as follows:

$$KL(D1, D2) = \frac{1}{2} \sum_{t=1}^{T} D1_t log_2(D1_t/D2_t) + \frac{1}{2} \sum_{t=1}^{T} D2_t log_2(D2_t/D1_t) \tag{6.6}$$

Constructing a TM matrix involves making choices regarding a number of factors, such as the number of topics, the seed for random number generation, alpha, beta, and the number of iterations. We have explored these factors and constructed a number of TM matrices in an effort to optimize the resulting matrix; however, for this preliminary evaluation, we use a TM matrix of 50 topics and a seed of 1.

The first TM-based system we tried was simply used in place of the LSA-based factors in the combined-system. The three benchmarks are still the same but sim-

ilarity is computed in two ways: (1) using cosines — comparing the explanation and the benchmark using the cosine formula (Referred as TM1) and (2) using KL distances — comparing the explanation and the benchmark using the KL distance (Referred as TM2). As before, formulae are constructed using Discriminant Analysis in order to categorize the quality of explanation as Levels 1, 2, or 3.

6.2.4 Metacognitive Statements

The feedback systems include a metacognitive filter that searches the trainees' self-explanations for patterns indicating a description of the trainee's mental state such as "now I see ..." or "I don't understand this at all." While the main purpose of the filter is to enable the system to respond to such non-explanatory content more appropriately, we also used the same filter to remove "noise" such as "What this sentence is saying is ..." from the explanation before further processing. We have examined the effectiveness of the systems with and without the filter and found that they all perform slightly better with than without it. Thus, the systems in this chapter all include the metacognitive filter.

The metacognitive filter also benefits the feedback system. When a metacognitive pattern is recognized, its category is noted. If the self-explanation contains only a metacognitive statement, the system will respond to a metacognitive category such as *understanding, not-understanding, confirmation, prediction,* or *boredom* instead of responding irrelevantly. Regular expressions are used to define multiple patterns for each metacognitive category. If any pattern is matched in the self-explanation, words matching the pattern are removed before evaluation. Examples of regular expression are shown below:

```
NOTUNDERSTAND :i(?:.?m|\W+am)(?:\W+\w+)?\W+\W+(?:(?:not
                (?:\W+\w+)?\W+(?:sure|certain|clear))|
                un(?:sure|certain|clear))
UNDERSTAND :now\W+i\W+(?:know|knew|underst(?:an|oo)d|
                remember(?:ed)?|recall(?:ed)?|recogniz(?:ed)?|get|
                got|see)
CONF :(?:so\W+)?i\W+(?:was|got\W+it)\W+(?:right|correct)
```

The first pattern will include "I'm not sure," "I am uncertain"; second pattern includes "Now I understand," "Now I remembered"; and the last pattern includes "So, I was right." We originally constructed over 60 patterns. These were reduced to 45 by running them on a large corpus of explanations and eliminating those that failed to match and adding those that were missed.

6.3 iSTART: Evaluation of Feedback Systems

Two experiments were used to evaluate the performance of various systems of algorithms that vary as a function of approach (word-based, LSA, combination of word-based and LSA, and combination of word-based TM). In Experiment 1, we

compare all eight systems in terms of the overall quality score by applying each system to a database of self-explanation protocols produced by college students. The protocols had been evaluated by a human expert on overall quality. In Experiment 2, we investigated two systems using a database of explanations produced by middle-school students. These protocols were scored to identify particular reading strategies.

6.3.1 Experiment 1

Self-Explanations. The self-explanations were collected from college students who were provided with SERT training and then tested with two texts, Thunderstorm and Coal. Both texts consisted of 20 sentences. The Thunderstorm text was self-explained by 36 students and the Coal text was self-explained by 38 students. The self-explanations were coded by an expert according to the following 4-point scale: 0 = vague or irrelevant; 1 = sentence-focused (restatement or paraphrase of the sentence); 2 = local-focused (includes concepts from immediately previous sentences); 3 = global-focused (using prior knowledge).

The coding system was intended to reveal the extent to which the participant elaborated the current sentence. Sentence-focused explanations do not provide any new information beyond the current sentence. Local-focused explanations might include an elaboration of a concept mentioned in the current or immediately prior sentence, but there is no attempt to link the current sentence to the theme of the text. Self-explanations that linked the sentence to the theme of the text with world knowledge were coded as "global-focused." Global-focused explanations tend to use multiple reading strategies, and indicate the most active level of processing.

Results. Each of the eight systems produces an evaluation comparable to the human ratings on a 4-point scale. Hence, we calculated the correlations and percent agreement between the human and system evaluations (see Table 6.2). Additionally, d primes (d′s) were computed for each strategy level as a measure of how well the system could discriminate among the different levels of strategy use. The d′s were computed from hit and false-alarm rates. A hit would occur if the system assigned the same self-explanation to a category (e.g., global-focused) as the human judges. A false-alarm would occur if the system assigned the self-explanation to a category (e.g., global-focused) that was different from the human judges (i.e., it was not a global-focused strategy). d′s are highest when hits are high and false-alarms are low. In this context, d′s refer to the correspondence between the human and system in standard deviation units. A d′ of 0 indicates chance performance, whereas greater d′s indicate greater correspondence.

One thing to note in Table 6.3 is that there is general improvement according to all of the measures going from left to right. As might be expected, the systems with LSA fared far better than those without LSA, and the combined systems were the most successful. The word-based systems tended to perform worse as the evaluation level increased (from 0 to 3), but performed relatively well at identifying poor self-explanations and paraphrases. All of the systems, however, identified the sentence-focused (i.e., 2's) explanations less successfully. However, the d′s for the sentence focused explanations approach 1.0 when LSA is incorporated, particularly when LSA is combined with the word-based algorithms.

Apart from better performance with LSA than without, the performance is also more stable with LSA. Whereas the word-based systems did not perform equally

Table 6.2. Measures of agreement for the Thunderstorm and Coal texts between the eight system evaluations and the human ratings of the self-explanations in Experiment 1.

Thunderstorm Text	WB-ASSO	WB-TT	WB2-TT	LSA1	LSA2	LSA2/ WB2-TT	TM1	TM2
Correlation	0.47	0.52	0.43	0.60	0.61	0.64	0.56	0.58
% Agreement	48%	50%	27%	55%	57%	62%	59%	60%
d' of 0's	2.21	2.26	0.97	2.13	2.19	2.21	1.49	2.37
d' of 1's	0.84	0.79	0.66	1.32	1.44	1.45	1.27	1.39
d' of 2's	0.23	0.36	-0.43	0.47	0.59	0.85	0.74	0.70
d' of 3's	1.38	1.52	1.41	1.46	1.48	1.65	1.51	1.41
Avg d'	1.17	1.23	0.65	1.34	1.43	1.54	1.25	1.23

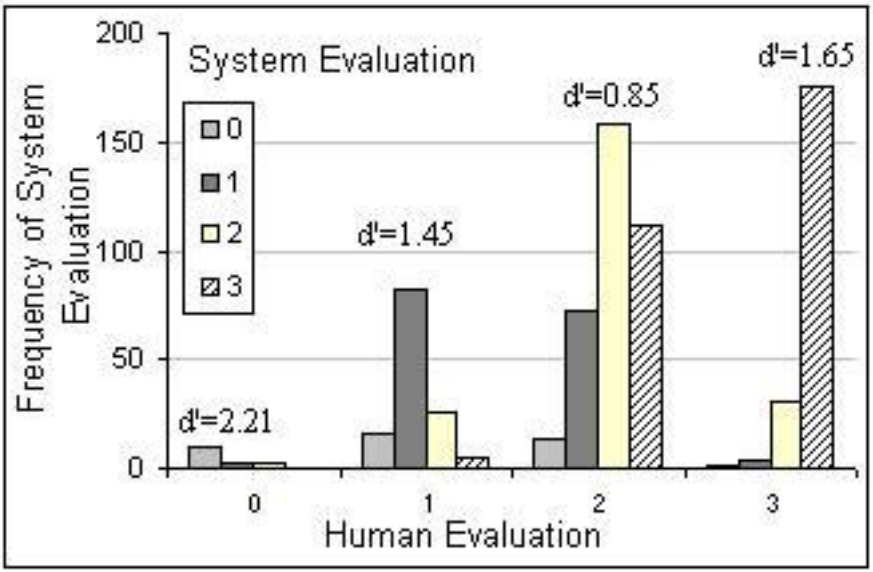
Coal Text	WB-ASSO	WB-TT	WB2-TT	LSA1	LSA2	LSA2/ WB2-TT	TM1	TM2
Correlation	0.51	0.47	0.41	0.66	0.67	0.71	0.63	0.61
% Agreement	41%	41%	29%	56%	57%	64%	61%	61%
d' of 0's	4.67	4.73	1.65	2.52	2.99	2.93	2.46	2.05
d' of 1's	1.06	0.89	0.96	1.21	1.29	1.50	1.38	1.52
d' of 2's	0.09	0.13	-0.37	0.45	0.49	0.94	0.74	0.61
d' of 3's	-0.16	1.15	1.28	1.59	1.59	1.79	1.60	1.50
Avg d'	1.42	1.73	0.88	1.44	1.59	1.79	1.54	1.42

well on the Thunderstorm and Coal texts, there is a high-level of agreement for the LSA-based formulas (i.e., the results are virtually identical in the two tables). This indicates that if we were to apply the word-based formulas to yet another text, we have less assurance of finding the same performance, whereas the LSA-based formulas are more likely to replicate across texts.

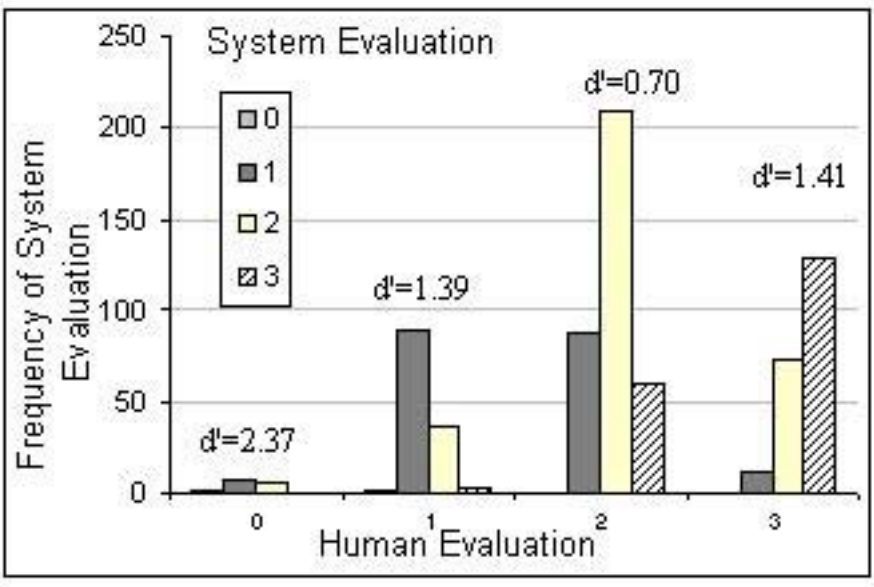
Figure 6.1.a provides a closer look at the data for the combined, automated system, LSA2/WB2-TT and Figure 6.1.b for the TM2 system. As the d′s indicated, both systems' performance is quite good for explanations that were given human ratings of 0, 1, or 3. Thus, the system successfully identifies poor explanations, paraphrases, and very good explanations. It is less successful for identifying explanations that consist of paraphrases in addition to some information from the previous sentence or from world knowledge. As one might expect, some are classified as paraphrases and some as global by the system. Although not perfect, we consider this result a success because so few were misclassified as poor explanations.

6.3.2 Experiment 2

Self-Explanations. The self-explanations were collected from 45 middle-school students (entering 8th and 9th grades) who were provided with iSTART training and then tested with two texts, Thunderstorm and Coal. The texts were shortened versions of the texts used in Experiment 1, consisting of 13 and 12 sentences, respectively. This chapter presents only the data from the Coal text.



a) LSA2/WB2-TT — LSA with Word-based



b) TM2 — Topic Models with KL distance

Fig. 6.1. Correspondence between human evaluations of the self-explanations and the combined system (LSA2/WB2-TT and TM2) for Thunderstorm text. Explanations were evaluated by humans as vague or irrelevant (0), sentence-focused (1), local-focused (2), or global (3).

The self-explanations from this text were categorized as paraphrases, irrelevant elaborations, text-based elaborations, or knowledge-based elaborations. Paraphrases did not go beyond the meaning of the target sentence. Irrelevant elaborations may have been related to the sentence superficially or tangentially, but were not related to the overall meaning of the text and did not add to the meaning of the text. Text-based elaborations included bridging inferences that made links to information presented in the text prior to the sentence. Knowledge-based elaborations included the use of prior knowledge to add meaning to the sentence. This latter category is analogous to, but not the same as, the global-focused category in Experiment 1.

Results. In contrast to the human coding system used in Experiment 1, the coding system applied to this data was not intended to map directly onto the iSTART evaluation systems. In this case, the codes are categorical and do not necessarily translate to a 0-3 quality range. One important goal is to be able to assess (or discriminate) the use of reading strategies and improve the system's ability to appropriately respond to the student. This is measured in terms of percent agreement with human judgments of each reading strategy shown in Table 6.3.

Table 6.3. Percent agreement to expert ratings of the self-explanations to the Coal text for the LSA2/WB2-TT and TM2 combined systems for each reading strategy in Experiment 2.

Reading Strategy	LSA2/WB2-TT	TM2
Paraphrase Only	69.9	65.8
Irrelevant Elaboration Only	71.6	76.0
Current Sentence Elaboration Only	71.9	71.2
Knowledge-Based Elaboration Only	94.6	90.3
Paraphrase + Irrelevant Elaboration	79.7	76.6
Paraphrase + Current Sentence Elaboration	68.2	67.3
Paraphrase + Knowledge-Based Elaboration	84.6	81.2

The results show that both systems perform very well, with an average of 77% for the LSA2/WB2-TT system and 75% for the TM2 system. This approaches our criteria of 85% agreement between trained experts who score the self-explanations. The automated systems could be thought of as 'moderately trained scorers.' These results thus show that either of these systems would guide appropriate feedback to the student user.

The score for each strategy score (shown in Table 6.3) can be coded either 0=present or 1=present. With the current coding scheme, only one strategy (out of seven) will be given a value of 1. We are currently redefining the coding scheme so that each reading strategy will have its own scores. For example, if the explanation contains both paraphrase and current sentence elaboration, with the current coding scheme, "Paraphrase + Current Sentence Elaboration" will be coded as a 1. On the other hand, with the new coding scheme, we will have at least 3 variables: (1) "Paraphrase" will be coded as a 1 for *present*, (2) "Elaboration" coded as a 1 for *present*, and (3) "Source of Elaboration" coded as a 2 for *current sentence elaboration*.

6.4 Discussion

The purpose of this chapter has been to investigate the ability of topic model algorithms to identify the quality of explanations as well as specific reading strategies in comparison to word-based and LSA-based algorithms. We found in Experiment 1 that TM systems performed comparably to the combined systems, though not quite as well. In Experiment 2, we found that the TM models performed nearly as well as the combined system in identifying specific strategies. These results thus broaden the scope of NLP models that can be applied to problems such as ours — providing real-time feedback in a tutoring environment. Indeed, the performance of both systems in Experiment 2 was highly encouraging. These results indicate that future versions of iSTART will be able to provide specific feedback about reading comprehension strategy use with relatively high confidence.

Our future work with the TM systems will be to attempt to combine the TM algorithms with the LSA and word-based algorithms. To venture toward that goal, we need to first identify the strengths of the TM algorithms so that the combined algorithm capitalizes on the strengths of the TM — much as we did when we created the combined word-based and LSA-based system. This will require that we analyze a greater variety of protocols, including self-explanations from a greater variety of texts and text genres. We are in the process of completing that work.

These NLP theories and their effectiveness have played important roles in the development of iSTART. For iSTART to effectively teach reading strategies, it must be able to deliver valid feedback on the quality of the self-explanations that a student types during practice. In order to deliver feedback, the system must understand, at least to some extent, what a student is saying in his or her self-explanation. Of course, automating natural language understanding has been extremely challenging, especially for non-restrictive content domains like self-explaining a text in which a student might say one of any number of things. Algorithms such as LSA opened up a horizon of possibilities to systems such as iSTART — in essence LSA provided a 'simple' algorithm that allowed tutoring systems to provide appropriate feedback to students (see [14]). The results presented in this chapter show that the topic model similarly offers a wealth of possibilities in natural language processing.

