SYLLABUS

Natural Language Processing - [410252(A)]

Credit	Examination Scheme :
03	In-Sem (Paper) : 30 Marks
	End-Sem (Paper) : 70 Marks

Unit I Introduction to Natural Language Processing

Introduction : Natural Language Processing, Why NLP is hard ? Programming languages Vs Natural Languages, Are natural languages regular ? Finite automata for NLP, Stages of NLP, Challenges and Issues(Open Problems) in NLP.

Basics of text processing : Tokenization, Stemming, Lemmatization, Part of Speech Tagging. **(Chapter - 1)**

Unit II Language Syntax and Semantics

Morphological Analysis : What is Morphology ? Types of Morphemes, Inflectional morphology & Derivational morphology, Morphological parsing with Finite State Transducers (FST).

Syntactic Analysis : Syntactic Representations of Natural Language, Parsing Algorithms, Probabilistic context-free grammars, and Statistical parsing.

Semantic Analysis : Lexical Semantic, Relations among lexemes & their senses - Homonymy, Polysemy, Synonymy, Hyponymy, WordNet, Word Sense Disambiguation (WSD), Dictionary based approach, Latent Semantic Analysis. **(Chapter - 2)**

Unit III Language Modelling

Probabilistic language modeling, Markov models, Generative models of language, Log-Liner Models, Graph-based Models.

N-gram models : Simple n-gram models, Estimation parameters and smoothing, Evaluating language models, **Word Embeddings / Vector Semantics** : Bag-of-words, TFIDF, word2vec, doc2vec, Contextualized representations (BERT).

Topic Modelling : Latent Dirichlet Allocation (LDA), Latent Semantic Analysis, Non Negative Matrix Factorization. **(Chapter - 3)**

Unit IV Information Retrieval using NLP

Information Retrieval : Introduction, Vector Space Model.

Named Entity Recognition : NER System Building Process, Evaluating NER System, Entity Extraction, Relation Extraction, Reference Resolution, Coreference resolution, Cross Lingual Information Retrieval. **(Chapter - 4)**

Unit V NLP Tools and Techniques

Prominent NLP Libraries : Natural Language Tool Kit (NLTK), spaCy, TextBlob, Gensim etc.

Linguistic Resources : Lexical Knowledge Networks, WordNets, Indian Language WordNet (IndoWordnet), VerbNets, PropBank, Treebanks, Universal Dependency Treebanks.

Word Sense Disambiguation : Lesk Algorithm Walker's algorithm, WordNets for Word Sense Disambiguation. **(Chapter - 5)**

Unit VI Applications of NLP

Machine Translation : Rule based techniques, Statistical Machine Translation (SMT), Cross Lingual Translation Sentiment Analysis, Question Answering, Text Entailment, Discourse Processing, Dialog and Conversational Agents, Natural Language Generation. **(Chapter - 6)**

TABLE OF CONTENTS

Unit I

Unit II

Unit III

Unit IV

Unit V

Unit VI

Unit I

1 Introduction to Natural Language Processing

Syllabus

Introduction : *Natural Language Processing, Why NLP is hard ? Programming languages Vs Natural Languages, Are natural languages regular ? Finite automata for NLP, Stages of NLP, Challenges and Issues(Open Problems) in NLP.*

Basics of text processing : *Tokenization, Stemming, Lemmatization, Part of Speech Tagging.*

Contents

1.1 Introduction

- By natural language processing, we refer to computational techniques that process spoken and written human language sentences, as input.
- Natural Language Processing (or NLP) is an area that is a confluence of artificial intelligence and linguistics. It involves intelligent analysis of written/spoken human language. Natural language refers to the way we, humans, communicate with each other, namely, speech and text.
- NLP is a part of computer science, human language and artificial intelligence. It is the technology that is used by computer to understand, analyse, manipulate and interpret human's languages.
- The result is a computer capable of "understanding" the contents of documents, including the contextual nuances of the language within them. The technology then accurately extracts information and insights contained in the documents as well as categorize and organize the documents themselves.
- Natural Language Processing (NLP) is a subfield of linguistics, computer science and artificial intelligence concerned with the interactions between computers and human language.
- Natural Language Processing (NLP) deals with how computers understand and translate human language. With NLP, machines can make sense of written or spoken text and perform tasks like translation, keyword extraction, topic classification and more.
- NLP draws from many disciplines, including computer science and computational linguistics, in its pursuit to fill the gap between human communication and computer understanding.
- NLP in particular deals with how to program computers to process and analyse large amounts of natural language data. The result is a computer capable of 'understanding' the contents of documents, including the contextual nuances of the language within them.
- The study of natural language processing has been around for more than 50 years and grew out of the field of linguistics with the rise of computers.
- While natural language processing isn't a new science, the technology is rapidly advancing thanks to an increased interest in human-to-machine communications, plus an availability of big data, powerful computing and enhanced algorithms.

History of natural language processing

Natural language processing has its roots in the 1950s. Already in 1950, Alan Turing published an article titled "Computing Machinery and Intelligence" which proposed what is now called the **Turing test** as a criterion of intelligence.

- In 1957, Noam Chomsky's Syntactic Structures revolutionized Linguistics with 'universal grammar', a rule based system of syntactic structures.
- The Georgetown experiment in 1954 involved fully automatic translation of more than sixty Russian sentences into English. The authors claimed that within three or five years, machine translation would be a solved problem.
- In 1969 Roger Schank introduced the conceptual dependency theory for natural language understanding. This model, partially influenced by the work of Sydney Lamb, was extensively used by Schank's students at Yale University, such as Robert Wilensky, Wendy Lehnert and Janet Kolodner.
- In 1970, William A. Woods introduced the Augmented Transition Network (ATN) to represent natural language input. Instead of phrase structure rules ATNs used an equivalent set of finite state automata that were called **recursively.**
- Up to the 1980s, most natural language processing systems were based on complex sets of hand-written rules. Starting in the late 1980s, however, there was a revolution in natural language processing with the introduction of machine learning algorithms for language processing.
- 1990s : Many of the notable early successes on statistical methods in NLP occurred in the field of machine translation, due especially to work at IBM research. These systems were able to take advantage of existing multilingual textual corpora.
- 2000s : With the growth of the web, increasing amounts of raw (unannotated) language data has become available since the mid-1990s. Research has thus increasingly focused on unsupervised and semi-supervised learning algorithms.
- In the 2010s, representation learning and deep neural network-style machine learning methods became widespread in natural language processing, due in part to a flurry of results showing that such techniques can achieve state-of-the-art results in many natural language tasks, for example in language modelling, parsing and many others.

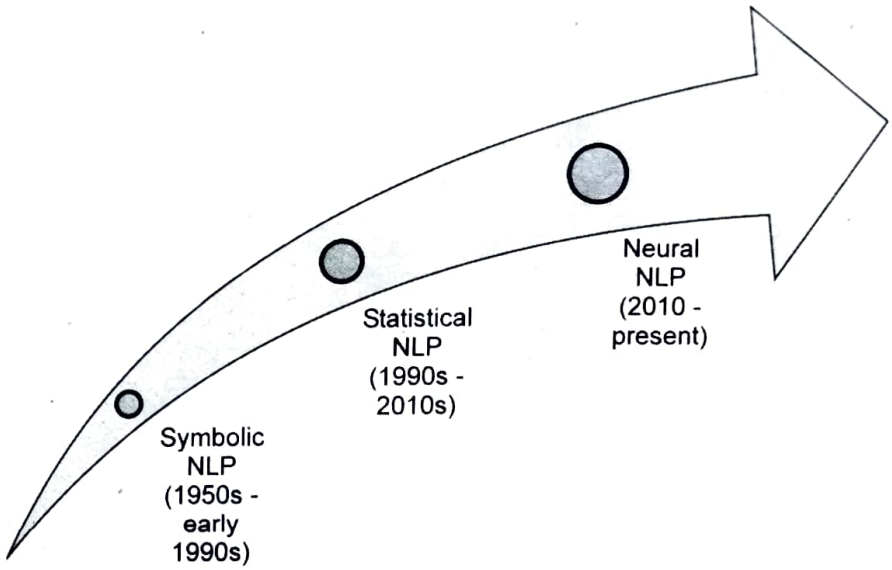


Fig 1.1.1 Symbolic representation of NLP history

- In the late 1950s and the early 1960s, speech and language processing had separated very cleanly into two paradigms : Symbolic and stochastic. The symbolic paradigm took off from two lines of research.
  - The first paradigm focused on on formal language theory and generative syntax, parsing algorithms, initially top-down and bottom-up and later with dynamic programming.
  - The second is the stochastic paradigm. It was mainly carried out in departments of statistics and of electrical engineering. It focused on optical character recognition, text recognition.
- Four paradigms in 1970-1983

  Explosion in research in speech and language processing, gave rise to the development of a number of research paradigms which still dominate the field. These are,
  - The **stochastic paradigm** played a huge role in the development of speech recognition algorithms in this period.
  - **The logic-based paradigm** focused on grammar based research, mainly on functional grammar, metamorphosis grammars.
  - **The natural language understanding** field took off with simulated robots that can accept human commands. This work was based on human conceptual knowledge such as scripts, plans and goals and human memory organization. The logic-based and natural-language understanding paradigms were unified on systems that used predicate logic as a semantic representation, such as the LUNAR question-answering system.

- **The discourse modelling paradigm** focused on four key areas in discourse, i.e. argument, narration, description and exposition.