6.5 Acknowledgments

This project was supported by NSF (IERI Award number: 0241144) and its continuation funded by IES (IES Award number: R305G020018). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF and IES.

References

1. Birtwisle, M. (2002) The Soundex Algorithm. Retrieved from: http://www.comp.leeds.ac.uk/matthewb/ar32/basic_soundex.htm
2. Bransford, J., Brown, A., & Cocking, R., Eds. (2000). How people learn: Brain, mind, experience, and school. Washington, D.C.: National Academy Press. Online at: http://www.nap.edu/html/howpeople1/

3. Chi, M. T. H., De Leeuw, N., Chiu, M., & LaVancher, C. (1994). Eliciting self-explanations improves understanding. Cognitive Science, 18, 439-477.
4. Chi, M. T. H., Bassok, M., Lewis, M. W., Reimann, R., & Glaser, R. (1989). Self-explanation: How students study and use examples in learning to solve problems. Cognitive Science, 13, 145-182.
5. Christian. P. (1998) Soundex — can it be improved? Computers in Genealogy, 6 (5)
6. Graesser, A. C., Penumatsa, P., Ventura, M., Cai, Z., & Hu, X. (2005). Using LSA in AutoTutor: Learning through mixed-initiative dialogue in natural language. In T. Landauer, D.S., McNamara, S. Dennis, & W. Kintsch (Eds.), LSA: A Road to Meaning. Mahwah, NJ: Erlbaum.
7. Graesser, A. C., Hu, X., & McNamara, D. S. (2005). Computerized learning environments that incorporate research in discourse psychology, cognitive science, and computational linguistics. In A. F. Healy (Ed.), Experimental Cognitive Psychology and its Applications: Festschrift in Honor of Lyle Bourne, Walter Kintsch, and Thomas Landauer. Washington, D.C.: American Psychological Association.
8. Graesser, A. C., Hu, X., & Person, N. (2001). Teaching with the help of talking heads. In T. Okamoto, R. Hartley, Kinshuk, J. P. Klus (Eds.), Proceedings IEEE International Conference on Advanced Learning Technology: Issues, Achievements and Challenges (460-461).
9. Graesser, A. C., McNamara, D. S., Louwerse, M. M., & Cai, Z. (2004). CohMetrix: Analysis of text on cohesion and language. Behavior Research Methods, Instruments, and Computers, 36, 193-202.
10. Griffiths, T., & Steyvers, M. (2004). Finding Scientific Topics. Proceedings of the National Academy of Science, 101 (suppl. 1), 5228-5235.
11. Kurby, C.A., Wiemer-Hastings, K., Ganduri, N., Magliano, J.P., Millis, K.K., & McNamaar, D.S. (2003). Computerizing Reading Training: Evaluation of a latent semantic analysis space for science text. Behavior Research Methods, Instruments, and Computers, 35, 244-250.
12. Kintsch, E., Caccamise, D., Dooley, S., Franzke, M., & Johnson, N. (2005). Summary street: LSA-based software for comprehension and writing. In T. Landauer, D.S., McNamara, S. Dennis, & W. Kintsch (Eds.), LSA: A Road to Meaning. Mahwah, NJ: Erlbaum.
13. Landauer, T. K., Foltz, P. W., & Laham, D. (1998). Introduction to Latent Semantic Analysis. Discourse Processes, 25, 259-284.
14. Landauer, T. K., McNamara, D. S., Dennis, S., & W. Kintsch. (2005) LSA: A Road to Meaning, Mahwah, NJ: Erlbaum.
15. Louwerse, M. M., Graesser, A. C., Olney, A., & the Tutoring Research Group. (2002). Good computational manners: Mixed-initiative dialog in conversational agents. In C. Miller (Ed.), Etiquette for Human-Computer Work, Papers from the 2002 Fall Symposium, Technical Report FS-02-02, 71-76.
16. Magliano, J. P., Todaro, S., Millis, K. K., Wiemer-Hastings, K., Kim, H. J., & McNamara, D. S. (2004). Changes in reading strategies as a function of reading training: A comparison of live and computerized training. Submitted for publication.
17. McNamara, D. S. (2004). SERT: Self-explanation reading training. Discourse Processes, 38, 1-30.
18. McNamara, D. S., Boonthum, C., Levinstein, I. B., & Millis, K. K. (2005) Using LSA and word-based measures to assess self-explanations in iSTART. In

T. Landauer, D.S. McNamara, S. Dennis, & W. Kintsch (Eds.), LSA: A Road to Meaning, Mahwah, NJ: Erlbaum.
19. McNamara, D. S., Levinstein, I. B., & Boonthum, C. (2004). iSTART: Interactive strategy training for active reading and thinking. Behavior Research Methods, Instruments, & Computers, 36, 222-233.
20. McNamara, D. S., Louwerse, M. M., & Graesser, A. C. (2002). Coh-Metrix: Automated cohesion and coherence scores to predict text readability and facilitate comprehension. Technical report, Institute for Intelligent Systems, University of Memphis, Memphis, TN.
21. McNamara, D. S., & Scott, J. L. (1999). Training reading strategies. In M. Hahn & S. C. Stoness (Eds.), Proceedings of the Twenty-first Annual Meeting of the Cognitive Science Society (pp. 387-392). Hillsdale, NJ: Erlbaum.
22. Millis, K. K., Kim, H. J., Todaro, S. Magliano, J. P., Wiemer-Hastings, K., & McNamara, D. S. (2004). Identifying reading strategies using latent semantic analysis: Comparing semantic benchmarks. Behavior Research Methods, Instruments, & Computers, 36, 213-221.
23. Millis, K. K., Magliano, J. P., Wiemer-Hastings, K., Todaro, S., & McNamara, D. S. (2005). Assessing comprehension with Latent Semantic Analysis. In T. Landauer, D.S. McNamara, S. Dennis, & W. Kintsch (Eds.), LSA: A Road to Meaning, Mahwah, NJ: Erlbaum.
24. O'Reilly, T., Best, R., & McNamara, D. S. (2004). Self-Explanation reading training: Effects for low-knowledge readers. In K. Forbus, D. Gentner, T. Regier (Eds.), Proceedings of the Twenty-sixth Annual Meeting of the Cognitive Science Society (pp. 1053-1058). Mahwah, NJ: Erlbaum.
25. O'Reilly, T., Sinclair, G. P., & McNamara, D. S. (2004). Reading strategy training: Automated verses live. In K. Forbus, D. Gentner, T. Regier (Eds.), Proceedings of the Twenty-sixth Annual Meeting of the Cognitive Science Society (pp. 1059-1064). Mahwah, NJ: Erlbaum.
26. Press, W.M., Flannery, B.P., Teukolsky, S.A., & Vetterling, W.T. (1986). Numerical recipes: The art of scientific computing. New York, NY: Cambridge University Press.
27. Steyvers, M., & Griffiths, T. (2005) Probabilistic topic models. In T. Landauer, D.S. McNamara, S. Dennis, & W. Kintsch (Eds.), LSA: A Road to Meaning, Mahwah, NJ: Erlbaum.
28. Steyvers, M., & Griffiths, T. (2005) Matlab Topic Modeling Toolbox 1.3. Retrieved from http://psiexp.ss.uci.edu/research/programs_data/toolbox.htm
29. Streeter, L., Lochbaum, K., Psotka, J., & LaVoie, N. (2005). Automated tools for collaborative learning environments. In T. Landauer, D.S., McNamara, S. Dennis, & W. Kintsch (Eds.), LSA: A Road to Meaning. Mahwah, NJ: Erlbaum.

CHAPTER 9

INFORMATION RETRIEVAL–1

CHAPTER OVERVIEW

The huge amount of information stored in electronic form, has placed heavy demands on information retrieval systems. This has made information retrieval an important research area. This chapter is concerned with the design of information retrieval systems. It discusses design features of systems and introduces various models, such as the classical (boolean, probabilistic, and vector space retrieval models), non-classical (information logic, situation theory, and interaction information retrieval model), and alternative information retrieval (cluster, fuzzy, and LSI) models. A detailed discussion of vector space model is also given. The final topic of discussion in this chapter is information retrieval evaluation models.

9.1 INTRODUCTION

Information retrieval (IR) deals with the organization, storage, retrieval, and evaluation of information relevant to a user's query. A user in need of information formulates a request in the form of a query written in a natural language. The retrieval system responds by retrieving the document that seems relevant to the query. Research in IR is not new. It dates back to the 1960s when text retrieval systems were introduced. Traditionally, however, it is not considered an important application area of NLP. Interest in the field was generated due to the insurgence of the World Wide Web. The interaction between NLP and IR is now strengthening. Many NLP techniques, including the probabilistic model, have found application in IR systems and techniques such as latent semantic indexing, vector space retrieval, etc.

Traditionally, IR systems are not expected to return the actual information, only documents containing that information. As Lancaster (1979) pointed out:

An information retrieval system does not inform (i.e., change the knowledge of) the user on the subject of her inquiry. It merely informs on the existence (or non-existence) and whereabouts of documents relating to her request.

The word *document* is a general term that includes non-textual information such as images and speech.

Our concern in this chapter is only with retrieval of text documents. This excludes question answering systems and data retrieval systems. The three systems differ in the nature of their queries and expected results of the queries. In both question answering systems and data retrieval systems, queries are very specific and precise in nature. On the other hand, queries submitted to IR systems are often vague and imprecise. A question answering system provides users with the answers to specific questions. Data retrieval systems retrieve precise data, usually organized in a well-defined structure. Unlike data retrieval systems, IR systems do not search for specific data. Nor do they search for direct answers to question, as question answering systems do.

9.2 DESIGN FEATURES OF INFORMATION RETRIEVAL SYSTEMS

Figure 9.1 illustrates the basic process of IR. It begins with the user's information need. Based on this need, he/she formulates a query. The IR system returns documents that seem relevant to the query. This is an engineering account of the IR system. The basic question involved is, 'what constitutes the information in the documents and the queries'. This, in turn is related to the problem of representation of documents and queries. The retrieval is performed by matching the query representation with document representation.

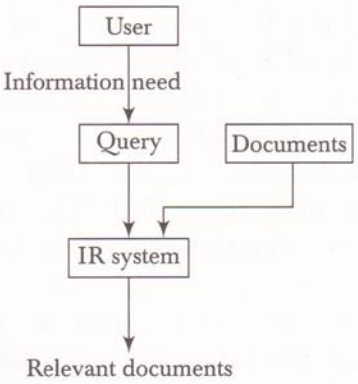


Figure 9.1 Basic information retrieval process

The actual text of the document is not used in the retrieval process. Instead, documents in a collection are frequently represented through a set of index terms or keywords. In this chapter, the word *term* and *keyword* will be used independently. Keywords can be single word or multi-word phrases. They might be extracted automatically or manually (i.e., specified by a human). Such a representation provides a logical view of the document. The process of transforming document text to some representation of it is known as *indexing*. There are different types of index structures. One used data structure, commonly by the IR system, is the inverted index. An inverted index is simply a list of keywords, with each keyword carrying pointers to the documents containing that keyword. The computational cost involved in adopting a full text logical view (i.e., using a full set of words to represent a document) is high. Hence, some text operations are usually performed to reduce the set of representative keywords. The two most commonly used text operations are *stop word elimination* and *stemming*. Stop word elimination removes grammatical or functional words, while stemming reduces words to their common grammatical roots. *Zipf's law* can be applied to further reduce the size of index set. Not all the terms in a document are equally relevant. Some might be more important in conveying a document's content. Attempts have been made to quantify the significance of index terms to a document by assigning them numerical values, called weights. The choice of index terms and weights is a difficult theoretical and practical problem and several technique are used to cope with it. A number of *term-weighting* schemes have been proposed in the literature over the years. We discuss a few of them in this chapter.

9.2.1 Indexing

In a small collection of documents, an IR system can access a document to decide its relevence to a query. However, in a large collection of documents, this technique poses practical problems. Hence, a collection of raw documents is usually transformed into an easily accessible representation. This process is known as indexing. Most indexing techniques involve identifying good document descriptors, such as keywords or terms, which describe the information content of documents. A good descriptor is one that helps describe the content of the document and discriminate it from other documents in the collection. Luhn (1957, 1958) is considered the first person to advance the notion of automatic indexing of documents based on their content. He assumed that the frequency of certain word-occurrences in an article gave meaningful

algorithms has been developed by Porter (1980). The stemmed representation of the text, *Design features of information retrieval systems*, is

{design, featur, inform, retriev, system}

Note that stop words have been removed in this representation and the remaining terms are in lower case.

One of the problems associated with stemming is that it may throw away useful distinctions. In some cases, it may be useful to help conflate similar terms, resulting in increased *recall*. In others, it may be harmful, resulting in reduced *precision* (e.g., when documents containing the term *computation* are returned in response to the query phrase *personal computer*). *Recall* and *precision* are the two most commonly used measures of the effectiveness of an information retrieval system, and are explained in detail later in this chapter.

9.2.4 Zipf's Law

Zipf made an important observation on the distribution of words in natural languages. This observation has been named Zipf's law. Simply stated, Zipf's law says that the frequency of words multiplied by their ranks in a large corpus is more or less constant. More formally,

$$\text{Frequent} \times \text{rank} \approx \text{constant}$$

This means that if we compute the frequencies of the words in a corpus, and arrange them in decreasing order of frequency, then the product of the frequency of a word and its rank is approximately equal to the product of the frequency and rank of another word. This indicates that the frequency of a word is inversely proportional to its rank. This relationship is shown in Figure 9.2.

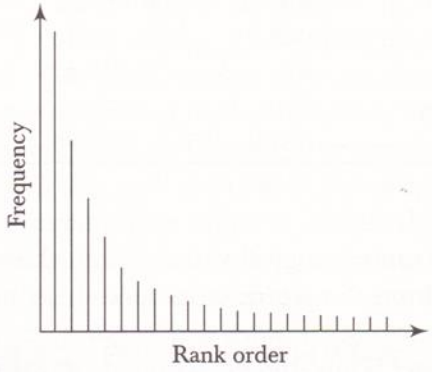


Figure 9.2 Relationship between the frequency of words and their rank order

Empirical investigation of Zipf's law on large corpuses suggest that human languages contain a small number of words that occur with high frequency and a large number of words that occur with low frequency. In between, is a middling number of medium frequency terms. This distribution has important significance in IR. The high frequency words being common, have less discriminating power, and thus, are not useful for indexing. Low frequency words are less likely to be included in the query, and are also not useful for indexing. As there are a large number of rare (low frequency) words, dropping them considerably reduces the size of a list of index terms. The remaining medium frequency words are content-bearing terms and can be used for indexing. This can be implemented by defining thresholds for high and low frequency, and dropping words that have frequencies above or below these thresholds. Stop word elimination can be thought of as an implementation of Zipf's law, where high frequency terms are dropped from a set of index terms.

INFORMATION RETRIEVAL MODELS

An IR model is a pattern that defines several aspects of the retrieval procedure, for example, how documents and user's queries are represented, how a system retrieves relevant documents according to users' queries, and how retrieved documents are ranked. The IR system consists of a model for documents, a model for queries, and a matching function which compares queries to documents. The central objective of the model is to retrieve all documents relevant to a query. This defines the central task of an IR system.

Several different IR models have been developed. These models differ in the way documents and queries are represented and retrieval is performed. Some of them consider documents as sets of terms and perform retrieval based merely on the presence or absence of one or more query terms in the document. Others represent a document as a vector of term weights and perform retrieval based on the numeric score assigned to each document, representing similarity between the query and the document. These models can be classified as follows:

- Classical models of IR
- Non-classical models of IR
- Alternative models of IR

The three classical IR models—Boolean, vector, and probabilistic—are based on mathematical knowledge that is easily recognized and well understood. These models are simple, efficient, and easy to implement.

Almost all existing commercial systems are based on the mathematical models of IR. That is why they are called classical models of IR.

Non-classical models perform retrieval based on principles other than those used by classical models, i.e., similarity, probability, and Boolean operation. These are best exemplified by models based on special logic technique, situation theory, or the concept of interaction.

The third category of IR models, namely alternative models, are actually enhancements of classical models, making use of specific techniques from other fields. The cluster model, fuzzy model, and latent semantic indexing (LSI) model are examples of alternative models of IR.

CLASSICAL INFORMATION RETRIEVAL MODELS

9.4.1 Boolean model

Introduced in the 50s, the Boolean model is the oldest of the three classical models. It is based on Boolean logic and classical set theory. In this model, documents are represented as a set of keywords, usually stored in an inverted file. An inverted file is a list of keywords and identifiers of the documents in which they occur. Users are required to express their queries as a Boolean expression consisting of keywords connected with Boolean logical operators (AND, OR, NOT). Retrieval is performed based on whether or not document contains the query terms.

Given a finite set

$$T = \{t_1, t_2, \ldots, t_i, \ldots, t_m\}$$

of index terms, a finite set

$$D = \{d_1, d_2, \ldots, d_j, \ldots, d_n\}$$

of documents and a Boolean expression—in a normal form—represent a query Q as follows:

$$Q = \wedge(\vee\theta_i),\ \theta_i \in \{t_i, \neg t_i\}$$

The retrieval is performed in two steps:

1. The set R_i of documents are obtained that contain or do not contain the term t_i:

   $$R_i = \{d_j \mid \theta_i \in d_j\},\ \theta_i \in \{t_i, \in t_i\}$$

   where $\neg t_i \in d_j$ means $t_i \notin d_j$
2. Set operations are used to retrieve documents in response to Q:

   $$\cap R_i$$

Example 9.1 Let the set of original documents be

$$D = \{D_1, D_2, D_3\}$$

where

D_1 = Information retrieval is concerned with the organization, storage, retrieval, and evaluation of information relevant to user's query.

D_2 = A user having an information needs to formulate a request in the form of query written in natural language.

D_3 = The retrieval system responds by retrieving the document that seems relevant to the query.

Let the set of terms used to represent these documents be

T = {information, retrieval, query}

Then, the set D of document will be represented as follows:

$D = \{d_1, d_2, d_3\}$

where

d_1 = {information, retrieval, query}

d_2 = {information, query}

d_3 = {retrieval, query}

Let the query Q be

Q = information $\wedge$ retrieval

First, the sets R_1 and R_2 of documents are retrieved in response to Q, where

$R_1 = \{d_j \mid \text{information} \in d_j\} = \{d_1, d_2\}$

$R_2 = \{d_j \mid \text{retrieval} \in d_j\} = \{d_1, d_3\}$

Then, the following documents are retrieved in response to query Q

$\{d_j \mid d_j \in R_1 \cap R_2\} = \{d_1\}$

This results in the retrieval of the original document D_1 that has the representation d_1. If more than one document have the same representation, every such document is retrieved. Boolean information retrieval does not differentiate between these documents. With an inverted index, this simply means taking an intersection of the list of the documents associated with the keywords *information* and *retrieval.*

Boolean retrieval models have been used in IR systems for a long time. They are simple, efficient, and easy to implement and perform well in terms of recall and precision if the query is well formulated. However, the model suffers from certain drawbacks.

First, the model is not able to retrieve documents that are only partly relevant to user query; all information is 'to be or not to be'.

Second, a Boolean system is not able to rank the returned list of documents. It distinguishes between presence and absence of keywords but fails to assign relevance and importance to keywords in a document.

Third, users seldom formulate their query in the pure Boolean expression that this model requires.

Numerous extensions of the Boolean model have been suggested to overcome these weaknesses. The best of these are the P-norm model developed by Salton et al. (1983) and a fuzzy-set model suggested by Paice (1984).

9.4.2 Probabilistic Model

The probabilistic model applies a probabilistic framework to IR. It ranks documents based on the probability of their relevance to a given query (Robertson and Jones 1976). Retrieval depends on whether probability of relevance (relative to a query) of a document is higher than that of non-relevance, i.e. whether it exceeds a threshold value. Given a set of documents D, a query q, and a cut-off value α, this model first calculates the probability of relevance and irrelevance of a document to the query. It then ranks documents having probabilities of relevance at least that of irrelevance in decreasing order of their relevance. Documents are retrieved if the probability of relevance in the ranked list exceeds the cut off value.

More formally, if $P(R/d)$ is the probability of relevance of a document d, for query q, and $P(I/d)$ is the probability of irrelevance, then the set of documents retrieved in response to the query q is as follows.

$$S = \{d_j \mid P(R/d_j) \geq P(I/d_j)\}\ P(R/d_j) \geq \alpha$$

Development of the probabilistic model was carried out largely by Maron and Kuhns (1960), Robertson and Sparck-Jones (1976), Robertson et al. (1982). Different mathematical methods for calculating the probabilities of relevance and irrelevance, as well as properties and applications, are discussed by Van Rijsbergen (1992), Callan et al. (1992), and Fur (1992).

Most of the systems assume that terms are independent when estimating probabilities for the probabilistic model. This assumption allows for accurate estimation of parameter values and helps reduce computational complexity of the model. However, this assumption seems to be inaccurate, as terms in a given domain usually tend to co-occur. For example, it is more likely that 'match point' will co-occur with 'tennis' rather than 'cricket'. Different forms of assumption are discussed in Robertson (1977). A comparison of the Boolean and probabilistic models can be found in Losee (1997). A comprehensive review of the probabilistic model is presented by Crestani et al. (1998). A baysian network version of the probabilistic model forms the basis of the INQUERY system (Callan et al. 1992).

The probabilistic model, like the vector model, can produce results that partly match the user query. Nevertheless, this model has drawbacks, one of which is the determination of a threshold value for the initially retrieved set; the number of relevant documents by a query is usually too small for the probability to be estimated accurately.

9.4.3 Vector Space Model

The vector space model is one of the most well-studied retrieval models. Important contribution to its development was made by Luhn (1959), Salton (1968), Salton and McGill (1983), and van Rijsbergen (1977). The vector space model represents documents and queries as vectors of features representing terms that occur within them. Each document is characterized by a Boolean or numerical vector. These vectors are represented in a multi-dimensional space, in which each dimension corresponds to a distinct term in the corpus of documents. In its simplest form, each feature takes a value of either zero or one, indicating the absence or presence of that term in a document or query. More generally, features are assigned numerical values that are usually a function of the frequency of terms. Ranking algorithms compute the similarity between document and query vectors, to yield a retrieval score to each document. This score is used to produce a ranked list of retrieved documents. Given a finite set of n documents

$$D = \{d_1, d_2, \ldots, d_j, \ldots, d_n\}$$

and a finite set of m terms

$$T = \{t_1, t_2, \ldots, t_i, \ldots, t_m\}$$

each document is represented by a column vector of weights as follows:

$$(w_{1j}, w_{2j}, w_{3j}, \ldots, w_{ij}, \ldots, w_{mj})^t$$

where w_{ij} is the weight of the term t_i in document d_j. The document collection as a whole is represented by an $m \times n$ term-document matrix as

$$\begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1j} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2j} & \cdots & w_{2n} \\ w_{i1} & w_{i2} & \cdots & w_{ij} & \cdots & w_{in} \\ w_{m1} & w_{m2} & \cdots & w_{mj} & \cdots & w_{mn} \end{pmatrix}$$

Different term-weighting functions have been introduced (Salton and Buckley 1988). We discuss some of them in this section.

Example 9.2 Consider the documents and terms in Example 9.1. Let the weights be assigned based on the frequency of the term within the document. Then, the associated vectors will be

(2, 2, 1)
(1, 0, 1)
(0, 1, 1)

The vectors can be represented as a point in Euclidean space, where the coordinates of point P_j are the components of d_j. The term-document matrix is

$$\begin{pmatrix} 2 & 1 & 0 \\ 2 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

This raw term frequency approach gives too much importance to the absolute values of various coordinates of each document. For example, a document with weights 4, 4, and 2, would be quite similar to document 1, except for the differences in magnitude of term weights.

To reduce the importance of the length of document vectors, we normalize document vectors. Normalization changes all vectors to a standard length. We convert document vectors to unit length by dividing each dimension by the overall length of the vector. Normalizing the term-document matrix shown in this example, we get the following matrix:

$$\begin{pmatrix} 0.67 & 0.71 & 0 \\ 0.67 & 0 & 0.71 \\ 0.33 & 0.71 & 0.71 \end{pmatrix}$$

Elements of each column are divided by the length of the column vector given by $\sqrt{\sum_i w_{ij}^2}$. The values shown in this matrix have been rounded to two decimal digits.

9.4.4 Term Weighting

Each term that is selected as an indexing feature for a document, acts as a discriminator between that document and all other documents in the corpus. Luhn (1958) attempted to quantify the discriminating power of the terms by associating the frequency of their occurrence (term frequency) within the document. He postulated that the most discriminating (content bearing) terms are mid frequency terms. This postulate can be refined by noting the following facts:

1. The more a document contains a given word, the more that document is about a concept represented by that word.
2. The less a term occurs in particular document in a collection, the more discriminating that term is.

The first factor simply means that terms that occur more frequently represent the document's meaning more strongly than those occurring less frequently, and hence should be given high weights. In the simplest form, this weight is the raw frequency of the term in the document, as discussed earlier. The second factor actually considers term distribution across the document collection. Terms occurring in a few documents are useful for distinguishing those documents from the rest of the collection. Similarly, terms that occur more frequently across the entire collection are less helpful while discriminating among documents. This requires a measure that favours terms appearing in fewer documents. The fraction n/n_i—where n is the total number of the document in the collection and n_i the number of the document in which the term i occurs—exactly gives this measure. This measure assigns the lowest weight 1 to a term that appears in all documents and the highest weight of n to a term that occurs in only one document. As the number of documents in any collection is usually large, the log of this measure is usually taken, resulting in the following form of inverse document frequency (idf) term weight:

$$\text{idfi} = \log\left(\frac{n}{n_i}\right)$$

Inverse document frequency (idf) attaches more importance to more specific terms. If a term occurs in all documents in a collection, its idf is 0. Researchers have attempted to include term distribution in the weighting function (Sparck-Jones 1972, Salton 1971) to give a more accurate quantification of term importance. Sparck-Jones showed experimentally that a weight of $\log(n/n_i)+1$, termed as inverse document frequency, leads to more effective retrieval. Later researchers attempted to combine term frequency (tf) and idf weights, resulting in a family of tf × idf weight schemes having the following general form:

$$w_{ij} = \text{tf}_{ij} \times \log\left(\frac{n}{n_i}\right)$$

According to this scheme, the weight of a term t_i in document d_j is equal to the product of document frequency of the term and log of its inverse document frequency within the collection. The tf × idf weighting scheme combines both the 'local' and 'global' statistics to assign term weight. The tf component is a document specific statistic that measures

the importance of a term within the document, whereas the idf is a global statistic that attempts to include distribution of the term across the document collection. These weighting schemes are well suited to the ad-hoc retrieval environment but pose problems in routing environment, where no fixed document collection exists. Usually, a training set of documents is used to compute statistics in such an environment and it is assumed that subsequent documents arriving at the system have the same statistical properties as the training set. We now give an example to explain how term weights can be calculated using the tf–idf weighting scheme.

Example 9.3 Consider a document represented by the three terms {tornado, swirl, wind} with the raw tf 4, 1, and 1 respectively. In a collection of 100 documents, 15 documents contain the term *tornado*, 20 contain *swirl*, and 40 contain *wind*. The idf of the term *tornado* can be computed as

$$\log\left(\frac{n}{n_i}\right) = \log\left(\frac{100}{15}\right) = 0.82$$

The idf of other terms are computed in the same way. Table 9.2 shows the weights assigned to the three terms using this approach.

Table 9.2 Computing tf-idf weight

Term	Frequency (tf)	Document frequency (n_i)	idf [$\log(n/n_i)$]	Weight (tf × idf)
Tornado	4	15	0.824	0.296
Swirl	1	20	0.699	0.699
Wind	1	40	0.398	0.389

Many variations of tf × idf measure have been reported. Some of these attempt to normalize tf and idf factors in different ways to allow for variations in document length (Salton and Buckley 1988). One way to normalize tf is to divide it by the frequency of the most frequent term in the document. This kind of normalization, often termed as maximum normalization, yields a value between 0 and 1. Normalization is needed because using absolute (raw) term frequency to weight terms favours longer documents over shorter ones. After frequency normalization, the weight of a term in a given document depends on the frequency of its occurrence in relation to the other terms in the same document, instead of its absolute frequency. Similarly, idf can be normalized by dividing it by the logrithm of the collection size (n).

$$w_{ij} = \frac{\text{tf}_{ij}}{\max(\text{tf}_{ij})} \times \log\left(\frac{n}{n_i}\right) / \log n$$

A third factor that may affect weighting function is the document length. A term appearing the same number of times in a short document and in a long document, will be more valuable to the former. Most weighting schemes can thus be characterized by the following three factors:

- Within-document frequency or term frequency (tf)
- Collection frequency or inverse document frequency (idf)
- Document length

Any term weighting scheme can be represented by a triple ABC. The letter A in this triple represents the way the tf component is handled, *B* indicates the way the idf component is incorporated, and *C* represents the length normalization component. Possible options for each of the three dimensions of the triple are shown in Table 9.3. Different combinations of options can be used to represent document and query vectors. The retrieval model themselves can be represented by a pair of triples like nnn.nnn (doc = 'nnn', query = 'nnn'), where the first triple corresponds to the weighting strategy used for the documents and the second triple to the weighting strategy used for the query term.

Table 9.3 Calculating weight with different options for the three weighting factors

	Term frequency within document		
A	*n*	$tf = tf_{ij}$	Raw term frequency
	b	tf = 0 or 1 (binary weight)	
	a	$tf = 0.5 + 0.5\left(\frac{tf_{ij}}{\max tf \text{ in } D_j}\right)$	Augmented term frequency
	l	$tf = \ln(tf_{ij}) + 1.0$	Logarithmic term frequency
	L	$tf = \frac{\ln(tf_{ij} + 1.0)}{1.0 + \ln[\text{mean}(tf \text{ in } D_j)]}$	Average term frequency-based normalization
	Inverse document frequency		
B	*n*	wt = tf	No conversion
	t	$wt = tf \cdot \ln\left(\frac{n}{n_i}\right)$	Multiply tf with idf
	Document length		
C	*n*	w_{ij} = wt	(no conversion)
	c	w_{ij} is obtained by dividing each wt by sqrt [sum of (wts squared)]	

There are many ways to compute each component. The simplest is to use either binary weight or raw term frequency. As shown in Table 9.3, the first occurrence of a term is more important than successive repeating occurrences. Thus, tf can be computed as 0.5 + 0.5 (tf_{ij}/max tf in d_j) in which normalization is achieved by dividing tf by maximum tf value for any term in the document, or as $\ln(tf_{ij}) + 1.0$, which is known as logarithmic term frequency. The former computation is called augmented normalized term frequency. It causes tf to vary between 0.5 and 1. The problem with maximum normalization and augmented normalization of the tf component is that a single term in a document, with an unusually high frequency, may degrade the weights of the other terms significantly. However, this effect is not too pronounced with the augmented tf, because the highest frequency term cannot degrade the frequency of other terms below 0.5. The logarithmic term frequency reduces the effect of unusually frequent terms within a document, and also the importance of raw term frequency in a collection of documents with significant variations in length. It actually decreases the effect of all sorts of variations in tf, because for any two term frequencies tf_1 and $tf_2 > 0$, such that $tf_1 > tf_2$, the ratio of the logarithmic term frequencies will be always less than the ratio of the raw term frequencies, i.e.,

$$\frac{\log(tf_1)+1}{\log(tf_2)+1} < \frac{tf_1}{tf_2}$$

Different choices for *A*, *B*, and *C* for query and document vectors yield different retrieval modes, for example, ntc-ntc, lnc-ltc, etc. The choices for tf are *n* (use the raw term frequency), *b* (binary, i.e., neglect term frequency, term frequency will be 1 if term is present in the document, otherwise 0), *a* (augmented normalized frequency), *l* (logarithmic term frequency), and *L* (logarithmic frequency normalized by average term frequency). The options for idf are *n* (use 1.0, ignore idf factor) and *t* (use idf). The possible options listed in Table 9.3 for document length normalization are *n* (no normalization) and *c* (cosine normalization). To achieve cosine normalization, every element of the term weight vector is divided by the Euclidean length of the vector. This is called cosine normalization because the length of the normalized vector is 1 and its projection on any axis in document space gives the cosine of the angle between the vector and the axis under consideration.