Review Question

1. *What is NLP ?*

1.2 Why NLP is Hard ?

NLP is different

- What distinguishes NLP applications from other data processing systems is their use of knowledge of language.
- Natural language processing is considered a difficult problem in computer science. It's the nature of the human language that makes NLP difficult. Human brains can easily master a language, the ambiguity and imprecise characteristics of the natural languages. These aspects of natural language make NLP difficult for computers to implement.
- The central challenge of natural language processing is ambiguity. It exists at every level or stage of NLP.
- For NLP, we say some input is ambiguous when there are multiple alternative linguistic structures than can be built for it. Here is an example.
- Here is an example that gives different meanings highlighting ambiguity at some level :

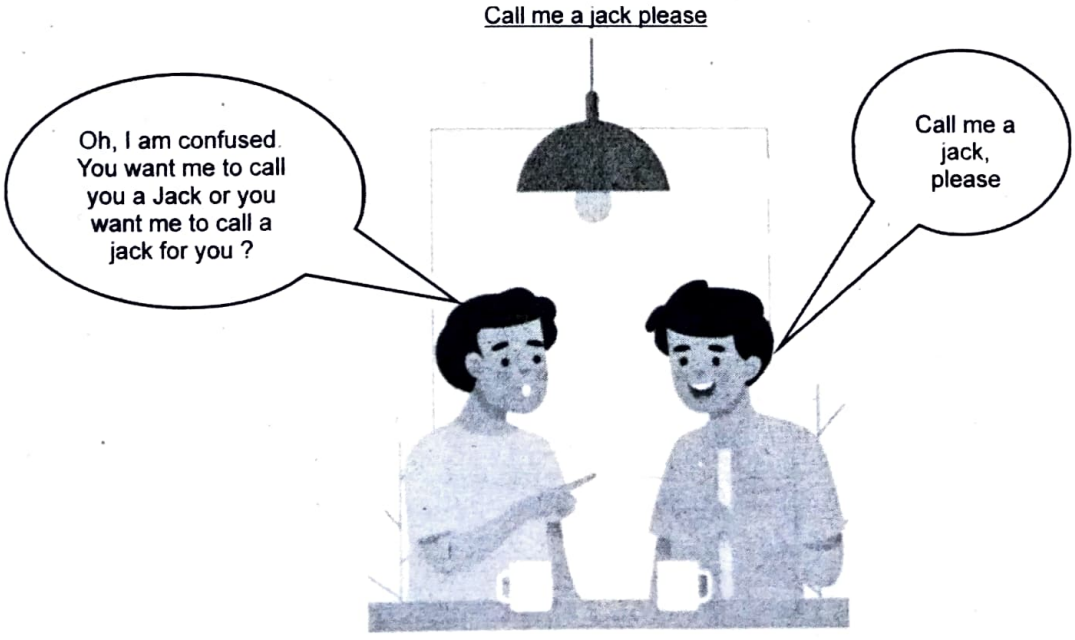


Fig 1.2.1 Example of ambiguity in natural language sentence(s)

- This sentence creates following different meaning for the listener :
  - The person in blue shirt is asking to be addressed as Jack (male / female not clear).
  - The person in blue shirt is requesting to be referred as a jack i.e. a helper.
  - The person in blue orders that he wants a jack (a portable device for raising or lifting heavy objects short heights).
  - The person in blue orders that he wants a jack (a connecting device in an electrical circuit designed for the insertion of a plug.)
  - The person in the blue shirt wants himself to be referred to as a jack - A playing card bearing the picture of a soldier or servant.
  - The person in blue wants to refer himself as the one who can to speed up or move (something) fast to make any progress.
- NLP research has introduced some processing models and algorithms resolve these ambiguities :
  - Deciding whether jack is a verb or a noun can be solved by part of speech tagging.
  - Deciding whether call means 'bring / order' or 'address' can be solved by word sense disambiguation.
  - Deciding whether call and jack are part of the same entity in various meanings can be solved by probabilistic parsing.
  - Ambiguities that may arise in this example (like whether a given sentence is a request or an order or a plain sentence) can be resolved, for example by speech act interpretation.

Review Question

1. *Why NLP is hard ?*

1.3 Programming Languages Vs Natural Languages

- Natural language processing is a field of linguistics and computer science which focuses on :
  1. Processing natural language.
  2. A natural language is a human spoken language, in opposition to artificial languages such as computer languages C or Cobol.
  3. In other words, a natural language is nothing more than a spoken language such as French, English or Hindi, etc.

4. Below are the main differences between natural language and computer language :

Parameter	Natural language	Computer language
Ambiguous	They are ambiguous in nature.	They are designed to be unambiguous.
Redundancy	Natural languages employ lots of redundancy.	Formal languages are less redundant.
Literalness	Natural languages are made of idiom & metaphor.	Formal languages mean exactly what they want to say.

- NLP tasks require knowledge about written human language i.e. word sense, grammar and disambiguation. Understanding variations of individual words (for example recognizing that doors is plural) requires knowledge about morphology, which captures information about the shape and behaviour of words in context.
- The knowledge needed to order and group words together comes under the heading of syntax.
- NLP requires knowledge of the meanings of the component words i.e. the domain of lexical semantics.

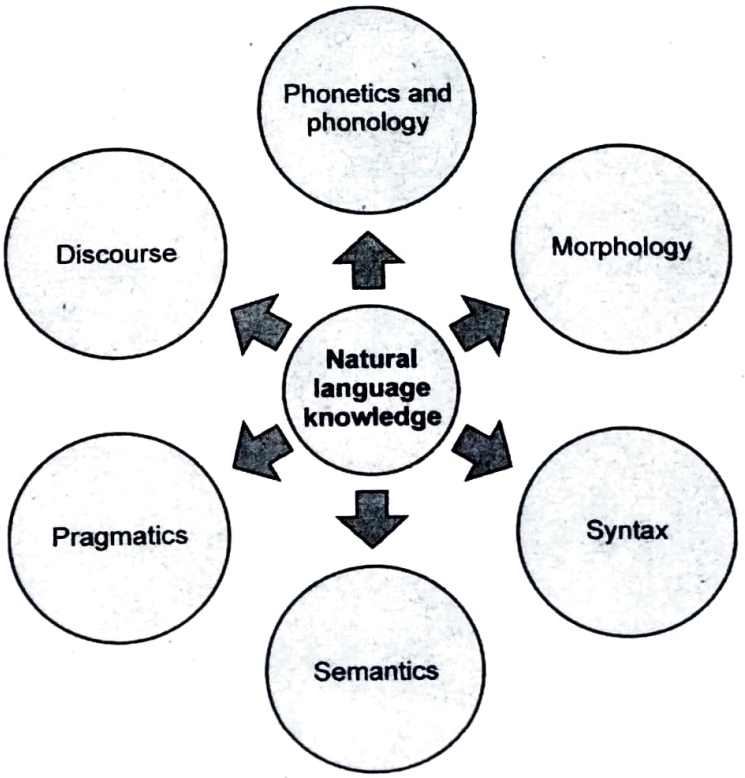


Fig 1.3.1 Categories of language knowledge

- The knowledge of how these components combine to form larger meanings, is referred as compositional semantics.
- The appropriate use of implied, indicative and indirect language comes under the heading of pragmatics.
- In addition to written language processing, spoken language processing tasks require additional computational knowledge about phonetics and phonology.
- Correctly structuring the words in conversations requires knowledge of discourse conventions. (See Fig 1.3.1 on previous page)
- These challenges in NLP frequently involve speech recognition, natural language understanding and natural-language generation.

Review Question

1. *Compare programming languages Vs natural languages.*

1.4 Are Natural Languages Regular ?

Natural languages are very ambiguous and many times, the meaning of words change with reference to the context and domain in which it is used. Hence natural language words have different levels of ambiguity like -

- **Lexical ambiguity** - It is at very primitive level such as word-level. For example, treating the word "board" as a noun or verb ?
- **Syntax level ambiguity** - A sentence can be parsed in different ways. For example, "He lifted the beetle with a red cap." - Did he use a cap to lift the beetle or he did lifted a beetle that had a red cap ?
- **Referential ambiguity** - Referring to something using pronouns. For example, Tom went to John. He said, "I am tired." - Exactly who is tired ? Here, one input can mean different meanings and sometimes many inputs can mean the same thing.

1.5 Finite Automata for NLP

- Natural language processing is different form that of formal languages or programming languages. This is primarily because,
  - Formal languages/ programming languages, can be fully specified.
  - All the reserved words in formal language can be defined and identified vice a versa.
- However, it is not possible with natural language. Natural languages are not designed; they evolve continually and therefore there is no formal specification.

- Despite this, regular expressions are used in various processing in NLP, e.g. phonology, morphology, text analysis, information extraction and speech recognition.
- In theory of computation, we have studied the relation between Finite State Automata (FSA0), Regular Expression(RE) and Regular language as shown below :

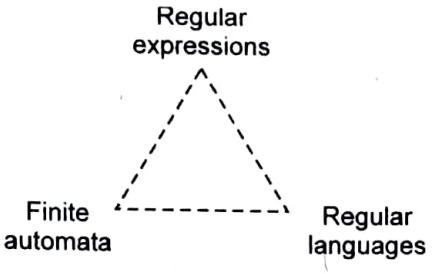
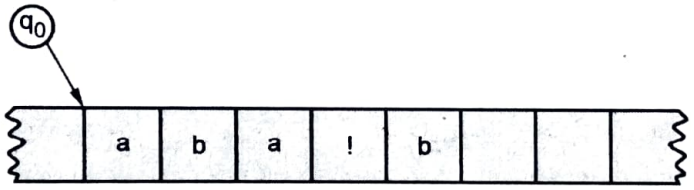


Fig. 1.5.1 Relation between regular expressions, finite automata and regular languages

- An FSA can be used for recognizing (we also say accepting) strings in the following way. First, think of the input as being written on a long tape broken up into cells, with one symbol written in each cell of the tape, as in below :



1.5.2 A tape with cells

FSA for sheep talk

- Let's begin with the 'sheep language'. The sheep language as can be defined as any string from the following (infinite) set :

baa!
baaa!
baaaa!
baaaaa!
baaaaaa!

Fig. 1.5.3

- The regular expression for this kind of 'sheep talk' is /baa+!/. Below Fig. 1.5.4 shows an automaton for modeling this regular expression.
- The automaton (i.e. machine, also called finite automaton, finite-state automaton, or FSA) recognizes a set of strings, in this case the strings characterizing sheep talk, in the same way that a regular expression does.

- The automaton is represented as a directed graph :
  - A finite set of vertices (also called nodes), vertices are represented with circles and arcs with arrows.
  - A set of directed links between pairs of vertices called **arcs.** It also has four TRANSITIONS, represented by arcs in the graph.
  - A set of STATES -
    - The automaton STATE has five states, which are represented by nodes in the graph.
    - State 0 is the START state which we represent by the incoming arrow.
    - State 4 is the FINAL STATE or ACCEPTING STATE, which we represent by the double circle.

Recognizing a string using FSA

- The FSA can be used for recognizing strings in the following way. The input was being written on a long tape. broken up into cells, with one symbol written in each cell of the tape, as in the below figure

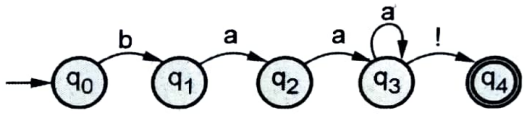


Fig. 1.5.4 A finite-state automation for talking sheep

- The machine starts in the start state (q_0) and iterates the following process :
- Check the next letter of the input. If it matches the symbol on an arc leaving the current state, then cross that arc, move to the next state and also advance one symbol in the input.
- If we are in the accepting state (q_4) when we run out of input, the machine has successfully recognized an instance (e.g. Sheeptalk - 'baa!' ,'baaa!','baaaa!'baaaaa!' and so on).
- If the machine never gets to the final state, either because it runs out of input, or it gets some input that doesn't match an arc (as in Fig. 1.5.5 above) or if it just happens to get stuck in some non-final state, the machine rejects or fails to accept an input.
- An automaton can also be represented with a state-transition table. As in the graph notation, the state-transition table represents the start state, the accepting states and what transitions leave each state with which symbols.