The widely known weighting scheme, ntc-ntc, normalizes both the document and query term weight in the range 0–1 and may prove

beneficial. The weighting scheme lnc-ltc means that document term weights are computed as the product of the logarithmic tf (*l*) of the given term, 1.0 (*n*) and cosine normalization (*c*) of the document vector. The query term weights are computed in the same way, except that each query term weight is also multiplied by the idf (t) of the given term in the document collection.

More recent weighting schemes integrate document length within the weighting formula yielding more complex retrieval models, for example, Okapi probabilistic search model (Robertson et al. 1995) and doc= "Lnu" model (Buckley et al. 1996). In TREC-3, the three systems with the best base performance were Okapi, INQUERY, and Cornell's Smart. The best performance was reported by Okapi system. Okapi uses the BM25 weighting algorithm introduced by developers of the probabilistic model during TREC-2 (Robertson et al. 1994) and TREC-3 (Robertson et al. 1995). Robertson and Walker (1994) developed the best match (BM) algorithms using the probabilistic model and some simple approximations to two-poisson model.

Considerable research efforts have been devoted to refining term weighting methods. As a result, a large number of term weighting schemes have been proposed in IR literature. (Salton and McGill 1983, Rijsbergen 1979). Some recent weighting schemes also consider the structure of the document.

A simple automatic method for obtaining indexed representation of the documents is as follows.

Step 1 Tokenization This extracts individual terms form a document, converts all the letters to lower case, and removes punctuation marks. The output of the first stage is a representation of the document as a stream of terms.

Step 2 Stop word elimination This removes words that appear more frequently in the document collection.

Step 3 Stemming This reduces the remaining terms to their linguistic root, to obtain the index terms.

Step 4 Term weighting This assigns weights to terms according to their importance in the document, in the collection, or some combination of both.

Table 9.4 shows the document vectors obtained after the application of these steps on sample documents shown in Figure 9.3.

Document 1:	Vector space model
Document 2:	Probabilistic retrieval model
Document 3:	Intelligent techniques in information retrieval

Figure 9.3 Sample documents

Table 9.4 Vector representation of sample documents after stemming

Stemmed terms	Document 1	Document 2	Document 3
inform	0	0	1
intellig	0	0	1
model	1	1	0
probabilist	0	1	0
retriev	0	1	1
space	1	0	0
technique	0	0	1
vector	1	0	0

9.4.5 Similarity Measures

Vector space model represents documents and queries as vectors in a multi-dimensional space. Retrieval is performed by measuring the 'closeness' of the query vector to document vector. Documents can then be ranked according to the numeric similarity between the query and the document. In the vector space model, the documents selected are those that are geometrically closest to the query according to some measure. The model relies on the intuitive notion that similar vectors define semantically related documents. Figure 9.4 gives an example of document and query representation in two-dimensional vector space. These dimensions correspond to the two index terms t_i and t_j. Document d_1 has two occurrences of t_i, document d_2 has one occurrence of t_i, and document d_3 has one occurrence of t_i and t_j each. Documents d_1, d_2, and d_3 are represented in this space using term weights—raw term frequency being used here—as coordinates. The angles between the documents and query are represented as θ_1, θ_2, and θ_3 respectively.

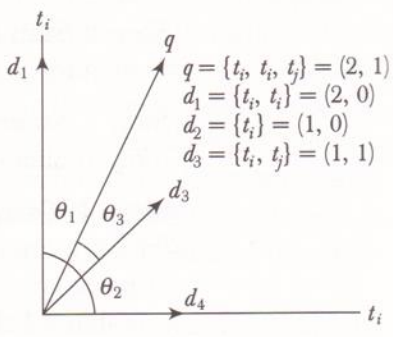


Figure 9.4 Representation in two-dimensional vector space

The simplest way of comparing document and query is by counting the number of terms they have in common. One frequently used similarity measure

is to take the 'inner product' between the query and the document vector. The inner product is given by

$$\text{sim}\ (d_j,\ q_k) = (d_j,\ q_k) = \sum_{i=1}^{m} w_{ij} \times w_{ik}$$

where m is the number of terms used to represent documents in the collection.

Other measures, e.g., dice coefficient, Jaccard's coefficient, and cosine coefficient, attempt to normalize the similarity by the length of document and query. The dice coefficient defines the similarity between query k and document j as

$$\text{sim}\ (d_j,\ q_k) = \frac{2 \times \left(\sum_{i=1}^{m} w_{ij} \times w_{ik} \right)}{\sum_{i=1}^{m} w_{ij}^2 + \sum_{i=1}^{m} w_{ik}^2}$$

Jaccard's coefficient is defined as

$$\text{sim}\ (d_j,\ q_k) = \frac{\sum_{i=1}^{m} w_{ij} \times w_{ik}}{\sum_{i=1}^{m} w_{ij}^2 + \sum_{i=1}^{m} w_{ik}^2 - \sum_{i=1}^{m} w_{ij} \times w_{ik}}$$

The cosine measure is commonly used for measuring similarity in IR. It computes cosine of the angle between the document and query vector, to give a similarity value between 0 and 1. A minimum value of 0 (angle 90°) indicates that the vectors are unrelated (i.e., they have no terms in common), and a value of 1 means that the vectors share common terms. If d_j and q_k are the document and query vector respectively, then the cosine similarity is computed as

$$\text{sim}\ (d_j,\ q_k) = \frac{(d_j, q_k)}{\|d_j\| \|q_k\|} = \frac{\sum_{i=1}^{m} w_{ij} \times w_{ik}}{\sqrt{\sum_{i=1}^{m} w_{ik}^2} \times \sqrt{\sum_{i=1}^{m} w_{ij}^2}}$$

If both the document and the query vectors have been cosine-normalized, then the inner product yields the cosine similarity.

The cosine measure divides the numerator by the product of the length of vectors. This tends to give low similarities to long vectors, i.e. vectors with many terms. The overlap coefficient compensates for this by dividing

by the vector having smaller sum of weights. More formally, this measure is defined as

$$\text{sim}\ (d_j,\ q_k) = \frac{\sum_{i=1}^{m} w_{ij} \times w_{ik}}{\min\left(\sum_{i=1}^{m} w_{ij}, \sum_{i=1}^{m} w_{ik}\right)}$$

9.5 NON-CLASSICAL MODELS OF IR

Non-classical IR models are based on principles other than similarity, probability, Boolean operations, etc., on which classical retrieval models are based. Examples include information logic model, situation theory model, and interaction model.

The *information logic model* is based on a special logic technique called logical imaging. Retrieval is performed by making inferences from document to query. This is unlike classical models, where a search process is used. Unlike usual implication, which is true in all cases except that when antecedent is true and consequent is false, this inference is uncertain. Hence, a measure of uncertainty is associated with this inference. The principle put forward by van Rijsbergen is used to measure this uncertainty. This principle says:

Given any two sentences x and y, a measure of the uncertainty of $y \rightarrow x$ *relative to a given data set is determined by the minimal extent to which one has to add information to the data set in order to establish the truth of* $y \rightarrow x$.

In fact, this model was developed in response to van Rijsbergen's realization that classical models were unable to enhance effectiveness and that new meaning-based models were required to do so.

The *situation theory* model is also based on van Rijsbergen's principle. Retrieval is considered as a flow of information from document to query. A structure called *infon,* denoted by ι, is used to describe the situation and to model information flow. An infon represents an n-ary relation and its polarity. The polarity of an infon can be either 1 or 0, indicating that the infon carries either positive or negative information.

For example, the information in the sentence, *Adil is serving a dish,* is conveyed by the infon

$$\iota = \langle\langle \textit{serving Adil, dish}; 1\rangle\rangle$$

The polarity of an infon depends on the support. The support of an infon is represented as $s \models \iota$ and means that the situation s makes the infon ι true. For example, the infon, $\iota = \langle\langle$*serving Adil, dish*; 1$\rangle\rangle$ is made true by a situation $s1$ = "I see Adil serving a dish."

A document d is relevant to a query q, if $d \models q$

If a document does not support the query q, it does not necessarily mean that the document is not relevant to the query. Additional information, such as synonymys, hypernyms/hyponyms, meronyms, etc., can be used to transform the document d into d' such that $d' \models q$. Semantic relationships in a thesaurus, like WordNet, are useful sources for this information. The transformation from d to d' is regarded as flow of information between situations.

The *interaction IR* model was first introduced in Dominich (1992, 1993) and Rijsbergen (1996). In this model, the documents are not isolated; instead, they are interconnected. The query interacts with the interconnected documents. Retrieval is conceived as a result of this interaction. This view of interaction is taken from the concept of interaction as realized in the Copenhagen interpretation of quantum mechanics. Artificial neural networks can be used to implement this model. Each document is modelled as a neuron, the document set as a whole forms a neural network. The query is also modelled as a neuron and integrated into the network. Because of this integration, new connections are built between the query and the documents, and existing connections are changed. This restructuring corresponds to the concept of interaction. A measure of this interaction is obtained and used for retrieval. Detailed mathematical treatments of the model have been discussed by Dominich (1992, 1993, 2001) and van Rijsbergen (1996).

ALTERNATIVE MODELS OF IR

9.6.1 Cluster Model

The cluster model is an attempt to reduce the number of matches during retrieval. The need for clustering was first pointed out by Salton. Before we discuss the cluster-based IR model, we would like to state the cluster hypothesis that explains why clustering could prove efficient in IR.

Closely associated documents tend to be relevant to the same clusters.

This hypothesis suggests that closely associated documents are likely to be retrieved together. This means that by forming groups (classes or clusters) of related documents, the search time reduced considerably. Instead of matching the query with every document in the collection, it is matched with representatives of the class, and only documents from a class whose representative is close to query, are considered for individual match.

Clustering can be applied on terms instead of documents. Thus, terms can be grouped to form classes of co-occurrence terms. Co-occurrence terms can be used in dimensionality reduction or thesaurus construction. A number of methods are used to group documents. We discuss here, a cluster generation method based on similarity matrix. This method works as follows:

Let $D = \{d_1, d_2, \ldots, d_j, \ldots, d_m\}$ be a finite set of documents, and let $E = (e_{ij})_{n,n}$ be the similarity matrix. The element E_{ij} in this matrix, denotes a similarity between document d_i and d_j. Let T be the threshold value. Any pair of documents d_i and d_j $(i \neq j)$ whose similarity measure exceeds the threshold $(e_{ij} \geq T)$ is grouped to form a cluster. The remaining documents form a single cluster. The set of clusters thus obtained is

$$C = \{C_1, C_2, \ldots, C_k, \ldots, C_p\}$$

A representative vector of each class (cluster) is constructed by computing the centroid of the document vectors belonging to that class. Representation vector for a cluster C_k is

$$r_k = \{a_{1k}, a_{2k}, \ldots, a_{ik}, \ldots, a_{mk}\}$$

An element a_{ik} in this vector is computed as

$$a_{ik} = \frac{\sum_{d_j \in C_k} a_{ij}}{|C_k|}$$

where a_{ij} is weight of the term t_i, of the document d_j, in cluster C_k. During retrieval, the query is compared with the cluster vectors

$$(r_1, r_2, \ldots, r_k, \ldots, r_p)$$

This comparison is carried out by computing the similarity between the query vector q and the representative vector r_k as

$$s_{ik} = \sum_{i=1}^{m} a_{ik} q_i, \quad k = 1, 2, \ldots, p$$

A cluster C_k whose similarity s_k exceeds a threshold is returned and the search proceeds in that cluster.

Example 9.4

Let

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

be the term-by-document matrix. The similarity matrix corresponding to these documents is

1.0
0.9 1.0
0.4 0.4 1.0

Using a threshold of 0.7, we get the following two clusters:

$C_1 = \{d_1, d_2\}$
$C_2 = \{d_3\}$

The cluster vectors (representatives) for C_1 and C_2 are

$r_1 = (1\ 0.5\ 1\ 0\ 1)$
$r_2 = (0\ 0\ \ 1\ 1\ 0)$

Retrieval is performed by matching the query vector with r_1 and r_2.

9.6.2 Fuzzy Model

In the fuzzy model, the document is represented as a fuzzy set of terms, i.e., a set of pairs $[t_i, \mu(t_i)]$, where μ is the membership function. The membership function assigns to each term of the document a numeric membership degree. The membership degree expresses the significance of term to the information contained in the document. Usually, the significance values (weights) are assigned based on the number of occurrences of the term in the document and in the entire document collection, as discussed earlier. Each document in the collection

$$D = \{d_1, d_2, \ldots, d_j, \ldots, d_n\}$$

can thus be represented as a vector of term weights, as in the following vector space model

$$(w_{1j}, w_{2j}, w_{3j}, \ldots, w_{ij}, \ldots, w_{mj})^t$$

where w_{ij} is the degree to which term t_i belongs to document d_j.

Each term in the document is considered a representative of a subject area and w_{ij} is the membership function of document d_j to the subject area represented by term t_i. Each term t_i is itself represented by a fuzzy set f_i in the domain of documents given by

$$f_i = \{(d_j, w_{ij})\} \mid i = 1, \ldots, m; j = 1, \ldots, n$$

This weighted representation makes it possible to rank the retrieved documents in decreasing order of their relevance to the user's query.

Typically, queries are Boolean queries. For each term that appears in the query, a set of documents is retrieved. Fuzzy set operators are then applied to obtain the desired result.

For a single-term query $q = t_q$, those documents from the fuzzy set $f_q = \{(d_j, w_{iq})\}$, are retrieved for which w_{iq} exceeds a given threshold. The threshold may also be zero.

Consider the case of an AND query $q = t_{q1} \wedge t_{q2}$

First, the fuzzy sets f_{q1} and f_{q2} are obtained and then, their intersection is obtained, using the fuzzy intersection operator $f_{q1} \vee f_{q2} = \min \{(d_j, w_{iq1}), (d_j, w_{iq})\}$

The documents in this set are returned.

Similarly, for an OR query $q = t_{q1} \wedge t_{q2}$, the union of fuzzy sets f_{q1} and f_{q2} is computed to retrieve documents as follows:

$$f_{q1} \vee f_{q2} = \max \{(d_j, w_{iq1}), (d_j, w_{iq})\}$$

Example 9.5 Consider the following three documents:

d_1 = {information, retrieval, query}
d_2 = {retrieval, query, model}
d_3 = {information, retrieval}

where the set of terms used to represent documents is

T = {information, model, query, retrieval}

The fuzzy sets induced by these terms are

$f_1 = \{(d_1, 1/3), (d_2, 0) (d_3, 1/2)\}$
$f_2 = \{(d_1, 0), (d_2, 1/3) (d_3, 0)\}$
$f_3 = \{(d_1, 1/3), (d_2, 1/3) (d_3, 0)\}$
$f_4 = \{(d_1, 1/3), (d_2, 1/3) (d_3, 1/2)\}$

If the query is $q = t_2 \wedge t_4$, then document d_2 will be returned.

9.6.3 Latent Semantic Indexing Model

Latent semantic indexing model is the application of single value decomposition to IR. The use of latent semantic indexing (LSI) is based on the assumption that there is some underlying 'hidden' semantic structure in the pattern of word-usage across documents, rather than just surface level word choice. LSI attempts to identify this hidden semantic structure through statistical techniques and use it to represent and retrieve information. This is done by modelling the association between terms and documents based on the manner in which terms co-occur across documents. LSI transforms the term-document vector space into a more compact latent semantic space. Each dimension in the reduced space corresponds to an 'artificial concept'. These concepts loosely correspond to a set of terms. It is believed that in the vector space of reduced dimensionality, the words referring to related concepts, i.e., words that

co-occur, are collapsed into the same dimension. Latent semantic space is thus able to capture similarities that go beyond term similarity. In the latent semantic space, a query and a document can have high similarity even if the document does not contain a query term, provided the terms are semantically related.

Now we discuss how the LSI technique is actually employed in IR. The document collection is first processed to get a $m \times n$ term-by-document matrix, W, where m is the number of index terms and n is the total number of documents in the collection. Columns in this matrix represent document vectors, whereas the rows denote term vectors. The matrix element W_{ij} represents the weight of the term i in document j. The weight may be assigned based on term frequency or some combination of local and global weighting, as in the case of vector space model. Singular value decomposition (SVD) of the term-by-document matrix is then computed. Using SVD, the matrix is represented as a product of three matrices

$$W = TSD^T$$

where T corresponds to term vectors and has m rows and r columns and $r = \min(m, n)$. S corresponds to singular values. D^T is the transpose of D and has r rows and n columns. D corresponds to the document vector.

T and D are orthogonal matrices containing the left and right singular vectors of W. S is a diagonal matrix, containing singular values stored in decreasing order. We eliminate small singular values and approximate the original term-by-document matrix using truncated SVD. For example, by considering only the first k number of the largest singular values, along with their corresponding columns in T and D, we get the following approximation of the original term-by-document matrix in a space of k orthogonal dimensions, where k is sufficiently less than n:

$$W_k = T_k S_k D_k^T$$

where T_k is the first k columns of T, D_k^T is the first k columns of D^T, and S_k is the k largest singular values.

The matrix W_k is used for retrieval. The idea is that the elimination of small singular values throws out the 'noise' resulting from term usage variation, and captures the underlying 'hidden' semantic structure (i.e., concepts). Each dimension in the reduced space corresponds to artificial or derived concepts. Each such concept loosely represents a set of terms in the original term-document matrix. Documents with varying word usage patterns are collapsed to the same vector in k-space.

The queries are also represented in k-dimensional space. Let $q = (q_1, q_2, \ldots, q_m)$ be the original query vector, where each element q_i is the frequency

of term i in the query q. The query q is represented in the k-dimensional space as

$$q_k = q^T T_k S_k^{-1}$$

where q^T is the transpose of the query vector, and T_k and S_k are the weights. $q^T T_k$ denotes the sum of k-dimensional term vectors and S_k^{-1}, the weights of each dimension. Thus, the query is represented as the weighted sum of its constituent term vectors.

Retrieval is performed by computing the similarity between query vector and document vector. For example, we can use the cosine similarity measure to rank documents to perform retrieval. In a keyword-based retrieval, relevant documents that do not share any term with the query are not retrieved. The LSI-based approach is capable of retrieving such documents, as similarity is computed based on the overall pattern of term usage across the document collection rather than on term overlap.

We now give an example to explain how a document in high-dimensional space is represented in a low, reduced, latent semantic space.

Example 9.6 Consider the matrix shown in Figure 9.5. This matrix defines five-dimensional space in which six documents, d_1, d_2, d_3, ..., d_6, have been represented. The five dimensions correspond to five index terms *tornado, storm, tree, forest,* and *farming.* For simplicity, tf has been used to weight index terms. Figure 9.6 shows the documents in a two-dimensional space. The vectors in the figure correspond to document vectors in the matrix R, which is the representation of X in reduced two-dimensional space. The two dimensions correspond to derived concepts obtained through the application of truncated SVD.

$X =$

	d_1	d_2	d_3	d_4	d_5	d_6
tornado	1	1	0	0	0	0
storm	1	0	1	0	1	0
tree	1	0	1	0	0	0
forest	0	0	1	1	0	0
farming	0	0	0	1	0	1

Figure 9.5 A term-document matrix representing six documents in five-dimensional space

We now explain how to arrive at the reduced dimensionality representation of X. First, the SVD of X is computed to get the three matrices T, S, and D.

$$X_{5\times 6} = T_{5\times 5}\ S_{5\times 5}\ (D_{6\times 5})^T$$

These matrices are shown in Figures 9.7, 9.8, and 9.9 respectively. Consider the first two largest singular values of S, and rescale $D^T_{2\times 6}$ with singular

values to get matrix $R_{2\times6} = S_{2\times2}D^T_{2\times6}$, as shown in Figure 9.10, where $S_{2\times2}$ is S restricted to two dimensions and $D^T_{2\times6}$ is D^T restricted to two columns. R is a reduced dimensionality representation of the original term-by-document matrix X and is used to plot the vectors in Figure 9.6.

To find out the changes introduced by the reduction, we compute document similarities in the new space and compare them with the similarities between documents in the original space. The document–document correlation matrix for the original n-dimensional space is given by the matrix $Y = X^TX$. Here, Y is a square, symmetric $n \times n$ matrix. An element Y_{ij} in this matrix gives the similarity between documents i and j. The correlation matrix for the original document vectors is shown in Figure 9.12. This matrix is computed using X, after normalizing the lengths of its columns. The document–document correlation matrix for the new space is computed analogously using the reduced representation R. Let N be the matrix R with length-normalized columns. Then, $M = N^TN$ gives the matrix of document correlations in the reduced space. The correlation matrix M is given in Figure 9.11. The similarity between document d_1, d_4(–0.0304), and d_6(–0.2322) is quite low in the new space because document d_1 is not topically similar to documents d_4 and d_6. In the original space, the similarity between documents d_2 and d_3 and between documents d_2 and d_5 is 0. In the new space, they have high similarity values (0.5557 and 0.8518 respectively) although documents d_3 and d_5 share no term with the document d_2. This topical similarity is recognized due to the co-occurrence of patterns in the documents.

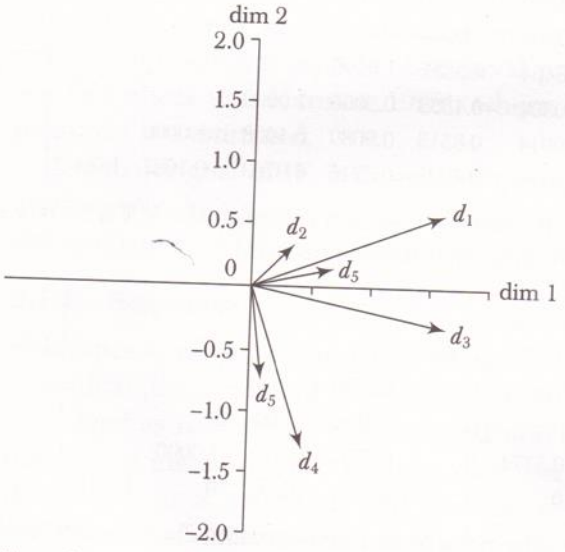


Figure 9.6 Documents in reduced two-dimensional space

$$T = \begin{pmatrix} 0.3318 & 0.3338 & 0.8064 & -0.2426 & -0.2634 \\ 0.6693 & 0.1616 & -0.2737 & 0.5853 & -0.3293 \\ 0.5514 & 0.1038 & -0.0961 & -0.2667 & 0.7777 \\ 0.3583 & -0.5745 & -0.2148 & -0.5778 & -0.4021 \\ 0.0974 & -0.7223 & 0.4684 & 0.4400 & 0.2362 \end{pmatrix}$$

Figure 9.7 Matrix T for the SVD of the term-document matrix X shown in Figure 9.5

$$S = \begin{pmatrix} 2.3830 & 0 & 0 & 0 & 0 \\ 0 & 1.6719 & 0 & 0 & 0 \\ 0 & 0 & 1.2415 & 0 & 0 \\ 0 & 0 & 0 & 0.8288 & 0 \\ 0 & 0 & 0 & 0 & 0.5454 \end{pmatrix}$$

Figure 9.8 The matrix S for singular values of the SVD of the term-document matrix X

$$D^T = \begin{pmatrix} 0.6515 & 0.1392 & 0.6626 & 0.1912 & 0.2809 & 0.0409 \\ 0.3584 & 0.1996 & -0.1848 & -0.7756 & 0.0967 & -0.4320 \\ 0.3516 & 0.6495 & -0.4710 & 0.2042 & -0.2205 & 0.3773 \\ 0.0916 & -0.2927 & -0.3127 & -0.1662 & 0.7062 & 0.5309 \\ 0.3392 & -0.4829 & 0.0849 & -0.3042 & -0.6037 & 0.4330 \end{pmatrix}$$

Figure 9.9 The matrix D^T for singular values of the SVD of the term-document matrix

$$R = \begin{pmatrix} 1.5526 & 0.3318 & 1.5790 & 0.4557 & 0.6693 & 0.0974 \\ 0.5992 & 0.3338 & -0.3090 & -1.2967 & 0.1616 & -0.7223 \end{pmatrix}$$

Figure 9.10 The matrix $R_{2\times6} = S_{2\times2}D^T_{2\times6}$ representing documents in two-dimensional space

$$M = \begin{pmatrix} 1.0000 & & & & & \\ 0.9131 & 1.0000 & & & & \\ 0.8464 & 0.5557 & 1.0000 & & & \\ -0.0304 & -0.4353 & 0.5066 & 1.0000 & & \\ 0.9914 & 0.8518 & 0.9089 & 0.1008 & 1.0000 & \\ -0.2322 & -0.6086 & 0.3215 & 0.9793 & -0.1027 & 1.0000 \end{pmatrix}$$

Figure 9.11 The matrix of document correlation $M = N^TN$ in the new space (N is matrix R with length-normalized columns.)

$$Z = \begin{pmatrix} 1.0000 & & & & & \\ 0.5774 & 1.0000 & & & & \\ 0.6667 & 0 & 1.0000 & & & \\ 0 & 0 & 0.4082 & 1.0000 & & \\ 0.5774 & 0 & 0.5774 & 0 & 1.0000 & \\ 0 & 0 & 0 & 0.7071 & 0 & 1.0000 \end{pmatrix}$$

Figure 9.12 The matrix of document correlation $Z = Y^TY$ in the new space (Y is matrix X with length-normalized columns.)

The LSI performs IR based on concept. It is completely automatic and has been applied successfully (Deerwester et al. 1990, Foltz 1990) in many IR systems. However, it is costly in terms of computation.

EVALUATION OF THE IR SYSTEM

The evaluation of IR systems is the process of assessing how well a system meets the information needs of its users (Voorhees 2001). Evaluating an IR system is a difficult task involving a number of areas including cognition, statistics, and man-machine interactions. IR evaluation models can be broadly classified as system driven models and user-centered models. System driven models (Cleverdon et al. 1966) measure how well a system ranks documents; user-centered models measure user satisfaction. Cleverdon listed the following six criteria that can be used for evaluation:

1. *Coverage of the collection*: The extent to which the system
2. *Time lag*: The time that elapses between submission of a query and getting back the response
3. *Presentation format*
4. *User effort*: The effort made by the user to obtain relevant information
5. *Precision*: The proportion of retrieved documents that are relevant
6. *Recall*: The proportion of relevant documents that are retrieved

Of these criteria, recall and precision have most frequently been applied in measuring IR. Both are related to effectiveness, i.e., the ability of a system to retrieve relevant documents in response to user query. A number of effectiveness measures have been formulated (van Rijsbergen 1979). We discuss them in the following section. To better understand the relationship between aspects of retrieval process and different measures, see (Voorhees and Harman 1999), where correlations between pairs of measures are estimated.

The major goal of IR is to search for documents that are relevant to a user's query. It is necessary to understand what constitutes relevance, as the evaluation of IR systems relies on the notion of relevance.

9.7.1 Relevance

Relevance is subjective in nature (Saracevic 1991), i.e., it depends on the individual judgements of users. Given a query, the same document may be judged as relevant by one user and non-relevant by another. It is not possible to measure this 'true relevance' because no human can read all documents in a collection and provide a relevance assessment. Most evaluations of IR systems have so far been done on test document

collections with known relevance judgments. These test document collections contain documents from a particular discipline; for a set of questions representing information needs, relevance assessments are obtained from experts of that discipline. This provides an experimental setup for evaluating the performance of a retrieval strategy. If a retrieval strategy performs well under these situations, it is expected to perform well in an operational environment where relevance is not known.

Another issue with relevance is the degree of relevance. Traditionally, relevance has been visualized as a binary concept, i.e., a document is either relevant or not relevant; whereas relevance is actually a continuous function (a document may be exactly what the user wants or it may be closely related). This is an attractive but difficult proposition and current evaluation techniques do not support it.

A number of relevance frameworks have been proposed by Saracevic (1996). This includes system, communication, psychological, and situational frameworks. The most inclusive of these is the situational framework, which is based on a cognitive view of the information seeking process and considers the importance of situation, context, multi-dimensionality, and time. A survey of relevance studies has been discussed by Mizzaro (1996).

9.7.2 Effectiveness Measures

Effectiveness is purely a measure of the ability of a system to satisfy the user in terms of the relevance of documents retrieved (Rijsbergen 1979). Aspects of effectiveness include whether the documents returned are relevant to the user, whether they are presented in order of relevance, whether a significant number of relevant documents in the collection are returned to the user, etc. A number of measures have been proposed to quantify effectiveness. As stated earlier, the most commonly used measures of effectiveness are precision and recall. These measures are based on relevance judgments.

Precision and Recall

Precision is defined as the proportion of relevant documents in a retrieved set. This can be seen as the probability that a relevant document is retrieved. *Recall* is the proportion of relevant documents in a collection that have actually been retrieved. Precision measures the accuracy of a system while recall measures its exhaustiveness. Precision and recall can be computed as follows:

$$\text{Precision} = \frac{\text{Number of relevant document retrieved } (NR_{\text{ret}})}{\text{Total number of documents retrieved } (N_{\text{ret}})}$$

$$\text{Recall} = \frac{\text{Number of relevant documents retrieved } (NR_{r})}{\text{Total number of relevant documents in the collection } (NR_{\text{rel}})}$$

These definitions of precision and recall are based on binary relevance judgment, which means that every retrievable item is recognizably 'relevant', or recognizably 'not relevant'. Hence, for every search result, all retrievable documents will be either (i) relevant or non-relevant and (ii) retrieved or not retrieved. Thus, each document will fall into one, and only one, of four cells of the matrix, as sown in Figure 9.13. This matrix is used to derive a number of measures.

	Relevant	Non-relevant	
Retrieved	$A \cap B$	$\bar{A} \cap B$	B
Not-retrieved	$A \cap \bar{B}$	$\bar{A} \cap \bar{B}$	$\bar{B}$
	A	$\bar{A}$	

Figure 9.13 Relevant matrix

Referring to Figure 9.13, precision and recall will be given as follows:

$$\text{Precision} = \frac{|A \cap B|}{|B|} = \frac{NR_{\text{ret}}}{N_{\text{ret}}}$$

$$\text{Recall} = \frac{|A \cap B|}{|A|} = \frac{NR_{\text{ret}}}{NR_{\text{rel}}}$$

where A = Set of relevant documents
$|A|$ = No. of relevant documents in the collection (NR_{rel})
B = Set of retrieved documents
$|B|$ = No. of retrieved documents (NR_{ret})

It is clear from the preceding definitions that the total number of relevant documents in a collection must be known in order for recall to be calculated. The amount of effort and time required from the user makes this almost impossible in most operating environment. To provide a framework of evaluation for IR systems, a number of test collections have been developed (Cranfield and TREC). These collections are accompanied by a set of queries and relevance judgements. These test

collections make it possible for IR researchers to efficiently evaluate their experimental approaches and compare the effectiveness of their system with that of others. In Table 9.5, basic statistics for a number of test collections are presented.

Table 9.5 IR test collections

Collection	Number of documents	Number of queries
Cranfield	1400	225
CACM	3204	64
CISI	1460	112
LISA	6004	35
TIME	423	83
ADI	82	35
MEDLINE	1033	30
TREC-1	742,611	100

There exists a trade-off between precision and recall, though a high value of both at the same time is desirable. The trade-off is shown in Figure 9.14. Precision is high at low recall values. As recall increases, precision decreases. The ideal case of perfect retrieval requires that all relevant documents be retrieved before the first non-relevant document is retrieved. This is shown in the figure by the line parallel to x-axis having a precision of 1.0 at all recall points. Recall is an additive process. Once the highest recall (1.0) is achieved, it remains 1.0 for any subsequent document retrieved. We can always achieve 100% recall by retrieving all documents in the collection, but this defeats the intent of an IR system.