- Here's the state-transition table for the FSA

		Input		
State		b	a	!
0		1	0	0
1		0	2	0
2		0	3	0
3		0	3	4
4		0	0	0

Fig. 1.5.5 The state-transition table for the FSA

Key points about FSA

- The regular expression language is a powerful tool for pattern-matching.
- Basic operations in regular expressions include concatenation of symbols, disjunction of symbols ([], |, and .), counters (*, +, and {n,m}), anchors (^, $) and precedence operators ((,)).
- Any regular expression can be realized as a finite automaton. memory (\1 together with ()) is an advanced operation which is often considered part of regular expressions, but which cannot be realized as a finite automaton.
- An automaton implicitly defines a formal language as the set of strings the automaton accepts.
- An automaton can use any set of symbols for its vocabulary, including letters, words, or even graphic images.
- The behaviour of a deterministic automata (DFSA) is fully determined by the state it is in.
- A non-deterministic automata (NFSA) sometimes has to make a choice between multiple paths to take given the same current state and next input.
- Any NFSA can be converted to a DFSA.
- The order in which a NFSA chooses the next state to explore on the agenda defines its search strategy.
- The depth-first search or LIFO strategy corresponds to the agenda-as-stack; the breadth-first search or FIFO strategy corresponds to the agenda-as-queue.
- Any regular expression can be automatically compiled into an NFSA and hence into an FSA.

English Morphology

- The spectrum of natural languages is very wide. As per the linguistic science There are thousands of spoken languages in the world. These languages can be grouped together as members of a language family. There are three main language in the world :
  - Indo-European (Includes English)
  - Sino-Tibetan (Includes Chinese)
  - Afro-Asiatic (Includes Arabic)

Focus of our syllabus is restricted to English as a natural language hence, English morphology will be discussed further in this book.

- English morphology is the branch of grammar that investigates the internal structure of English words. Many words can be subdivided into smaller meaningful units called **morphemes.**
- Morphology is the study of the variable forms and functions of words, while syntax is concerned with the arrangement of words into phrases, clauses and sentences.
- According to the classical approach in linguistics, words are formed of morphemes, which are the minimal (i.e. non-decomposable) linguistic units that carry meaning.
- Many language processing applications need to extract the information encoded in the words.
  - Parsers which analyse sentence structure need to know/check agreement between :
    - Subjects and verbs
    - Adjectives and nouns
  - Information retrieval systems benefit from know what the stem of a word is ?
  - Machine translation systems need to analyse words to their components and generate words with specific features in the target language.
- According to the classical approach in linguistics, words are formed of morphemes, which are the minimal (i.e. non-decomposable) linguistic units that carry meaning.
- A morpheme is the smallest meaningful unit in a language.
- A morpheme is not identical to a word. The main difference between them is that a morpheme sometimes does not stand alone, but a word, by definition, always stands alone.

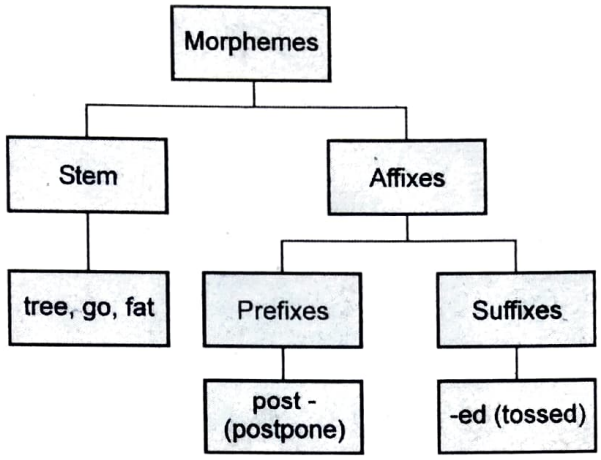


Fig. 1.5.6 Structure of english morphology

- When a morpheme stands by itself, it is considered as a root because it has a meaning of its own (such as the morpheme dog).
- When one morpheme depends on another morpheme to express an idea, it is an affix because it has a grammatical function (such as the - s in dogs to indicate that it is plural).
- Natural languages use different techniques by which morphs and morphemes are combined into word forms. The simplest morphological process concatenates morphs one by one.

Basics of English Morphology

English morphology has the following terminologies used. Fig. 1.5.7 shows a pictorial example of morphemes for the word unpleasantness :

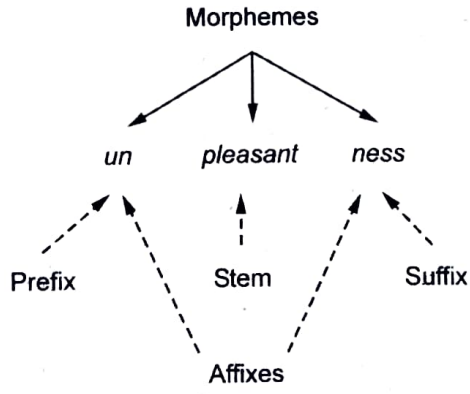


Fig. 1.5.7 Morphological example for word unpleasantness

- Morpheme - minimal meaning-bearing unit in a language.
- Stems - central meaning-bearing morpheme of the word.
- Affixes - supply "additional" meanings :

- Prefixes - precede the stem
- Suffixes - follow the stem
- Circumfixes - precede and follow the stem
- Infixes - inserted inside the stem.

- Non-concatenative - morphemes are intermingled rather than concatenated.
- Root-and-pattern morphology - e.g. condition.

Lexical morphology

- Lexical morphology is the branch of morphology that deals with the lexicon, which, morphologically conceived, is the collection of lexemes in a language. As such, it concerns itself primarily with word formation : Derivation and compounding.
- There are three principal approaches to morphology and each tries to capture the distinctions above in different ways :
  - Morpheme-based morphology, which makes use of an item-and-arrangement approach.
  - Lexeme-based morphology, which normally makes use of an item-and-process approach.
  - Word-based morphology, which normally makes use of a word-and-paradigm approach.

Morpheme-based morphology

- Morpheme-based morphology is a tree based representation of language morphemes, represented independently.
- In morpheme-based morphology, word forms are analyzed as arrangements of morphemes. A morpheme is defined as the **minimal meaningful unit of a language.**
- In a word such as independently, the morphemes are said to be in-, de-, pend, -ent, and -ly; pend is the (bound) root and the other morphemes are, in this case, derivational affixes.
- In words such as dogs, dog is the root and the -s is an inflectional morpheme.
- In its simplest and most naïve form, this way of analyzing word forms, called "item-and-arrangement", treats words as if they were made of morphemes put after each other ("concatenated") like beads on a strings.

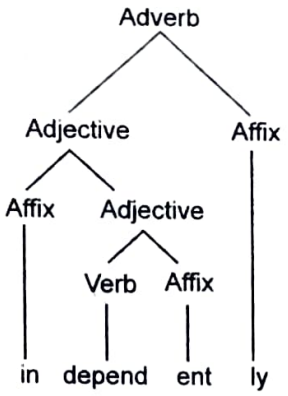


Fig. 1.5.8

Lexeme-based morphology

- Lexeme-based morphology usually takes what is called an **item-and-process approach.** Instead of analyzing a word form as a set of morphemes arranged in sequence, a word form is said to be the result of applying rules that alter a word-form or stem in order to produce a new one.
- An inflectional rule takes a stem, changes it as is required by the rule and outputs a word form ; a derivational rule takes a stem, changes it as per its own requirements and outputs a derived stem; a compounding rule takes word forms and similarly outputs a compound stem.

Word-based morphology

- Word-based morphology is (usually) a word-and-paradigm approach.
- The theory takes paradigms as a central notion. Instead of stating rules to combine morphemes into word forms or to generate word forms from stems, word-based morphology states generalizations that hold between the forms of inflectional paradigms.
- The major point behind this approach is that many such generalizations are hard to state with either of the other approaches. Word-and-paradigm approaches are also well-suited to capturing purely morphological phenomena, such as morphomes.
- The approaches treat these as whole words that are related to each other by analogical rules. Words can be categorized based on the pattern they fit into. This applies both to existing words and to new ones. Application of a pattern different from the one that has been used historically can give rise to a new word, such as older replacing elder (where older follows the normal pattern of adjectival superlatives) and cows replacing kine (where cows fits the regular pattern of plural formation).

Types of Morphemes

Morphemes, the smallest meaning-bearing units, can be divided into two types -

- Stems
- Word order.

Stems

It is the core meaningful unit of a word. We can also say that it is the root of the word. For example, in the word foxes, the stem is fox.

- **Affixes** - As the name suggests, they add some additional meaning and grammatical functions to the words. For example, in the word foxes, the affix is - es.

Further, affixes can also be divided into following four types -

  - **Prefixes** - As the name suggests, prefixes precede the stem. For example, in the word unbuckle, un is the prefix.
  - **Suffixes** - As the name suggests, suffixes follow the stem. For example, in the word cats, -s is the suffix.
  - **Infixes** - As the name suggests, infixes are inserted inside the stem. For example, the word cupful, can be pluralized as cupsful by using -s as the infix.
  - **Circumfixes** - They precede and follow the stem. There are very few examples of circumfixes in the English language. A very common example is 'A-ing' where we can use -A precede and -ing follows the stem.

- Prefixes and suffixes are often called **concatenative morphology** since a word is composed of a number of morphemes concatenated together.
- A number of languages have extensive non-concatenative morphology, in which morphemes are combined in more complex ways.
- Another kind of non-concatenative morphology is called **templatic morphology** or rootand-pattern morphology. This is very common in Arabic, Hebrew and other Semitic languages.
- Languages that tend to string affixes together like Turkish does are called **agglutinative languages.**

English Morphology for NLP

- There are two broad (and partially overlapping) classes of ways to form words from morphemes : Inflection and derivation.

Inflection	Deviation
• Inflection is the combination of a word stem with a grammatical morpheme, usually resulting in a word of the same class as the original stem and usually filling some syntactic function like agreement.	• Derivation is the combination of a word stem with a grammatical morpheme, usually resulting in a word of a different class, often with a meaning hard to predict exactly.
• For example, English has the inflectional morpheme-s for marking the plural on nouns and the inflectional morpheme-ed for marking the past tense on verbs.	• For example, the verb computerize can take the derivational suffix-ation to produce the noun computerization.