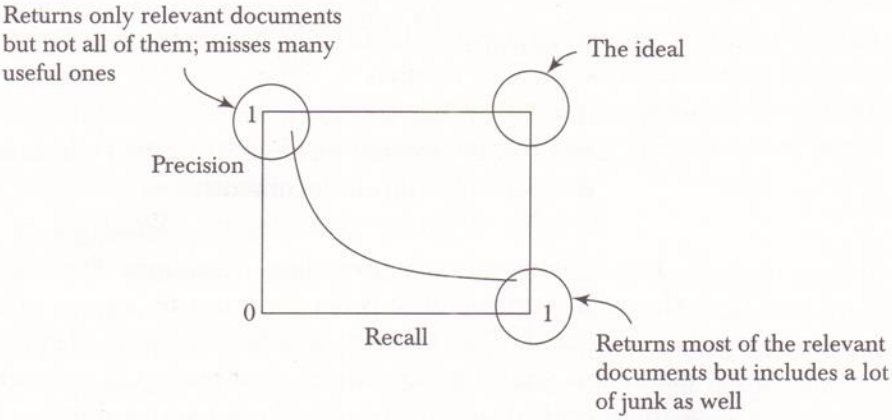


Figure 9.14 The trade-off between recall and precision

A number of researchers have discussed the relationship between recall and precision (Cleverdon 1972, Robertson 1975, Gordon and Kochen 1989, Buckland and Gey 1994). Some of them modelled precision and recall as continuous functions (Robertson 1975, Gordon and Kochen 1989), while others (Bookstein 1974) described recall and precision in terms of two-Poisson discrete model. Buckland and Gey (1994) studied the relationship between precision and recall, and suggested that a two-stage, or more generally, a multi-stage retrieval procedure is likely to achieve the goal of improving both precision and recall simultaneously, even though the trade-off between them cannot be avoided.

In order to evaluate the performance of an IR system, recall and precision are almost always used together. One measure is to calculate precision at a particular cut-off. Typical cut-offs are 5 documents, 10 documents, 15 documents, etc.

Yet another measure is *non-interpolated average precision*, which is average of the precision at observed recall points. Observed recall points correspond to points where a relevant document is retrieved. We first compute precision at each point where a relevant document is found and then compute average of these precision numbers to get a single number. Precision at relevant documents not in the returned set is assumed to be zero. We now give an example to illustrate how precision is calculated.

Table 9.6 An example of retrieval

Rank	Document #	Relevance
1	10	x
2	8	x
3	5	
4	3	
5	1	x
6	2	
7	4	
8	7	
9	9	x
10	6	x

Example 9.7 Table 9.6 shows the ranking of 10 documents for a particular retrieval. The crossed documents are those that are relevant. Let the total number of relevant document be five.

Precision values at 5 and 10 documents are given as follows:

Precision at 5 $3/10 = 0.3$

Precision at 10 $5/10 = 0.5$

Non-interpolated Average Precision The observed recall points are 0.2, 0.4, 0.6, 0.8, and 1.0. These recall values correspond to the documents marked relevant in the table. We have one of five relevant documents retrieved after retrieving only one document. This correspond to a recall value of $1/5 = 0.2$. After two documents, the recall is $2/5 = 0.4$. As the third document retrieved is not relevant, recall value does not change. The next relevant document is found after five documents have been retrieved, resulting in a recall value of $3/5 = 0.6$. Similarly, the other two recall points are calculated. The precision values at these points are as follows:

Precision at recall point 0.2: $1/1 = 1.0$
Precision at recall point 0.4: $2/2 = 1.0$
Precision at recall point 0.6: $3/5 = 0.6$
Precision at recall point 0.8: $4/9 = 0.4$
Precision at recall point 1.0: $5/10 = 0.5$
Non-interpolated average precision = 0.7

Considering the fact that a system may not always retrieve all the relevant documents, and that the number of relevant documents is not the same for all queries, precision values are interpolated for a set of recall points. The most widely used recall levels are 0.0, 0.1, 0.2, 0.3... 1.0. The precision values are calculated at each of these 11 recall levels and then averaged to get a single value. This is known as 11-point interpolated average precision. This has become almost a standard in evaluating the performance of an IR system. The interpolation used at TREC states that, precision at a given recall level is the greatest known precision at any recall level greater than or equal to this given level. For example, if the observed recall points are 0.25, 0.4, 0.55, 0.8, and 1.0, then precision at recall level 0.3 will be the maximum of the precision at recall levels 0.4, 0.55, 0.8, and 1.0, and not precision at recall point 0.4 where the 30% recall (i.e., recall level 0.3) is first reached. The interpolated precision at standard recall points for the documents shown in the Table 9.5 is computed in Example 9.8.

Example 9.8 Consider the following precision values at observed recall points:

0.25	1.0
0.4	0.67
0.55	0.8
0.8	0.6
1.0	0.5

The interpolated precision at the standard 11 recall levels will be

0.0	1.0
0.1	1.0
0.2	1.0
0.3	0.8
0.4	0.8
0.5	0.8
0.6	0.6
0.7	0.6
0.8	0.6
0.9	0.5
1.0	0.5

Interpolated average precision = 0.745

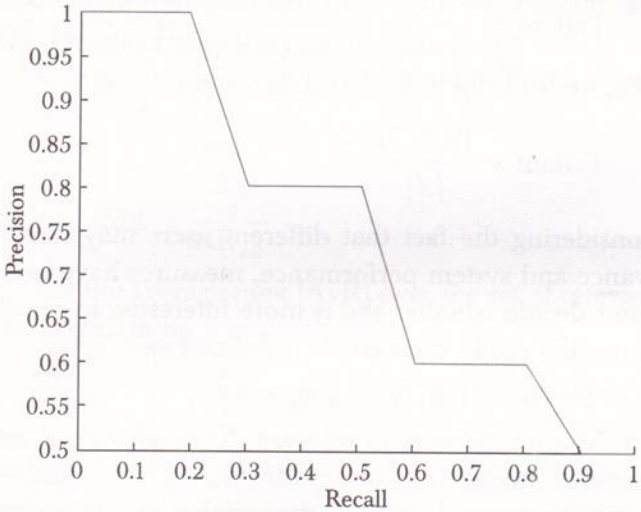


Figure 9.15 Recall–precision curve for interpolated precision

Often, precision values are calculated at different recall levels and a recall–precision graph, like the one shown in Figure 9.15, is plotted. As a retrieval system is evaluated over several queries, such a graph is usually plotted using precision figures averaged over all queries (Salton and McGill 1983, van Rijsbergen 1979). The most standard method for deriving a recall–precision graph is to plot the average over all queries in the interpolated precision values for a set of 11 standard recall points, namely 0.0, 0.1, 0.2, 0.3, ..., 1.0.

Instead of the recall–precision graph, the mean average precision is sometimes used to evaluate an IR system. The non-interpolated average

precision is averaged over all queries to get the mean average precision (MAP). Geometrically, MAP is the area below the non-interpolated recall–precision curve (Voorhees and Harman 1999). The 11-point interpolated average precisions (11 avgP) can also be used for calculating MAP, though the non-interpolated measure has the advantage that it rewards systems that quickly retrieve (give high ranks to) relevant documents.

The R-precision is the precision after a total number of R documents relevant to the query have been retrieved.

Recall is not defined if there is no relevant document in a collection. An alternative measure is fallout, which may be seen as the inverse of recall. It is not defined only if all the documents in the collection are relevant (Salton 1983). Fallout is the ratio of non-relevant documents retrieved to non-relevant documents in the collection.

$$\text{Fallout} = \frac{\text{Number of non-relevant documents retrieved } (N_n)}{\text{Number of non-relevant documents in the collection}}$$

For Figure 9.13, the fallout will be computed as

$$\text{Fallout} = \frac{|\bar{A} \cap B|}{|\bar{A}|}$$

Considering the fact that different users may have different ideas of relevance and system performance, measures have been developed to let the user decide whether she is more interested in recall or precision. For instance, the utility measure U is defined as

$$U = \alpha \cdot N_r + \beta \cdot \bar{N}_r + \delta N_n + \gamma \bar{N}_n$$

where N_r is the number of relevant documents retrieved, $\bar{N}_r$ the number of relevant documents not retrieved, N_n the number of non-relevant documents retrieved, and $\bar{N}_n$ the number of non-relevant documents not retrieved (α, β, δ, and γ are positive weights specified by the user). This measure was later simplified by considering only retrieved documents.

The F-measure takes into account both precision and recall. It is defined as the harmonic mean of recall and precision.

$$F = \frac{2PR}{P + R}$$

Compared to the arithmetic mean, both recall and precision need to be high for harmonic mean to be high.

The E-measure is a variant of the F-measure. It allows weighting to emphasize precision rather than recall. It is defined as

$$E = \frac{(1+\beta^2)PR}{\beta^2 P + R} = \frac{(1+\beta^2)}{\frac{\beta^2}{R} + \frac{1}{P}}$$

where P is precision, R is recall, and β is the relative importance of P compared to R. The value of β controls the trade-off between precision and recall. Setting β to 1 gives equal weight to precision and recall, resulting in a harmonic mean of recall and precision ($E = F$). $\beta > 1$ gives more weight to precision, and $\beta < 1$ gives more weight to recall.

Swets (1969) developed a model that received attention in the literature (Heine 1974; Bookstein 1977). None of these alternative measures, however, received as widespread acceptance as did the recall–precision model.

Normalized recall measures how close the set of retrieved documents is to an ideal retrieval, in which the most relevant NR_{rel} document appears in the first NR_{rel} position. Relevant documents are ranked 1, 2, 3, ..., NR_{rel}, where NR_{rel} is the number of relevant documents. The ideal rank is given by

$$IdR = \frac{\sum_{r=1}^{NR_{rel}} r}{NR_{rel}}$$

Let the average rank (AvR) over the set of relevant documents retrieved by a system be

$$AvR = \frac{\sum_{r=1}^{NR_{rel}} Rank_r}{NR_{rel}}$$

where $Rank_r$ represents the rank of the rth relevant document. The difference between AvR and IdR given by AvR – IdR, represents a measure of the effectiveness of the system. This difference can range from 0, for the perfect retrieval (AvR – IdR), to ($N - NR_{rel}$), for the worst case (N is the total number of documents in the collection). The worst case is when all the N documents are retrieved and the relevant documents, NR_{rel}, are the last retrieved. The expression AR–IR can be normalized by dividing it by ($N - NR_{rel}$) and then subtracting the result from 1. The normalized recall (NR) is given by

$$NR = 1 - \frac{AvR - IdR}{N - NR_{rel}}$$

This measure ranges from 1 for the best case, to 0 for the worst case. If the value of NR is close to 1, the ranks of relevant documents in average case deviate very little from the ideal case. A high value of NR indicates that the ranks of the relevant documents in the average case deviate considerably from the ideal case.

9.7.3 User-centred Evaluation

The system-driven model is still the dominant approach followed in IR research for evaluation of IR systems. The evaluation here, is made on a test collection having known relevance judgments. These relevance judgements were usually provided by problem domain experts, and are binary, objective, topical, and static in nature, and lack a user's viewpoint. There is also major disagreement among experts in providing relevance judgements (Haynes et al. 1990, Hersh and Hickam 1994). Further, these judgements of relevance are affected not only by the expertise of the judge, but also by the order of the documents (Eisenberg and Barry 1988; Schamber et al. 1990). It has been argued that relevance is not fixed, that it varies over time (Meadow 1992). The meaning and the relevance of a document can thus be different for different users and can be inferred only in the context of the user's situation. Relevance, therefore, is subjective, dynamic, and multi-dimensional in nature (Saracevic 1975, Mizzaro 1998, Harter 1992).

Another drawback of the system-driven approach is that it removes the end users from the retrieval process, substituting them with the queries and judgements provided with the test collection. This allows fast experimentation but makes it difficult to evaluate the effect of interactive IR techniques and is suitable only for non-interactive environment (Draper and Dunlop). In an interactive setting, a user normally starts with a query, which goes through many refinements to eventually get the desired documents. The test-collection approach poses a problem in such an environment. As performance of the IR system will eventually be measured in terms of it ability to retrieve documents relevant to a user's query, it seems realistic to follow a user-centered approach to evaluation. Such an approach will result in a much more direct measure of the overall goal. A number of measures have been proposed for interactive IR including relative relevance (RR), ranked half life (RHL), and cumulated gain (CG). Details of these measures can be found in Hersh et al. (1995), Borlund and Ingwerson (1998), and Järvelin and Kekäläinen (2000).

The subjective nature of interactive IR has been highlighted (Borlund and Ingwerson 1997) and attempts have been made to integrate cognitive

theory into IR evaluation. However, efforts in this direction have been limited. A task oriented, user-centred, non-interactive evaluation methodology has been proposed by Reid (2000), in which the basic unit of evaluation is task rather than query. More recently, an interactive IR evaluation model has been proposed by Borlund (2000, 2003) to evaluate interactive IR systems. The key elements of an interactive IR model are the use of realistic scenarios (simulated work tasks) and alternative performance measures such as RR and RHL. However, user-centred evaluation methods are expensive both in terms of time and resources. A properly designed user-centred evaluation, with a few exceptions, requires a sufficiently large, representative sample of actual users of retrieval systems. The systems to be compared must be equally well-developed and equipped with the appropriate user interface. The subject must be trained with these systems. Further, it is difficult to develop a standard interactive evaluation methodology that will allow for comparison across different systems and users (Reid 2000). Because of these considerations, recall and precision remain the most popular and standard measure for evaluating IR system performance.

- Information retrieval (IR) deals with the organization, storage, retrieval, and evaluation of information relevant to a user's query.
- An IR system does not return the actual information but returns the documents containing that information.
- The actual text of the document is not used in the retrieval process. Instead, documents in a collection are frequently represented through a set of index terms or keywords.
- The process of transforming document text to some representation of it is known as *indexing*.
- A common lexical processing of index terms involves elimination of stop words.
- An IR model is a pattern that defines several aspects of the retrieval procedure, for example, how the documents and users' queries are represented, how the system retrieves relevant documents according to users' queries, and how retrieved documents are ranked.
- Classical IR models, such as Boolean, vector space, and probabilistic, are based on mathematical knowledge that is easily recognized and well understood.

CHAPTER 12

LEXICAL RESOURCES

CHAPTER OVERVIEW

This chapter introduces various tools and lexical resources used in text processing applications and provides a ready reference of these. In particular, it introduces the reader with tools such as stemmers and taggers, lexical resources such as WordNet and FrameNet, and test collections (corpuses) that are freely available for research purpose.

12.1 INTRODUCTION

A whole range of tools and lexical resources have been developed to ease the task of researchers working with natural language processing (NLP). Many of these are open sources, i.e., readers can download them off the Internet. This chapter introduces some of the freely available resources. The motivation behind including this chapter comes from the belief that *knowing where the information is, is half of the information.*

We hope that providing a ready reference of what is available where, will save a lot of time and effort, especially for young researchers and those who are new to the field. All the material presented in this chapter is already available, at the links provided with the discussion or in the form of scholarly articles published on that resource. We bring these resources together and offer a brief discussion on them. In particular, we discuss lexical resources such as WordNet and FrameNet, and tools such as stemmers, taggers, and parsers, and freely available test corpuses for various text-processing applications. We begin our discussion with WordNet in Section 12.2. Section 12.3 discusses FrameNet. Stemmers are discussed in Section 12.4. We present a list of available part-of-speech taggers in Section 12.5. The next section presents a list of document collections. And finally, relevant journals and conferences are listed in Section 12.7.