Inflectional Morphology

- English has a relatively simple inflectional system; only nouns, verbs and sometimes adjectives can be inflected and the number of possible inflectional affixes is quite small.
- Noun inflection / Nominal inflection.
  - English nouns have only two kinds of inflection : An affix that marks plural and an affix that marks possessive. For example, many (but not all) English nouns can either appear in the bare stem or singular form, or take a plural suffix.
  - Here are examples of the regular plural suffix -s, the alternative spelling -es, and irregular plurals.

	Regular nouns		Irregular nouns	
Singular	cat	thrush	mouse	ox
Plural	cats	thrushes	mice	oxen

  - While the regular plural is spelled -s after most nouns, it is spelled -es after words ending in -s (ibis/ibises) , -z, (waltz/waltzes) -sh, (thrush/thrushes)-ch, (finch/finches) and sometimes -x (box/boxes). Nouns ending in -y preceded by a consonant change the -y to -i (butterfly/butterflies).
  - The possessive suffix is realized by apostrophe + -s for regular singular nouns (llama's) and plural nouns not ending in -s (children's) and often by alone apostrophe after regular plural nouns (llamas') and some names ending in -s or -z (Euripides' comedies).

- **Verb Inflection**
  - English verbal inflection is more complicated than nominal / noun inflection. English has three kinds of verbs;
    1. Main verbs, (eat, sleep, impeach)
    2. Modal verbs (can, will, should)
    3. Primary verbs (be, have, do).
  - Here the focus is with the main and primary verbs, because they have inflectional endings. This class of verbs have the same endings marking the same functions. These regular verbs (e.g. walk or inspect).
  - These verbs are called regular because just by knowing the stem we can predict the other forms, by adding one of three predictable endings and making some regular spelling changes.
  - These regular verbs and forms are significant in the morphology of English first because they cover a majority of the verbs and second because the regular class is productive.
  - The main and primary verbs have four morphological forms, as below :

Morphological form classes	Regularly inflected verbs			
Stem	walk	merge	try	map
- s form	walks	merges	tries	maps
- ing participle	walking	merging	trying	mapping
Past form or – ed participle	walked	merged	tried	mapped

  - The -ed participle is used in the perfect construction (He's eaten lunch already) or the passive construction (The verdict was overturned yesterday).

- **Irregular verbs**
  - The irregular verbs are those that have some more or less idiosyncratic forms of inflection.
  - Irregular verbs in English often have five different forms, but can have as many as eight (e.g. the verb be) or as few as three (e.g.cut or hit).
  - While constituting a much smaller class of verbs estimate that there are only about 250 irregular verbs, not counting auxiliaries.
  - This class includes most of the very frequent verbs of the language. The table below shows some sample irregular forms.

- An irregular verb can inflect in the past form (also called the preterite) by changing its vowel (eat/ate) or its vowel and some consonants (catch/caught), or with no ending at all (cut/cut).

Morphological form classes	Irregularly inflected verbs		
Stem	eat	catch	cut
- s form	eats	catches	cuts
- ing participle	eating	catching	cutting
Past form	ate	caught	cut
Past form or – ed participle	eaten	caught	cut

- **Gerund use of verbs**

The -ing participle is used when the verb is treated as a noun; this particular kind of nominal use of a verb is called a gerund use.

e.g. Dicing is a form of cutting salad.

- The English verbal system is much simpler than for example the European Spanish system, which has as many as fifty distinct verb forms for each regular verb.

Derivational Morphology

- English inflection is relatively simple compared to other languages, however derivation in English is quite complex.
- Derivation is the combination of a word stem with a grammatical morpheme, usually resulting in a word of a different class, often with a meaning hard to predict exactly.
- A very common kind of derivation in English is the formation of new nouns, often from verbs or adjectives. This process is called **nominalization.**
- For example, the suffix - ation produces nouns from verbs ending often in the suffix -ize (computerize !computerization). Here are examples of some particularly productive English nominalizing suffixes.

Suffix	Base verb/Adjective	Derived noun
- ation	computerize (V)	computerization
- ee	appoint (V)	appointee
- er	kill	killer
- ness	fuzzy (A)	fuzziness

- Adjectives can also be derived from nouns and verbs. Here are examples of a few suffixes deriving adjectives from nouns or verbs.

Suffix	Base noun/Verb	Derived adjective
- al	computation (N)	computational
- able	embrace (V)	embraceable
- less	clue (N)	clueless

- Derivation in English is more complex than inflection for the following reasons.
  - It is generally less productive; even a nominalizing suffix like -ation, which can be added to almost any verb ending in -ize, cannot be added to absolutely every verb.
  - There are subtle and complex meaning differences among nominalizing suffixes. E.g. Sincerity has a subtle difference in meaning from sincereness.

Review Questions

1. *How a string is recognized using FSA.*
2. *Write a note on english morphology.*
3. *Explain types of morphemes.*
4. *Explain inflection and derivational morphology.*
5. *What are irregular verbs.*

1.6 Stages of NLP

- Although natural language processing tasks are closely intertwined, they can be subdivided into categories for convenience. A coarse division is given below :
- The following is a list of some of the most commonly researched phases in natural language processing. Some of these phases have direct real-world applications, while others more commonly serve as subtasks that are used to aid in solving larger tasks.
- Stages of NLP

1. Lexical Analysis and Morphological Analysis

The first phase of NLP is the Lexical Analysis. 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.

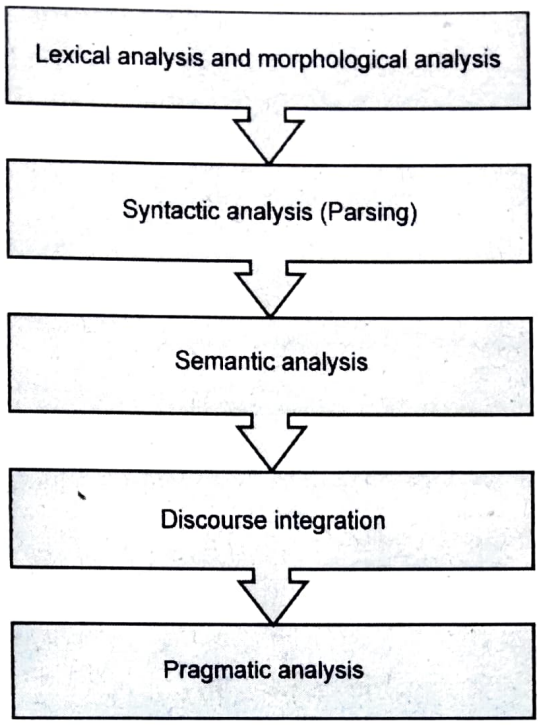


Fig. 1.6.1 Phases of NLP

2. Syntactic Analysis (Parsing)

Syntactic Analysis is used to check grammar, word arrangements and shows the relationship among the words.

Example : Dog eats bread

In the real world, bread eats dog, does not make any sense, so this sentence is rejected by the syntactic analyzer.

3. Semantic Analysis

Semantic analysis is concerned with the meaning representation. It mainly focuses on the literal meaning of words, phrases and sentences.

4. Discourse Integration

Discourse Integration depends upon the sentences that proceeds it and also invokes the meaning of the sentences that follow it.

5. Pragmatic Analysis

Pragmatic is the fifth and last phase of NLP. It helps you to discover the intended effect by applying a set of rules that characterize cooperative dialogues.

For Example : "Open the door" is interpreted as a request instead of an order.

Review Question

1. *Explain stages of NLP.*

1.7 Challenges and Issues (Open Problems) in NLP

Advantages of using NLP :

Using NLP systems at the workplace or in day to day life, helps users extensively in reducing efforts on mundane tasks. NLP provides a good basis for human interaction to automate several processes. Here is a list of some of the advantages of using NLP components in the system.

- NLP processes help computers communicate with a human in their language and scales other language-related tasks.
- It is easy to implement in a limited scope.
- It is very time efficient.
- NLP helps users to ask questions about any subject and get a direct response within seconds.
- The NLP system provides answers to the questions in natural language.
- NLP improves the efficiency of documentation processes, accuracy of documentation and identifying the information from large databases.
- Using NLP has advantages (less costly than employing human staff, provides quicker customer service response times and is easy to implement)
- The NLP system offers exact answers to the questions, no unnecessary or unwanted information.
- The accuracy of the answer increases with the amount of relevant information provided in the questions.
- Structuring a high unstructured data source.
- Users can ask questions about any subject and get a direct response in seconds.
- Using an NLP system is less costly than hiring a person. A person can take two or three times longer than a machine to execute the tasks mentioned.
- NLP allows us to perform more language-based data compared to a human being without fatigue and in an unbiased and consistent way.
- It is a faster customer service response time.

Challenges and Issues of NLP

- NLP inherits the limitations due to differences in the scope of natural language and the ability of a computer within the defined scope. Also a natural language continuously adopts new words and keeps growing in terms of style and context. This puts some limitations on the systems that use NLP. Some of these disadvantages are listed below.
- NLP systems or subcomponents can work only within the domain/ scope for which these are designed. Hence these systems fail to recognize context outside the scope.
- As a result of this NLP systems are unable adapt to new domains and have a limited function.
- NLP is built for a single and specific task.
- The NLP system doesn't have a user interface that lacks features that allow users to further interact with the system.
- If it is necessary to develop a model with a new one without using a pre-trained model, it can take a week to achieve a good performance depending on the amount of data.
- The system is built for a single and specific task only, it is unable to adapt to new domains and problems because of limited functions.
- In complex query language, the system may not be able to provide the correct answer to a question that is poorly worded or ambiguous.
- NLP is not 100 % reliable, It is never 100 % dependable. There is the possibility of error in its prediction and results.

Review Questions

1. *Explain advantages of NLP.*
2. *Explain challenges and issues of NLP.*

1.8 Basics of Text Processing

NLP uses language processing pipelines to read, decipher and understand human languages. These pipelines consist of some prime processes. Refer Fig. 1.8.1 for the NLP pipeline stages That breaks the whole voice or text into small chunks, reconstructs it, analyses and processes it to bring us the most relevant data from the Search Engine Result Page.