12.2 WORDNET

WordNet[1] (Miller 1990, 1995) is a large lexical database for the English language. Inspired by psycholinguistic theories, it was developed and is being maintained at the Cognitive Science Laboratory, Princeton University, under the direction of George A. Miller. WordNet consists of three databases—one for nouns, one for verbs, and one for both adjectives and adverbs. Information is organized into sets of synonymous words called *synsets*, each representing one base concept. The synsets are linked to each other by means of lexical and semantic relations. Lexical relations occur between word-forms (i.e., senses) and semantic relations between word meanings. These relations include synonymy, hypernymy/hyponymy, antonymy, meronymy/holonymy, troponymy, etc. A word may appear in more than one synset and in more than one part-of-speech. The meaning of a word is called sense. WordNet lists all senses of a word, each sense belonging to a different synset. WordNet's sense-entries consist of a set of synonyms and a gloss. A gloss consists of a dictionary-style definition and examples demonstrating the use of a synset in a sentence, as shown in Figure 12.1. The figure shows the entries for

12.1.1 Noun

1. **read** (something that is read) *"the article was a very good read"*

12.1.2 Verb

1. **read** (interpret something that is written or printed) *"read the advertisement"; "Have you read Salman Rushdie?"*
2. **read**, say (have or contain a certain wording or form) *"The passage reads as follows"; "What does the law say?"*
3. **read** (look at, interpret, and say out loud something that is written or printed) *"The King will read the proclamation at noon"*
4. **read**, scan (obtain data from magnetic tapes) *"This dictionary can be read by the computer"*
5. **read** (interpret the significance of, as of palms, tea leaves, intestines, the sky; also of human behaviour) *"She read the sky and predicted rain"; "I can't read his strange behavior"; "The fortune teller read his fate in the crystal ball"*
6. take, **read** (interpret something in a certain way; convey a particular meaning or impression) *"I read this address as a satire"; "How should I take this message?"; "You can't take credit for this!"*
7. learn, study, **read**, take (be a student of a certain subject) *"She is reading for the bar exam"*
8. **read**, register, show, record (indicate a certain reading; of gauges and instruments) *"The thermometer showed thirteen degrees below zero"; "The gauge read `empty'"*
9. **read** (audition for a stage role by reading parts of a role) *"He is auditioning for `Julius Caesar' at Stratford this year"*
10. **read** (to hear and understand) *"I read you loud and clear!"*
11. understand, **read**, interpret, translate (make sense of a language) *"She understands French"; "Can you read Greek?"*

Figure 12.1 WordNet 2.0 entry for 'read'

[1] http://wordnet.princeton.edu/

the word 'read'. 'Read' has one sense as a noun and 11 senses as a verb. Glosses help differentiate meanings. Figures 12.2, 12.3, and 12.4 show some of the relationships that hold between nouns, verbs, and adjectives and adverbs. Nouns and verbs are organized into hierarchies based on the hypernymy/hyponymy relation, whereas adjectives are organized into clusters based on antonym pairs (or triplets). Figure 12.5 shows a hypernym chain for 'river' extracted from WordNet. Figure 12.6 shows the troponym relations for the verb 'laugh'.

Relation	*Definition*	*Example*
Hypernym	From concepts to super-ordinates	oak→ tree
Hyponym	From concepts to subtypes	oak→ white oak
Meronym	From wholes to parts	tree → trunk
Holonym	From parts to wholes	trunk → tree
Antonym	Opposites	victory → defeat

Figure 12.2 Noun relations in WordNet

Relation	*Definition*	*Example*
Hypernym	From events to super-ordinate events	wander→ travel
Troponym	From events to their subtypes	walk → stroll
Entails	From events to the events they entail	snore → sleep
Antonym	Opposites	increase → decrease

Figure 12.3 Verb relations in WordNet

Relation	*Definition*	*Example*
Antonym (adjective)	Opposite	heavy → light
Antonym (adverb)	Opposite	quickly → slowly

Figure 12.4 Adjective and adverb relations in WordNet

1 sense of 'river'
Sense 1
river — (a large natural stream of water (larger than a creek); 'the river was navigable for 50 miles')
=> stream, watercourse — (a natural body of running water flowing on or under the earth)
=> body of water, water — (the part of the earth's surface covered with water (such as a river or lake or ocean); 'they invaded our territorial waters'; 'they were sitting by the water's edge')
=> thing — (a separate and self-contained entity)
=> entity — (that which is perceived or known or inferred to have its own distinct existence (living or nonliving))

Figure 12.5 Hypernym chain for 'river'

WordNet is freely and publicly available for download from *http://wordnet.princeton.edu/obtain.*

WordNets for other languages have also been developed, e.g., EuroWordNet and Hindi WordNet. EuroWordNet covers European languages, including English, Dutch, Spanish, Italian, German, French, Czech, and Estonian. Other than language internal relations, it also contains multilingual relations from each WordNet to English meanings.

Hindi WordNet has been developed by CFILT (Resource Center for Indian Language Technology Solutions), IIT Bombay.[2] Its database consists of more than 26,208 synsets and 56,928 Hindi words.[3] It is organized using the same principles as English WordNet but includes some Hindi specific relations (e.g., causative relations). A total of 16 relations have been used in Hindi WordNet. Each entry consists of synset, gloss, and position of synset in ontology. Figure 12.7 shows the Hindi WordNet entry for the word 'आकांक्षा' (*aakanksha*).

Sense 1
laugh, express joy, express mirth — (produce laughter)
=> bray — (laugh loudly and harshly)
=> bellylaugh — (laugh a deep, hearty laugh)
=> roar, howl — (laugh unrestrainedly and heartily)
=> snicker, snigger — (laugh quietly)
=> giggle, titter — (laugh nervously; 'The girls giggled when the rock star came into the classroom')
=> break up, crack up — (laugh unrestrainedly)
=> cackle — (emit a loud, unpleasant kind of laughing)
=> guffaw, laugh loudly — (laugh boisterously)
=> chuckle, chortle, laugh softly — (laugh quietly or with restraint)
=> convulse — (be overcome with laughter)
=> cachinnate — (laugh loudly and in an unrestrained way)

Figure 12.6 Troponym relation for the word 'laugh'

Hindi WordNet can be obtained from the URL http://*www.cfilt.iitb.ac.in/wordnet/webhwn/.*

CFLIT has also developed a Marathi WordNet. Figure 12.8 shows the Marathi WordNet (*http://www.cfilt.iitb.ac.in/wordnet/webmwn/wn.php*) entry for the word 'पाव' (*pau*).

12.2.1 Applications of WordNet

WordNet has found numerous applications in problems related with IR and NLP. Some of these are discussed here.

p://www.cfilt.iitb.ac.in/
ndi WordNet Documentation http://www.cfilt.iitb.ac.in/hindi-wordnet-license01.pdf

Concept Identification in Natural Language

WordNet can be used to identify concepts pertaining to a term, to suit them to the full semantic richness and complexity of a given information need.

Word Sense Disambiguation

WordNet combines features of a number of the other resources commonly used in disambiguation work. It offers sense definitions of words, identifies synstes of synonyms, defines a number of semantic relations and is freely available. This makes it the (currently) best known and most utilized resource for word sense disambiguation. One of the earliest attempts to use WordNet for word sense disambiguation was in IR by Voorheese (1993). She used WordNet noun hierarchy (hypernym / hyponym) to achieve disambiguation. A number of other researchers have also used WordNet for the same purpose (Resnik 1995, 1997; Sussna 1993).

1. (R) **अपेक्षा, आकांक्षा, अन्वेक्षा** – किसी पर भरोसा रखने की क्रिया कि अमुक कार्य उसके द्वारा हो जायेगा " हर पिता की अपने पुत्र से यह अपेक्षा रहती है कि वह अपने जीवन में सफल हो "
2. (R) **इच्छा, अभिलाषा, आकांक्षा, ख्वाहिश, आरज़ू, तमन्ना, कामना, तलब, चेष्टा, हसरत, मुराद, पिपासा, प्यास, तृष्णा, मनोकामना, मनोकामना, मनोवांछा, मनोरथ, मनोभावना, मरज़ी, रज़ा, मर्ज़ी, मन, रज़ा, मंशा, लिप्सा, लालसा, तृषा, चाह, अरमान, क्षुधा, भूख, भूक, छुधा, हवस, स्पृहा, अभीप्सा, अनु, अपेक्षिता, अभिकांक्षा, अभिकाम, वांछा, वाञ्छा, वाञ्छा, अभिध्या, अभिलाष, अभिप्रीति, अभिमत, अभिमतता, अभिमति, अभिलास, अभिलासा, अभिलाखा, अभिलाख, अभिलाखना** — वह मनोवृत्ति जो किसी बात या वस्तु की प्राप्ति की ओर ध्यान ले जाती है " इंसान की हर इच्छा पूरी नही होती / उसकी ज्ञान पिपासा बढ़ती जा रही है / मेरा आज खाने का मन नहीं है "

Figure 12.7 WordNet entry for the Hindi word आकांक्षा (*aakanksha*)

1. (R) पाव एक चतुर्थांश-चौथा भाग "मी बाजारातून एक पाव चुरमुरे आणले"
2. (R) पाव, पावरोटी-गव्हाचे पीठ आंबवून केलेल एक खाद्यविशेष "मुंबईत बरेच लोक पाव खाऊन गुजराण करतात"
3. (R) पाव-पावाचे वजन "ढुकानदार चहाच्या पूडीचे वजन करण्यासाठी पाव शोधत आहे"
4. (R) पाव-एक चतुर्थांश बाटली दारू "एक पाव प्यायल्यावर तो वटवट करू लागलाए"

Figure 12.8 WordNet entry for the Marathi word पाव (*pau*)

Automatic Query Expansion

WordNet semantic relations can be used to expand queries so that the search for a document is not confined to the pattern-matching of query terms, but also covers synonyms. The work performed by Voorhees (1994) is based on the use of WordNet relations, such as synonyms, hypernyms, and hyponyms, to expand queries.

Document Structuring and Categorization

The semantic information extracted from WordNet, and WordNet conceptual representation of knowledge, have been used for text categorization (Scott and Matwin 1998).

Document Summarization

WordNet has found useful application in text summarization. The approach presented by Barzilay and Elhadad (1997) utilizes information from WordNet to compute lexical chains.

FRAMENET

FrameNet[4] is a large database of semantically annotated English sentences. It is based on principles of frame semantics. It defines a tagset of semantic roles called the frame element. Sentences from the British National Corpus are tagged with these frame elements. The basic philosophy involved is that each word evokes a particular situation with particular participants. FrameNet aims at capturing these situations through case-frame representation of words (verbs, adjectives, and nouns). The word that invokes a frame is called *target word* or *predicate*, and the participant entities are defined using semantic roles, which are called *frame elements.* The FrameNet ontology can be viewed as a semantic level representation of predicate argument structure.

Each frame contains a main lexical item as predicate and associated frame-specific semantic roles, such as AUTHORITIES, TIME, and SUSPECT in the ARREST frame, called frame elements. As an example, consider sentence (12.1) annotated with the semantic roles AUTHORITIES and SUSPECT. The target word in sentence (12.1) is 'nab' which is a verb in the ARREST frame.

[Authorities The police] nabbed [suspect the snatcher]. (12.1)

A COMMUNICATON frame has the semantic roles ADDRESSEE, COMMUNICATOR, TOPIC, and MEDIUM. Figure 12.9 shows the core and non-core frame elements of the COMMUNICATION frame, along with other details. A JUDGEMENT frame contains roles such as JUDGE, EVALUEE, and REASON. A frame may inherit roles from another frame. For example, a STATEMENT frame may inherit from a COMMUNICATION frame; it contains roles such as SPEAKER, ADDRESSEE, and MESSAGE. The following sentences show some of these roles:

[Judge She] [Evaluee blames the police] [Reason for failing to provide enough protection]. (12.2)

[Speaker She] told [Addressee me] [Message'I'll return by 7:00 pm today']. (12.3)

//framenet.icsi.berkeley.edu/

12.3.1 FrameNet Applications

Gildea and Jurafsky (2002) and Kwon et al. (2004) used FrameNet data for automatic semantic parsing. The shallow semantic role obtained from FrameNet can play an important role in information extraction. For example, a semantic role makes it possible to identify that the theme role played by 'match' is same in sentences (12.4) and (12.5) though the syntactic role is different.

The umpire stopped the match. (12.4)

The match stopped due to bad weather. (12.5)

In sentence (12.4), the word 'match' is the object, while it is the subject in sentence (12.5).

Semantic roles may help in the question-answering system. For example, the verb 'send' and 'receive' would share the semantic roles SENDER, RECIPIENT, GOODS, etc., (Gildea and Jurafsky 2002) when defined with respect to a common TRANSFER frame. Such common frames allow a question-answering system to answer a question such as 'Who sent packet to Khushbu?' using sentence (12.6).

Khushbu received a packet from the examination cell. (12.6)

Other applications include IR (Mohit and Narayanan 2003), interlingua for machine translation, text summarization, and word sense disambiguation.

Communication

Frame Elements	
Core:	
Addressee [Add]	Receiver of Message from the Communicator.
Communicator [Com]	The person conveying (written or spoken) a message to another person.
Message [Msg]	A proposition or set of propositions that the Communicator wants the Addressee to convey
Topic [Top]	The entity that the proposition(s) are about.
Non-core:	
Amount_of_information [Amo]	The amount of information exchanged when communication occurs.
Depictive [Dep-Act]	The Depictive describes the state of the Communicator.
Duration []	The length of time during which the communication takes place.
Manner [Manr]	The Manner in which the Communicator communicates.
Means [Mns]	The Means by which the Communicator communicates.
Medium [Medium]	The physical or abstract setting in which the Message is conveyed.
Time []	The time at which the communication takes place.

Inherits From:
Is Inherited By: Communication_noise, Statement
Subframe of:
Has Subframes:
Uses: Topic
Is Used By: Claim_ownership, Communication_response, Contacting, Deny_permission, Discussion, Hear, Questioning, Reasoning, Reporting, Request, etc
Is Inchoative of:
Is Causative of:
See Also:
Sample Predicates
communicate, indicate, signal, speech

Figure 12.9 Frame elements of communication frame

12.4 STEMMERS

As discussed in Chapter 3, stemming, often called conflation, is the process of reducing inflected (or sometimes derived) words to their base or root form. The stem need not be identical to the morphological base of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. Stemming is useful in search engines for query expansion or indexing and other NLP problems. Stemming programs are commonly referred to as stemmers. The most common algorithm for stemming English is Porter's algorithm[5] (Porter 1980). Other existing stemmers include Lovins[6] stemmer (Lovins 1968) and a more recent one called the Paice/Husk stemmer[7] (Paice 1990). Figure 12.10 shows a sample text and output produced using these stemmers.

Input Text:
Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation.

Output:
Lovins stemmer: such an analys can rev feature that ar not eas vis from the vari in th individu gen and can lead to a picture of expres that is mor biolog transpar and acces to interpres
Porter's stemmer: such an analysi can reveal feature that ar not easily visible from the variat in the individu gene and can lead to a picture of express that is more biolog transpar and access to interpret
Paice stemmer: Such an analys can rev feat that are not easy vis from the vary in the individ gen and can lead to a pict of express that is mor biolog transp and access to interpret

Figure 12.10 Stemmed text using different stemmers

12.4.1 Stemmers for European Languages

There are many stemmers available for English and other languages. Snowball[8] presents stemmers for English, Russian, and a number of other European languages, including French, Spanish, Portuguese, Hungarian, Italian, German, Dutch, Swedish, Norwegian, Danish, and Finnish. The links for stemming algorithms for these languages can be found at *http://snowball.tartarus.org/texts/stemmersoverview.html.*

[5]http://tartarus.org/~martin/PorterStemmer/
[6]http://sourceforge.net/project/showfiles.php?group_id=24260
[7]http://www.comp.lancs.ac.uk/computing/research/stemming/Links/implementations.htm
[8]http://snowball.tartarus.org/

12.4.2 Stemmers for Indian Languages

Standard stemmers are not yet available for Hindi and other Indian languages. The major research on Hindi stemming has been accomplished by Ramanathan and Rao (2003) and Majumder et al. (2007). Ramanathan and Rao (2003) based their work on the use of handcrafted suffix lists. Majumder et al. (2007) used a cluster-based approach to find classes of root words and their morphological variants. They used a task-based evaluation of their approach and reported that stemming improves recall for Indian languages. Their observation on Indian languages was based on a Bengali data set. The Resource Centre of Indian Language Technology (CFILT), IIT Bombay has also developed stemmers for Indian languages, which are available at *http://www.cfilt.iitb.ac.in.*

12.4.3 Stemming Applications

Stemmers are common elements in search and retrieval systems such as Web search engines. Stemming reduces the variants of a word to same stem. This reduces the size of the index and also helps retrieve documents that contain variants of a query terms. For example, a user issuing a query for documents on 'astronauts' would like documents on 'astronaut' as well. Stemming permits this by reducing both versions of the word to the same stem. However, the effectiveness of stemming for English query systems is not too great, and in some cases may even reduce precision.

Text summarization and text categorization also involve term frequency analysis to find features. In this analysis, stemming is used to transform various morphological forms of words into their stems.

PART-OF-SPEECH TAGGER

Part-of-speech tagging is used at an early stage of text processing in many NLP applications such as speech synthesis, machine translation, IR, and information extraction. In IR, part-of-speech tagging can be used in indexing (for identifying useful tokens like nouns), extracting phrases and for disambiguating word senses. The rest of this section presents a number of part-of-speech taggers that are already in place.

12.5.1 Stanford Log-linear Part-of-Speech (POS) Tagger

This POS Tagger is based on maximum entropy Markov models. The key features of the tagger are as follows:

(i) It makes explicit use of both the preceding and following tag contexts via a dependency network representation.

(ii) It uses a broad range of lexical features.
(iii) It utilizes priors in conditional log-linear models.

The reported accuracy of this tagger on the Penn Treebank WSJ is 97.24%, which amounts to an error reduction of 4.4% on the best previous single automatically learned tagging result (Tuotanova et al. 2003). Details on the tagger can be found at the link *http://nlp.stanford.edu/software/tagger.shtml.*

12.5.2 A Part-of-Speech Tagger for English[9]

This tagger uses a bi-directional inference algorithm for part-of-speech tagging. It is based on maximum entropy Markov models (MEMM). The algorithm can enumerate all possible decomposition structures and find the highest probability sequence together with the corresponding decomposition structure in polynomial time. Experimental results of this part-of-speech tagger show that the proposed bi-directional inference methods consistently outperform unidirectional inference methods and bi-directional MEMMs give comparable performance to that achieved by state-of-the-art learning algorithms, including kernel support vector machines (Tsuruoka and Tsujii 2005).

12.5.3 TnT tagger[10]

Trigrams'n'Tags or TnT (Brants 2000) is an efficient statistical part-of-speech tagger. This tagger is based on hidden Markov models (HMM) and uses some optimization techniques for smoothing and handling unknown words. It performs at least as well as other current approaches, including the maximum entropy framework. Table 12.1 shows tagged text of document #93 of the CACM collection.

Table 12.1 Doc #93 of CACM collection tagged using TnT tagger

A	DT	simple	JJ
technique	NN	algebraic	JJ
is	VBZ	formulas	NNS
shown	VBN	into	IN
for	IN	a	DT
enabling	VBG	three	CD
a	DT	address	NN
computer	NN	computer	NN
to	TO	code	NN
translate	VB		

[9]http://www-tsujii.is.s.u-tokyo.ac.jp/~tsuruoka/postagger/
[10]http://www.coli.uni-saarland.de/~thorsten/tnt/

12.5.4 Brill Tagger

Brill (1992) described a trainable rule-based tagger that obtained performance comparable to that of stochastic taggers. It uses transformation-based learning to automatically induce rules. A number of extensions to this rule-based tagger have been proposed by Brill (1994). He describes a method for expressing lexical relations in tagging that stochastic taggers are currently unable to express. It implements a rule-based approach to tagging unknown words. It demonstrates how the tagger can be extended into a k-best tagger, where multiple tags can be assigned to words in some cases of uncertainty. Brill tagger is available for download at the link *http://www.cs.jhu.edu/~brill/RBT1_14.tar.Z.*

12.5.5 CLAWS Part-of-Speech Tagger for English

Constituent likelihood automatic word-tagging system (CLAWS) is one of the earliest probabilistic taggers for English. It was developed at the University of Lancaster (*http://ucrel.lancs.ac.uk/claws*). The latest version of the tagger, CLAWS4, can be considered a hybrid tagger as it involves both probabilistic and rule-based elements. It has been designed so that it can be easily adapted to different types of text in different input formats. CLAWS has achieved 96–97% accuracy. The precise degree of accuracy varies according to the type of text. For more information on the CLAWS tagger, see Garside (1987), Leech, Garside, and Bryant (1994), Garside (1996), and Garside and Smith (1997).

12.5.6 Tree-Tagger

Tree-Tagger (Schmidt 1994) is a probabilistic tagging method. It avoids problems faced by the Markov model methods when estimating transition probabilities from sparse data, by using a decision tree to estimate transition probabilities. The decision tree automatically determines the appropriate size of the context to be used in estimation. The reported accuracy for the tagger is above 96% on the Penn-Treebank WSJ corpus. The tagger is available at the link *http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html.*

12.5.7 ACOPOST: A Collection of POS Taggers[11]

ACOPOST is a set of freely available POS taggers. The taggers in the set are based on different frameworks. The programs are written in C. ACOPOST currently consists of the following four taggers.

[11] copost.sourceforge.net/

Maximum Entropy Tagger (MET)

This tagger is based on a framework suggested by Ratnaparkhi (1997). It uses an iterative procedure to successively improve parameters for a set of features that help to distinguish between relevant contexts.

Trigram Tagger (T3)

This tagger is based on HMM. The states in the model are tag pairs that emit words. The technique has been suggested by Rabiner (1990) and the implementation is influenced by Brants (2000).

Error-driven Transformation-based Tagger (TBT)

This tagger is based on the transformation-based tagging approach proposed by Brill (1993). It uses annotated corpuses to learn transformation rules, which are then used to change the assigned tag using contextual information.

Example-based Tagger (ET)

The underlying assumption of example-based models (also called memory-based, instance-based or distance-based models) is that cognitive behaviour can be achieved by looking at past experiences that match the current problem, instead of learning and applying abstract rules. This framework has been suggested for NLP by Daelemans et al. (1996).

12.5.7 POS Tagger for Indian Languages

The automatic text processing of Hindi and other Indian languages is constrained heavily due to lack of basic tools and large annotated corpuses. Research groups are now focusing on removing these bottlenecks. The work on the development of tools, techniques, and corpora is going on at several places such as CDAC, IIT Bombay, IIIT Hyderabad, University of Hyderabad, CIIL Mysore, and University of Lancaster. IIT Bombay is involved in the development of morphology analysers and part-of-speech taggers for Hindi and Marathi. Both these languages have rich morphological structures. Their approach is based on *bootstrapping on a small corpus tagged* by a rule-based tagger and then applying statistical techniques to train a machine. More information can be found at *http://ltrc.iiit.net* and *www.cse.iitb.ac.in.* Work on Urdu part-of-speech taggers has been reported by Hardie (2003) and Baker et al. (2004).

RESEARCH CORPORA

Research corpora have been developed for a number of NLP-related tasks. In the following section, we point out few of the available standard document collections for a variety of NLP-related tasks, along with their Internet links.

12.6.1 IR Test Collection

We have already provided a list of IR test document collection in Chapter 9. Glasgow University, UK, maintains a list of freely available IR test collections. Table 12.2 lists the sources of those and few more IR test collections.

LETOR (learning to rank) is a package of benchmark data sets released by Microsoft Research Asia. It consists of two datasets OHSUMED and TREC (TD2003 and TD2004). LETOR is packaged with extracted features for each query-document pair in the collection, baseline results of several state-of-the-art learning-to-rank algorithms on the data and evaluation tools. The data set is aimed at supporting future research in the area of learning ranking function for information retrieval.

Table 12.2 IR test collection

LETOR	*http://research.microsoft.com/users/tyliu/LETOR/*
LISA	
CACM	
CISI	
MEDLINE	*http://www.dcs.gla.ac.uk/idom/ir_resources/test_collections/*
Cranfield	
TIME	
ADI	

12.6.2 Summarization Data

Evaluating a text summarizing system requires existence of 'gold summaries'. DUC provides document collections with known extracts and abstracts, which are used for evaluating performance of summarization systems submitted at TREC conferences. Figure12.11 shows a sample document and its extract from DUC 2002 summarization data.

<DOC>
<DOCNO> AP880911-0016 </DOCNO>
<FILEID>AP-NR-09-11-88 0423EDT</FILEID>
<FIRST>r i BC-HurricaneGilbert 09-11 0339</FIRST>
<SECOND>BC-Hurricane Gilbert,0348</SECOND>
<HEAD>Hurricane Gilbert Heads Toward Dominican Coast</HEAD>
<BYLINE>By RUDDY GONZALEZ</BYLINE>
<BYLINE>Associated Press Writer</BYLINE>
<DATELINE>SANTO DOMINGO, Dominican Republic (AP) </DATELINE>
<TEXT>

Hurricane Gilbert swept toward the Dominican Republic Sunday, and the Civil Defense alerted its heavily populated south coast to prepare for high winds, heavy rains, and high seas.

The storm was approaching from the southeast with sustained winds of 75 mph gusting to 92 mph.

"There is no need for alarm," Civil Defense Director Eugenio Cabral said in a television alert shortly before midnight Saturday.

Cabral said residents of the province of Barahona should closely follow Gilbert's movement. An estimated 100,000 people live in the province, including 70,000 in the city of Barahona, about 125 miles west of Santo Domingo.

Tropical Storm Gilbert formed in the eastern Caribbean and strengthened into a hurricane Saturday night. The National Hurricane Center in Miami reported its position at 2 a.m. Sunday at latitude 16.1 north, longitude 67.5 west, about 140 miles south of Ponce, Puerto Rico, and 200 miles southeast of Santo Domingo.

The National Weather Service in San Juan, Puerto Rico, said Gilbert was moving westward at 15 mph with a "broad area of cloudiness and heavy weather" rotating around the center of the storm.

The weather service issued a flash flood watch for Puerto Rico and the Virgin Islands until at least 6 p.m. Sunday.

Strong winds associated with the Gilbert brought coastal flooding, strong southeast winds and up to 12 feet to Puerto Rico's south coast. There were no reports of casualties.

San Juan, on the north coast, had heavy rains and gusts Saturday, but they subsided during the night.

On Saturday, Hurricane Florence was downgraded to a tropical storm and its remnants pushed inland from the U.S. Gulf Coast. Residents returned home, happy to find little damage from 80 mph winds and sheets of rain.

Florence, the sixth named storm of the 1988 Atlantic storm season, was the second hurricane. The first, Debby, reached minimal hurricane strength briefly before hitting the Mexican coast last month.

</TEXT>
</DOC>

Extract

Tropical Storm Gilbert in the eastern Caribbean strengthened into a hurricane Saturday night. The National Hurricane Center in Miami reported its position at 2 a.m. Sunday to be about 140 miles south of Puerto Rico and 200 miles southeast of Santo Domingo. It is moving westward at 15mph with a broad area of cloudiness and heavy weather with sustained winds of 75mph gusting to 92mph. The Dominican Republic's Civil Defense alerted that country's heavily populated south coast and the National Weather Service in San Juan, Puerto Rico issued a flood watch for Puerto Rico and the Virgin Islands until at least 6 p.m. Sunday.

Figure 12.11 Sample document from DUC 2002 and its extract

12.6.3 Word Sense Disambiguation

SEMCOR[12] is a sense-tagged corpus used in disambiguation. It is a subset of the Brown corpus, sense-tagged with WordNet synsets. Open Mind Word Expert[13] attempts to create a very large sense-tagged corpus. It collects word sense tagging from the general public over the Web.

12.6.4 Asian Language Corpora

The multilingual EMILLE corpus is the result of the enabling minority language engineering (EMILLE) project at Lancaster University, UK. The project focuses on generation of data, software resources and basic language engineering tools for the NLP of south Asian languages. Central Institute for Indian Languages (CIIL), the Indian partner in the project, extended the set of target languages to include a number of Indian languages. CIIL provides a wider range of data in these languages from a wide range of genres. The data sources that EMILLE made available include monolingual written and spoken corpuses, parallel and annotated corpuses. The full EMILLE/CIIL corpus is available for free, but for research use only, at the link *http://www.elda.org/catalogue/en/text/W0037.html.* Further details about the corpus can be found in the manual at the site *http://www.emille.lancs.ac.uk/manual.pdf.*

Corpus building in these languages is constrained by the scarcity of repositories of electronic text. The monolingual corpus includes written data for 14 South Asian languages and spoken data for five languages (Hindi, Bengali, Gujrati, Punjabi, and Urdu). The spoken corpus was constructed from radio broadcasts on the BBC Asia network. The parallel corpus contains English text and its translation in five languages. The text includes UK government advice leaflets which are published in multiple languages. The corpus is aligned at sentence level. The parallel corpus provided by EMILLE corpus is a valuable resource for statistical machine translation research. The annotated component includes Urdu data annotated for part-of-speech tagging, and a Hindi corpus annotated to show nature of demonstrative use.

12.7 JOURNALS AND CONFERENCES IN THE AREA

A wide number of conference proceedings and journals report research in the various areas of NLP. Most notable among them are those associated with Association for Computing Machinery (ACM), Association for

[12] http://www.cs.unt.edu/~rada/downloads.html#semcor
[13] http://teach-computers.org.

Computational Linguistics (ACL), its European counterpart EACL, Recherche d'Information Assistie par Ordinateur (RIAO) and the International Conferences on Computational Linguistics (COLING). The ACM SIGIR Conference is one of the major conferences held on research and development in information retrieval. It provides the international forum for dissemination of research and demonstration of new systems and techniques. The 30th Annual International ACM SIGIR Conference[14] was held on 23–27 July 2007 at Amsterdam. The Proceedings of Text Retrieval Conferences (TRECs)[15] are another important source of information. These proceedings report results from standardized evaluations organized by the US government. The TRECs have been organized regularly since 1992 as a part of the TIPSTER text retrieval. They were earlier known as the Document Understanding Conference or Message Understanding Conferences. The conference series is sponsored by the National Institute of Standards and Technology (NIST) with additional support from other US government agencies. The ACM Special Interest Group on Information Retrieval (ACM-SIGIR) focuses on IR related tasks, and ECIR is its European counterpart. The NTCIR (NII test collection for IR) focuses on information retrieval with Japanese and other Asian languages.

KES[16] International Conferences in Knowledge-Based and Intelligent Engineering & Information Systems have been a regular feature since 1997. The conference mainly focuses on applications of intelligent systems. The topics covered by KES includes general intelligent topics like neural networks, fuzzy techniques, genetic algorithms, knowledge representation and management, applications using intelligent techniques (e.g., speech processing and synthesis and NLP) and emerging intelligent technologies like intelligent information retrieval, intelligent web mining and applications, intelligent user interfaces, etc.

HLT-NACCL is sponsored by the North American chapter of the Association for Computational Linguistics.

The *Journal of Computational Linguistics* is a leading premier publication focussing on theoretical and linguistics aspects. More practical applications are covered in the *Natural Language Engineering Journal. Information Retrieval* by Kluwer, *Information Processing and Management* by Elsevier, ACM's *Transactions on Information Systems* (TOIS), *Journal of American Society for Information Sciences* are major journals covering a wide range of information

[4] http://www.sigir2007.org/
[5] http://trec.nist.gov/
[6] http://www.kesinternational.org/conferences.php

Research, International Journal of Information Technology and Decision Making (World Scientific), and *Journal of Digital Information Management and Information System.*

A few AI publications also report work on language processing. Among these are *Artificial Intelligence, Computational Intelligence,* IEEE's T*ransaction on Intelligent Systems,* and *Journal of AI Research.*

SUMMARY

- Lexical resources such as WordNet and FrameNet can be used in a number of NLP-related tasks.
- Stemmers are useful in a number of information processing tasks such as information retrieval, text summarization, and text categorization.
- Widely known stemmers include Porter's and Lovins stemmers.
- Part-of-speech tagger is used to assign a part-of-speech, such as noun, verb, pronoun, preposition, adverb, and adjective, to each word in a sentence (or text).
- Taggers include stanford log-linear part-of-speech tagger, TnT, CLAWS, and Brill's tagger.
- TREC and SIGIR conferences offer useful resources for a number of information processing-related tasks.

REFERENCES

Ananthkrishnan, R. and Durgesh Rao, 2003, 'A lightweight stemmer for Hindi,' Workshop on Computational Linguistics for South Asian Languages, *The 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03)*, ACL, Morristown, NJ.

Baker, P., A. Hardie, A.M. McEnery, and B.D. Jayaram, 2004, 'Corpus linguistics and South Asian languages: corpus creation and tool development,' *Literary and Linguistic Computing,* 19(4), 509–24.

Barzilay, Regina and Michael Elhadad, 1997, 'Using lexical chains for text summarization,' *Proceedings of the Intelligent Scalable Text Summarization Workshop (ISTS'97)*, ACL, Madrid.

Brants, Thosrten, 2000, 'TnT–as statistical part-of-speech tagger,' *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP-2000)*, Seattle, WA.

Brill E., 1992, 'A simple rule-based part-of-speech tagger,' *Proceedings of the Third Conference on Applied Natural Language Processing,* ACL, Budapest, Hungary.