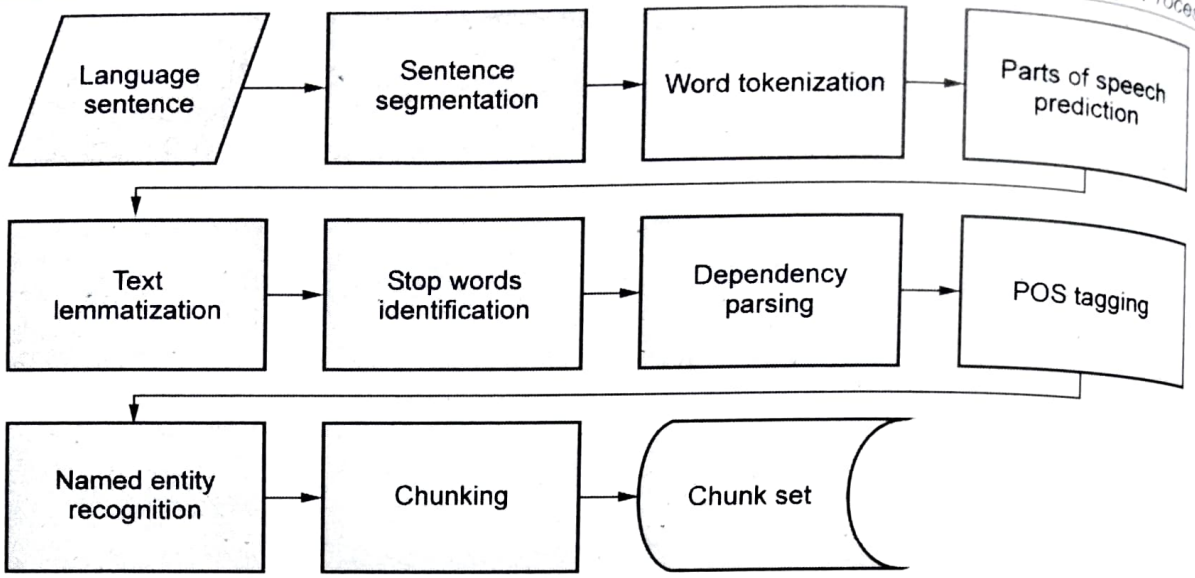


Fig. 1.8.1 NLP pipeline and stages

- **Sentence Segmentation**
  - When a language paragraph is to be processed, the best way to proceed is to go with one sentence at a time. It reduces the complexity and simplifies the process, even getting you the most accurate results. Computers never understand language the way humans do, but they can always do a lot if you approach them in the right way.
  - For example, consider the above paragraph. Then, your next step would be breaking the paragraph into single sentences.
- **Word Tokenization**
  - Tokenization is the process of breaking a phrase, sentence, paragraph, or entire documents into the smallest unit, such as individual words or terms. And each of these small units is known as **tokens.**
  - These tokens could be words, numbers, or punctuation marks. Based on the word's boundary - ending point of the word. Or the beginning of the next word. It is also the first step for stemming and lemmatization.
  - This process is crucial because the meaning of the word gets easily interpreted through analysing the words present in the text.
  - Let's take an example :

    That dog is a husky breed.

    When you tokenize the whole sentence, the answer you get is ['That', 'dog', 'is', a, 'husky', 'breed'].

There are numerous ways you can do this, but we can use this tokenized form to :

1. Count the number of words in the sentence.
2. Measure the frequency of the repeated words.

- **Parts of Speech (PoS)Prediction**
  - In a part of the speech, we have to consider each token. And then, try to figure out different parts of the speech - whether the tokens belong to nouns, pronouns, verbs, adjectives and so on. All these help to know which sentence we all are talking about.
  - Here are some keywords used in PoS
    - **Corpus :** Body of text, singular. Corpora are the plural of this.
    - **Lexicon :** Words and their meanings.
    - **Token :** Each "entity" that is a part of whatever was split up based on rules.
- **Text Lemmatization**
  - English is also one of the languages where we can use various forms of base words. When working on the computer, it can understand that these words are used for the same concepts when there are multiple words in the sentences having the same base words. The process is what we call lemmatization in NLP.
  - It goes to the root level to find out the base form of all the available words. They have ordinary rules to handle the words and most of us are unaware of them.
- **Identifying Stop Words**
  - When you finish the lemmatization, the next step is to identify each word in the sentence. English has a lot of filler words that don't add any meaning but weakens the sentence. It's always better to omit them because they appear more frequently in the sentence.
  - Most data scientists remove these words before running into further analysis. The basic algorithms to identify the stop words by checking a list of known stop words as there is no standard rule for stop words.

- **Dependency Parsing**
  - Parsing is divided into three prime categories further. And each class is different from the others. They are part of speech tagging, dependency parsing, and constituency phrasing.
  - The dependency phrasing case : Analyses the grammatical structure of the sentence. Based on the dependencies in the words of the sentences. Whereas in constituency parsing : The sentence breakdown into sub-phrases. And these belong to a specific category like Noun Phrase (NP) and Verb Phrase (VP).
- **POS tags**
  - POS stands for parts of speech, which includes nouns, verbs, adverbs and adjectives. It indicates that how a word functions with its meaning as well as grammatically within the sentences. A word has one or more parts of speech based on the context in which it is used.
- **Named Entity Recognition (NER)**
  - It is a process of detecting the named entity such as person name, movie name, organization name or location.
- **Chunking**
  - Chunking is used to collect the individual pieces of information and grouping them into bigger pieces of sentences.

Review Question

1. *Explain basics of text processing.*

❑❑❑

Unit II

2 Language Syntax and Semantics

Syllabus

Morphological Analysis : *What is Morphology ? Types of Morphemes, Inflectional morphology & Derivational morphology, Morphological parsing with Finite State Transducers (FST)*

Syntactic Analysis : *Syntactic Representations of Natural Language, Parsing Algorithms, Probabilistic context-free grammars, and Statistical parsing*

Semantic Analysis : *Lexical Semantic, Relations among lexemes & their senses - Homonymy, Polysemy, Synonymy, Hyponymy, WordNet, Word Sense Disambiguation (WSD), Dictionary based approach, Latent Semantic Analysis*

Contents

2.1 Morphological Analysis

What is Morphology ?

- The term morphological parsing is related to the parsing of morphemes. We can define morphological parsing as the problem of recognizing that a word breaks down into smaller meaningful units called morphemes producing some sort of linguistic structure for it. For example, we can break the word foxes into two, fox and - es. We can see that the word foxes, is made up of two morphemes, one is fox and other is - es.
- In other sense, we can say that morphology is the study of -
  - The formation of words.
  - The origin of the words.
  - Grammatical forms of the words.
  - Use of prefixes and suffixes in the formation of words.
  - How parts-of-speech (PoS) of a language are formed.
- Morphological parsing yields information that is useful in many NLP applications. In parsing, e.g., it helps to know the agreement features of words. Similarly, grammar checkers need to know agreement information to detect such mistakes.
- But morphological information also helps spell checkers to decide whether something is a possible word or not, and in information retrieval it is used to search not only cats, if that's the user's input, but also for cat.
- Let's take a simple example of Morphological parsing in NLP. This example is of parsing just the productive nominal plural (-s) and the verbal progressive (-ing). The goal is to take input forms like those in the first column below and produce output forms like those in the second column.

Input	Morphological Parsed Output
cats	cat + N + PL
cat	cat + N + SG
cities	city + N + PL
geese	goose + N + PL
goose	(goose + N + SG) or (goose + V)
gooses	goose + V + 3 SG
merging	merge + V + PRES-PART
caught	(catch + V + PAST-PART) or (catch + V + PAST)

- The second column contains the stem of each word as well as assorted morphological features. These features specify additional information about the stem.
- Some of the commonly used features are,
  1. N - Noun
  2. PL - Plural
  3. SG - Singular
  4. V - Verb
  5. PAST - Past tense
  6. PRES - Present tense
  7. FUTU - Future tense
  8. PART - Participle

Morphological Parser

- Building a morphological parser for a given input language is very important step in NLP. In order to build a morphological parser, following information is needed -
  1. **A lexicon :** The list of stems and affixes, together with basic information LEXICON about them (whether a stem is a Noun stem or a Verb stem, etc).
  2. **Morphotactics :** The model of morpheme ordering that explains which classes of morphemes can follow other classes of morphemes inside a word. For example, the rule that the English plural morpheme follows the noun rather than preceding it.
  3. **Orthographic rules :** These spelling rules are used to model the changes that occur in a word, usually when two morphemes combine (e.g. Cherry + -s to cherries rather than cherrys).

Morphological recognition

- Finite State Automata (FSA)s can be used to solve the problem of morphological recognition; that is, of determining whether an input string of letters makes up a legitimate English word or not.
- It is done by expanding each arc (e.g. the reg-noun-stem arc) with all the morphemes that make up the set of reg-noun-stem. The resulting FSA can then be defined at the level of the individual letter. (Refer below Fig. 2.1.1)
- Morphological recognition is performed by taking the morphotactic FSAs and plugging in each 'sub-lexicon' into the FSA. i.e. each arc (e.g. the reg-noun-stem arc)

is expanded with all the morphemes that make up the set of reg-noun-stem. The resulting FSA can then be defined at the level of the individual letter.

- Below is an FSA for a fragment of English derivational morphology.

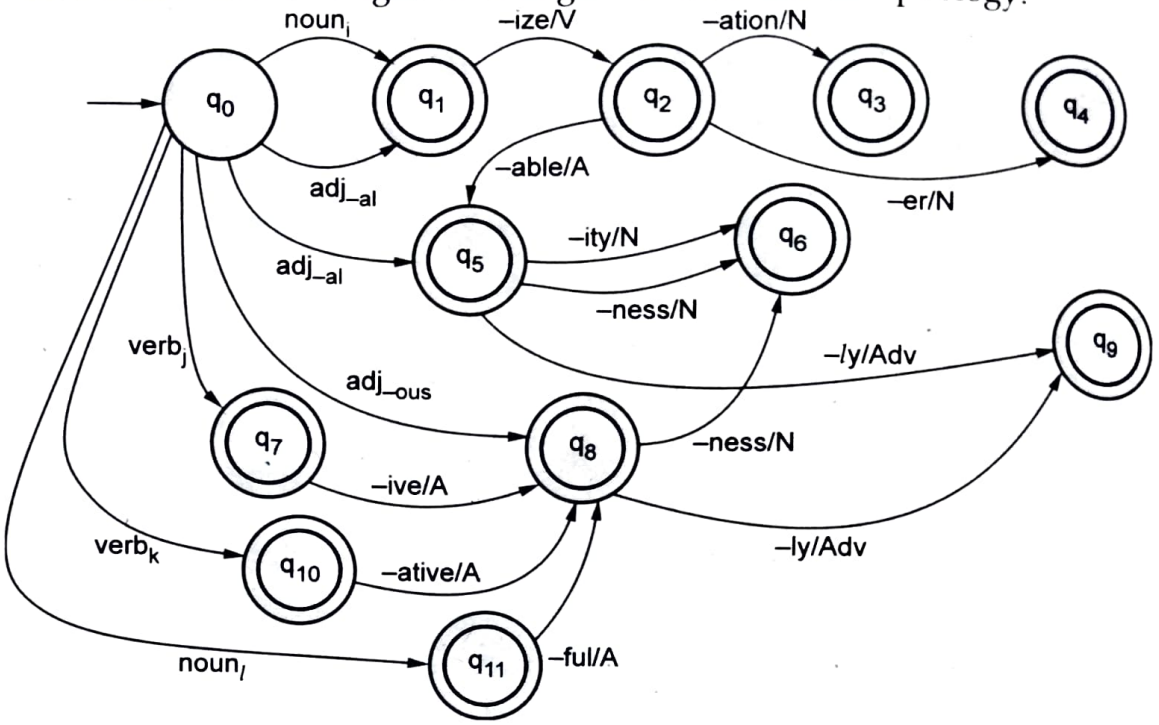


Fig. 2.1.1 : For a fragment of english derivational morphology

Morphological Parsing with Finite-State Transducers

- Finite-state machines have been used in various domains of natural language processing. Koskenniemi (1983), proposed the two-level morphology for parsing.
- Two level morphology represents a word as a correspondence between a lexical level and surface level.
  - The lexical level represents a simple concatenation of morphemes making up a word.
  - The surface level represents the actual spelling of the final word.
- Morphological parsing is implemented by building mapping rules that map letter sequences.

  Example - The word like cats on the surface level is mapped into morpheme and features sequences likecat +N +PL on the lexical level.
- The below Fig. 2.1.2 shows two levels for the word cats. The lexical level has the stem for a word, followed by the morphological information +N +PL which tells us that cats is a plural noun.

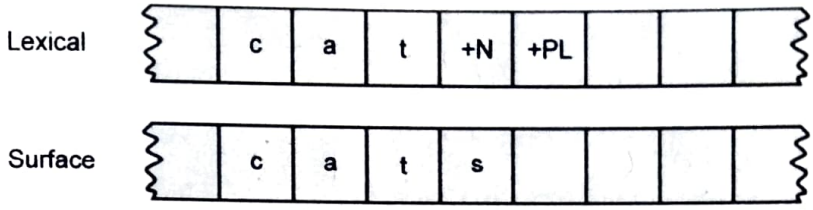


Fig. 2.1.2 : Example of lexical and surface tape

- Two-level morphology is based on three ideas :
  1. Rules are symbol-to-symbol constraints that are applied in parallel, not sequentially like rewrite rules.
  2. The constraints can refer to the lexical context, to the surface context or to both contexts at the same time.
  3. Lexical lookup and morphological analysis are performed in tandem.

Finite-State Transducer or FST

- The automaton used for performing the mapping between lexicon level and surface level is called finite-state transducer or FST.
- A transducer maps between one set of symbols to the another.
- A finite-state transducer does this via a finite automaton.
- An FST can be visualized as a two-tape automaton which recognizes or generates pairs of strings.
- The FST has a more general function than an FSA; where an FSA defines a formal language by defining a set of strings, an FST defines a relation between sets of strings.
- This leads to another perspective towards an FST; as a machine that reads one string and generates another.
- FSTs have four-fold way of thinking about transducers as :
  1. **FST as recognizer :** A transducer that takes a pair of strings as input and outputs accept if the string-pair is in the string-pair language a reject if it is not.
  2. **FST as generator :** A machine that outputs pairs of strings of the language. Thus the output is a yes or no and a pair of output strings.
  3. **FST as translator :** A machine that reads a string and outputs another string.
  4. **FST as set relater :** A machine that computes relations between sets.

Definition of FST

There are several ways of defining an FST. The below definition is based on the Mealy machine, a basis for computational mathematics.

Let

Q : A finite set of N states $q_0, q_1, \ldots q_N$

Σ : A finite alphabet of complex symbols. Each complex symbol is composed of an input-output pair i : o; one symbol i from an input alphabet I and one symbol o from an alphabet O, thus $\Sigma \subseteq I \times O$ and O may each also include the epsilon symbol ε.

q_0 : The start state

F : The set of final states, $F \subseteq Q$.

δ (q, i,: 0) : The transition function or transition matrix between states. Given a state $q \in Q$ and complex symbol i : $0 \in \Sigma$, δ (q, i : 0) returns a new state $q' \in Q$. δ is thus a relation from $Q \times \Sigma$ to Q;

Example

- If an FSA accepts a language stated over a finite alphabet of single symbols, such as the alphabet of the our sheep language.

$$\Sigma = \{b, a, !\}$$

- Then an FST accepts a language stated over *pairs* of symbols, as in :

$$\Sigma = \{a : a, b : b, ! : !, a : !, a : \varepsilon, \varepsilon : !\}$$

- In two-level morphology, the pairs of symbols in Σ are also called **feasible pairs**.

Comparing FSA and FST

- Where FSAs are isomorphic to regular languages, FSTs are isomorphic to regular relations.
- FSTs and regular relations are closed under union, although in general they are not closed under difference, complementation and intersection.
- Besides union, FSTs have two additional closure properties that turn out to be extremely useful :
  - **Inversion :** The inversion of a transducer T (T^{-1}) simply switches the input and output labels. Thus if T maps from the input alphabet I to the output O, T^{-1} maps from O to I.
  - **Composition :** If T_1 is a transducer from I_1 to O_1 and T_2 a transducer from I_2 to O_2, then T_1 to T_2 maps from I_1 to O_2.

- Here is a transducer for English nominal number inflection T_{num}. Since both q_1 and q_2 are accepting states, regular nouns can have the plural suffix or not. The morpheme-boundary symbol ^ and word-boundary marker # are discussed below.

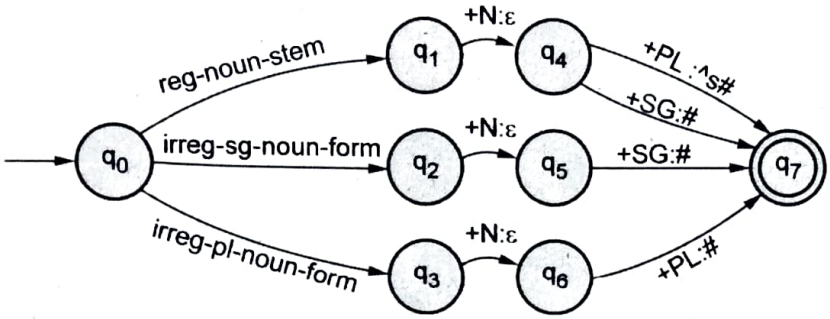


Fig. 2.1.3 A transducer for english nominal number inflection

Orthographic Rules and Finite-State Transducers

- English language requires spelling changes at morpheme boundaries by introducing spelling rules (or orthographic rules). These rules are listed in below table

Name	Description of Rule	Example
Consonant doubling	1-Letter constant doubled before-*ING/-ED*	beg/begging
E deletion	Silent e dropped before *-ING* and *-ED*	make/making
E insertion	e added after *-S, -Z,-X, -SH* before *-S*	watch/watches
Y replacement	*-y* changes to *-IE* before *-S*, *-I* before *-ED*	try/tries
K insertion	verb ending with vowel + *-C* add *-K*	panic/panicked

Generating or parsing with FST lexicon and rules

- A cascade is a set of transducers in series, in which the output from one transducer acts as the input to another transducer; cascades can be of arbitrary depth and each level might be built out of many individual transducers. The cascade can be run top-down to generate a string or bottom-up to parse it.

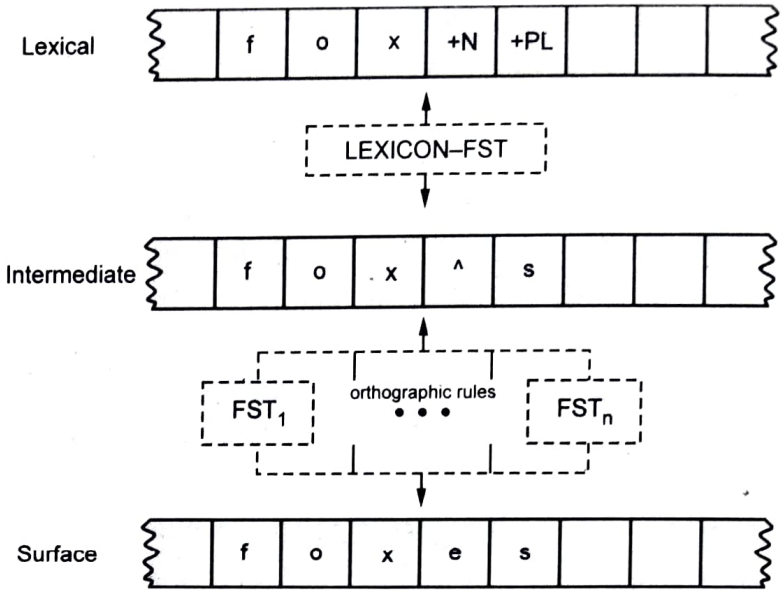


Fig. 2.1.4 : Generating or parsing with FST lexicon and rules

- The architecture in below Fig. 2.1.4 is a two-level cascade of transducers that is used for generating or parsing with FST lexicon and rules.
- Parsing can be slightly more complicated than generation, because of the problem of ambiguity. Disambiguating requires some external evidence such as the surrounding words.
- The power of finite-state transducers is that the exact same cascade with the same state sequences is used when the machine is generating the surface tape from the lexical tape or when it is parsing the lexical tape from the surface tape.
- Transducers in parallel can be combined by automaton intersection. The automaton intersection algorithm just takes the cartesian product of the states.

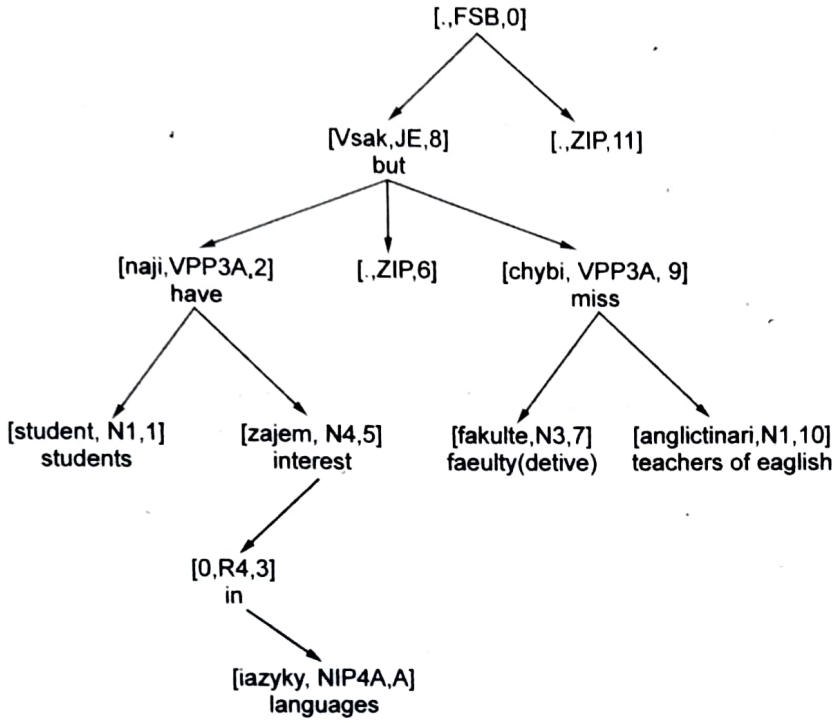
Review Questions

1. *What is morphology ?*
2. *Explain morphological passing with finite state transduces.*
3. *What is Finite State Transducer or FST ?*

2.2 Syntactic Analysis : Syntactic Representations of Natural Language

- Dependency graphs connects head of a phrase to its dependents.
- According to definition of Co NLL, in dependency graph nodes are words of input sentence and arcs are binary relations from head to dependent.
- It labelled dependency parsing a label is assigned to each dependency relation between head and dependent word.

- The example dependency graph for Czech sentence from Prague Dependency Treebank is shown in Fig. 2.2.1.



The students are interested in languages, but the faculty is missing teachers of English

Fig. 2.2.1 : An example of a dependency graph syntax analysis for a Czech sentence taken from Prague Dependency Treebank. Each node in the graph is a word, its part of speech, and the not of the word in the sentence, for example [fakulte, N3, 7] is the seventh wood in the sentence with F tag N3, which also tells us that the word has dative case. The node [#, ZSB,0] is the root node of dependency tree. The English equivalent is provided for each node

- It is observed that dependency analysis make minimal assumption about syntactic structure for avoiding annotation of hidden structure like empty elements to represent missing arguments of predicates.

Projectivity in dependency analysis

- Projectivity is the constraint on syntactic analysis due to effect of linear order of words on dependencies between words.
- The example shown in Fig. 2.2.2 shows the english sentence with the requirements of crossing dependencies.

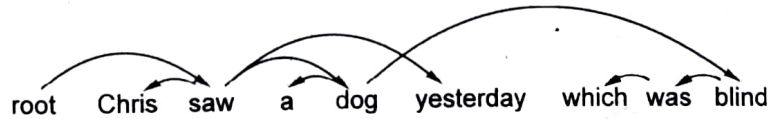


Fig. 2.2.2 : An unlabeled nonprojective dependency tree with a crossing dependency

Syntax analysis using phrase structure tree

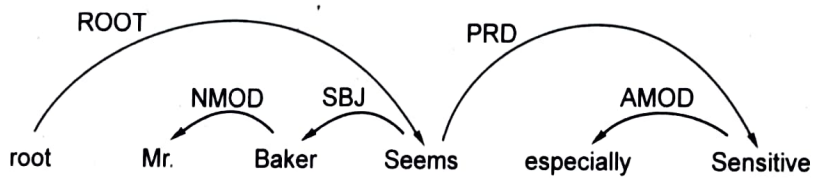
- Phrase structure syntax analysis is based on the concept that the sentences can be divided in constituents and larger constituents are formed by merging smaller ones.
- **For ex. :** Consider the sentence from penn Treebank "Mr. Baker especially sensitive" The predicate argument structure can be written as,

```
( S (NP – SBJ (NNP Mr. )
              (NNP Baker))
    (VP (VBZ seems)
        (ADJP – PRD(RB especially)
                ( JJ sensitive ) ) ) )
```

Predicate - argument structure :

Seems ((especially (sensitive)) ((Mr. Banker))

- The subject is marked with - SBJ marker and predicate with - PRD marker.
- The dependency tree can be drawn as,



Review Questions

1. *Explain dependency based graph syntax analysis.*
2. *Explain syntax analysis using phrase structure tree.*

2.3 Parsing Algorithms

Shift reduce parser

- In parsing for a given input string we need to do right most derivation of grammer.
- In shift reduce parsing the concept of pushdown automaton (PDA) is used.
- PDA is an automaton that uses stack.
- The shift reduce parser has two steps
  1. **Shift step :** In this the input stream is advanced by one symbol. The shifted symbol is considered as single node parse tree.
  2. **Reduce step :** It applies completed grammar rule to recent parse trees and combine them together as one tree with new root symbol.

- The working of shift reduce parsing algorithm is shown in Fig. 2.3.1.

Parse Tree	Stack	Input	Action
		a and b or c	Init
a	a	and b or c	Shift a
(N a)	N	and b or c	Reduce N → a
N(a) and	N and	b or c	Shift and
(N a) and b	N and b	or c	Shift b
(N a) and (N b)	N and N	or c	reduce N → b
(N (N a) and (N b))	N	or c	reduce N → a
(N (N a) and (N b)) or	N or	c	shift or
(N (N a) and (N b)) or c	N or c		Shift c
(N (N a) and (N b)) or (N c)	N or N		Reduce N → c
(N (N (N a) and (N b)) or (N c))	N		Reduce N → N or N
(N (N (N a) and (N b)) or (N c))	N		Accept!

Fig. 2.3.1 : The individual steps of the shift-reduced parsing algorithm for the input a and b or c for the grammar G defined at the beginning of this section

- The algorithm is defined as Fig. 2.3.2.

1. Start with an empty stack and the buffer containing the input string.
2. Exit with success if the top of the stack contains the start of the grammar and if the buffer is empty.
3. Choose between the following two steps (if the choice is ambiguous, choose one based on an oracle) :
   - Shift a symbol from the buffer onto the stack.
   - If the top k symbols of the stack are α_1, ……α_k, which corresponds to the right hand side of a CFG rule A → α_1, …. α_k, then replace the top k symbols with the left-hand side nonterminal A.
4. Exit with failure if no action can be taken in previous step.
5. Else, go to step 2.

Fig. 2.3.2

Hypergraphs and chart parsing

CYK algorithm

- CFG requires the use of database.

- Due to linear parsing technique in CFG in the worst case ran time of algorithm is exponential in the grammar size.
- To address this instead of left to right parsing statistical parser which search the space for possible sub trees is used.
- As shown in below example the example grammar can be rewritten in which right hand side contain only two non-terminals.

Example G

$N \rightarrow N$ 'and' N

$N \rightarrow N$ 'or' N

$N \rightarrow$ 'a' | 'b' | 'c' |

New Gc

$N \rightarrow N\ N^{\wedge}$

$N^{\wedge} \rightarrow$ 'and' N

$N \rightarrow N\ N_{\vee}$

$N_{\vee} \rightarrow N$ 'or' N

$N \rightarrow$ 'a' | 'b' | 'c' |

- This can be further made compact by linking input sentence into spans 0 a 1 and 2 b 3 or 4 c 5 etc. i.e. string a is in span 0, 1.

  b or c is in span 2, 5.
- So a new Grammar G_f is written using this concept as below :

New Gf

$N\ [0, 5] \rightarrow N\ [0, 1]\ N^{\wedge}\ [1, 5]$

$N\ [0, 3] \rightarrow N\ [0, 1]\ N\wedge\ [1, 3]$

$N\wedge\ [1, 3] \rightarrow$ 'and' $[1, 2]\ N\ [2, 3]$

$N\wedge\ [1, 5] \rightarrow$ 'and' $[1, 2]\ N\ [2, 5]$

$N\ [0, 5] \rightarrow N\ [0, 3]\ N\vee\ [3, 5]$

$N\ [2, 5] \rightarrow N\ [2, 3]\ N\vee\ [3, 5]$

$N\vee\ [3, 5] \rightarrow$ 'or' $[3, 4]\ N\ [4, 5]$

$N\ [0, 1] \rightarrow$ 'a' $[0, 1]$

$N\ [2, 3] \rightarrow$ 'b' $[2, 3]$

$N\ [4, 5] \rightarrow$ 'c' $[4, 5]$

- The parse tree that starts from a start nonterminal and spans complete string is shown in Fig. 2.3.3.

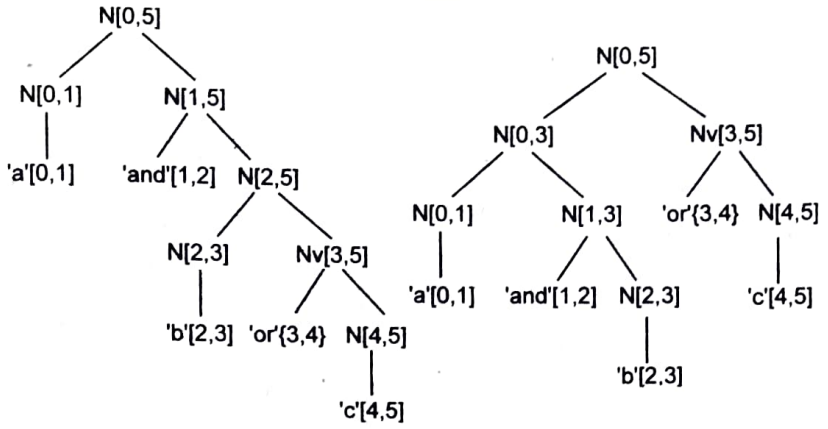


Fig. 2.3.3 : Parse trees embedded in the specialized CFG for a particular input string. The nodes with the same label, such as N[0,5], N[0,1] and [1,2], N[2,3] and Nv[3,5] can be merged to form a hypergraph representation of all parses for the input

- To construct this specialized CFG following are steps :
- First rules generating lexical items are considered

  For ex. $N\ [0, 1] \rightarrow$ 'a' [0, 1]

  $N\ [2, 3] \rightarrow$ 'b' [2, 3]

  $N\ [4, 5] \rightarrow$ 'c' [4, 5]

- The pseudorod is written as shown in Fig. 2.3.4.

```
for i = 0 ..... n do
    if N → x with score s for any x spanning i, i + 1 exists then
      add specialized rule N [i, i + 1] → x [i, i + 1] with score s
      written as N[i, i + 1] : s
    end if
end for
```

Fig. 2.3.4

- In the next step specialized rules are created recursively. For ex. If Y[i, k] and Z[k, j] are left hand sides of previously created rules then rule rate $X \rightarrow YZ$ can be converted to

  $$X\ [i, j] \rightarrow Y\ [i, k]\ Z\ [k, j]$$

- Score S is assigned to each non-terminal span.
- The algorithm is shown in Fig. 2.3.5 and is known as CKY (Cocke, Kasami and younger) algorithm.

```
for j = 2 .... n do
    for i = j – 1 ... 0 do
     for k = i + 1 .... j do
```

```
            if Y [i, k] : s1 and Z[k, j] : s2 are in the specialized grammar
                then
                if X → YZ with score s exists in the original grammar
                then
                    add specialized rule X[i, j] → Y[i, k] Z[k, j] with score s
                        + s1 + s2 keep only the highest scoring rule : X[i,
                        j] → α
                end if
            end if
        end for
    end for
end for
```

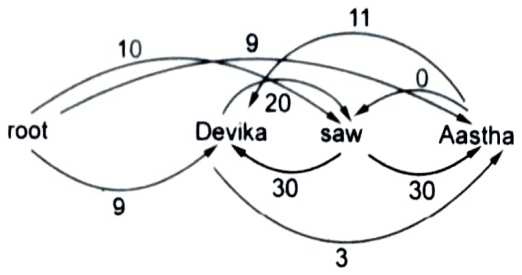
Fig. 2.3.5

Minimum spanning trees and dependency parsing

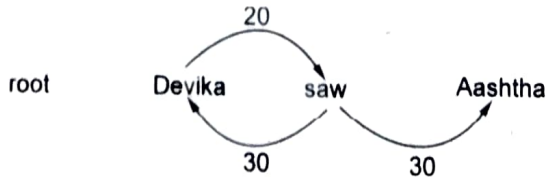
- The Minimum Spanning Tree (MST) corresponds to the optimum branching problem in directed graphs. Which are rooted and does not have cycles.
- The basic prequisite is all the dependency links between the words must have score.
- To understand the working of MST consider the input sentence as

  Devika saw Aastha

- The fully connected graph of this sentence can be drawn as follows,



- The scoring function is used to assign the weights to the edges.
- The algorithm starts with finding the incoming edge with highest score.



- We have a cycle in this graph. We can combine the cycle into single node and recalculate edge weight.
- The edge weight from each node to this newly combined node is computed.
- Also record the maximum weight of a node in this combination.

- For example the incoming edge.

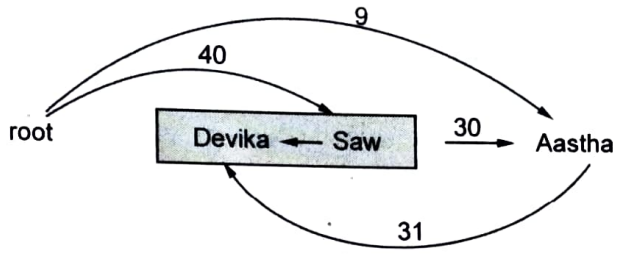
root → **saw → Devika** is having weight 10 + 30 = 40

Whereas root → **Devita → saw** is having weight 9 + 20 = 29

Aastha → **saw → Devika** is having weight 0 + 30 = 30

and Aastha → **Devika → saw** is having weight 11 + 20 = 31

So the graph can be viewed as

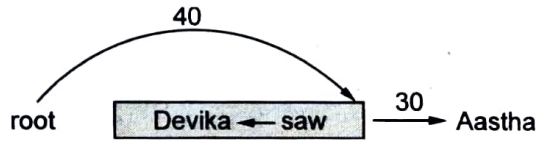


- The MST algorithm is recursively applied to this graph and best incoming edges to each word are found.
- So in the next iteration

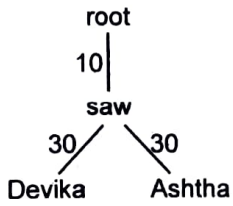
root → Aastha → **Devika → saw** will have weight 9 + 31 = 40

and root → **Devika → saw** → Aastha will have weight 40 + 30 = 70

- So the best nodes chosen are shown in below graph



And we get a highest scoring dependency parse as



Review Questions

1. *Explain shift reduce parser.*
2. *Explain CYK algorithm.*
3. *Explain minimum spanning trees and dependency passing.*

2.4 Probabilistic Context Free Parsing and Statistical Parsing

- Consider the example sentence "Atul bought a shirt with pockets" explain a in θ.
- Please refer to the CFG and parse trees of this sentence from θ_1.

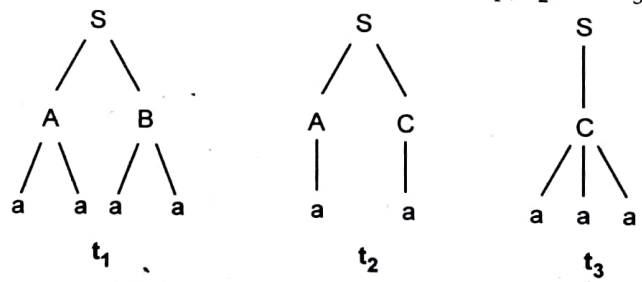
- To resolve the ambiguity of such type, one way is to assign the probabilities to the rules in CFG.
- Due to this the CFG is known as Probabilistic Context Free Grammar or PCFG. For example for the rule $N \rightarrow \alpha$ the probability can be defined as $P(N \rightarrow \alpha/N)$ such that rule probability is stated at left hand side.
- Due to this assignment when non-terminal is expanded, probability distribution is done among all expansions of non-terminals. i.e.

$$1 = \sum_{\alpha} P(N \rightarrow \alpha)$$

- So for the example sentence the probability distribution can be viewed as,

S → NP VP (1.0)
NP → 'Atul' (0.1) | 'pockets' (0.1) | DN (0.3) | NP PP (0.5)
VP → V NP (0.9) | VP PP (0.1)
V → 'bought' (1.0)
D → 'a' (1.0)
N → 'shirt' (1.0)
PP → P NP (1.0)
P → 'with' (1.0)

- From the assigned probabilities we can observe that the decision should be taken from two rules NP → NP PP and VP → VP PP and as probability of NP → NP PP is having higher probability it generates a more feasible sentence.
- Consider the following example of derivation of rule probabilities from treebank. The figure below shows treebank with three trees t_1, t_2 and t_3.



Assume that t_1 occur 10 times, t_2 : 20 times and t_3 : 50 times in treebank.

PCFG for this can be given as,

$$\frac{10}{10+20+50} = 0.125 \qquad S \rightarrow A\ B$$

$$\frac{20}{10+20+50} = 0.25 \qquad S \rightarrow A\ C$$

$$\frac{50}{10+20+50} = 0.625 \qquad S \rightarrow C$$

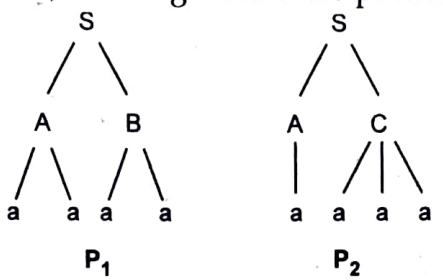
$$\frac{10}{10+20} = 0.334 \qquad A \rightarrow a\ a$$

$$\frac{20}{10+20} = 0.667 \qquad A \rightarrow a$$

$$\frac{20}{20+50} = 0.285 \qquad B \rightarrow a\ a$$

$$\frac{50}{20+50} = 0.714 \qquad C \rightarrow a\ a\ a$$

- If we consider input a a a a, we can generate two parses.



Where, $P_1 = 0.125 * 0.334 * 0.285 = 0.01189$

$P_2 = 0.25 * 0.667 * 0.714 = 0.119$

So P_2 is most feasible for parsing.

Generative models for parsing for ambiguity resolution

- A parse tree is typically built by sequence of decisions.
- Based on the CFG rules there can be multiple derivations.
- Consider each such derivation as, $D = d_1, \ldots\ldots, d_n$
- Parser has to choose from these derivations, the most feasible one.
- Let us consider input sentence as x and output parse tree need to be generated as y.
- For each derivation of parse tree the probability can be assigned as ,

$$P(x, y) = P(d_1, \ldots\ldots, d_n) = \prod_{i=1}^{n} (d_i \mid d_1, \ldots\ldots, d_{i-1})$$

- In this equation a partial parse tree is built by probability $(d_i \mid d_1, \ldots\ldots, d_{i-1})$ which is known as history.
- These histories are grouped into conditional classes by function ϕ as :

$$P(d_1, \ldots\ldots, d_n) = \prod_{i=1}^{n} (d_i \mid \phi(d_1, \ldots\ldots, d_{i-1})$$

- History $H_i = d_1, \ldots\ldots, d_{i-1}$ for all x, y is represented by finite set of feature functions k

$$\phi_1 (H_i) , \ldots\ldots \phi_k (H_i)$$

$$P (d_1 , \ldots\ldots , d_n) = \prod_{i=1}^{n} [d_i \mid \phi_1 (H_i) , \ldots\ldots , \phi_k (H_i)]$$

Global linear discriminative model ambiguity resolution parsing

Global linear model

- Discriminative model in developed by Collins for creating simple framework for describing discriminative approaches, which is also called as **global linear model.**
- Consider X as set of inputs and Y as output which is sequence of POS tags as parse tree.
- Each input $x \in X$ and $y \in Y$ is mapped to d-dimensional feature vector $\phi (x, y)$. Each dimension is a real number.
- Weigh is assigned to each feature is $\phi(x, y)$ by weight parameter vector $w \in R^d$.

  $\phi(x, y) \cdot w$ = Score of (x, y)

  If the score is higher y is the most feasible output for x.

- Possible outputs y from x are computed from function GEN(x).
- The highest scoring candidate y* from GEN(x) is computed as

$$F(x) = \underset{y \in GEN(x)}{\text{argmax}} \; P (y \mid x, w)$$

- Conditional random field C (RF) computer conditional probability as

$$\log p (y \mid x, w) = \phi(x, y) \cdot w - \log \sum_{y' \in GEN(x)} \exp [\phi(x, y') \cdot w]$$

- Global linear model is stated as,

$$F(x) = \underset{y \in GEN(x)}{\text{argmax}} \; \phi (x, y) \cdot w$$

The original perceptron learning algorithm for ambiguity resolution in parsing

- Perceptron is a single layed neural network.
- It process a example at a time.
- The weight adjustment is done of weight parameter vector.
- This vector is applied to input to generate the related output.
- The features present in the truth are "recognized.
- Consider a training set with in examples.
- As shown in Fig. 2.4.1 Algorithm 2.4.1 the original perceptron learning 2.4.1 Algorithm works like below.

Algorithm 2.4.1 : The original perceptron learning algorithm

Inputs : Training data $\{ (x_1, y_1), \ldots\ldots (x_m , y_m) \}$; number of iterations T

Initialization : Set w = 0

Algorithm :

1. **for t** = 1, ..., T **do**
2. **for** i = 1, ... , m **do**
3. Calculate y_i , where $y_i' = \underset{y \in GEN(x)}{\text{argmax}}\ \Phi(x_i , y) \cdot w$
4. **if** $y_i' \neq y_i$ **then**
5. $w = w + \Phi(x_i , y_i) - \Phi(x_i , y_i')$
6. **end if**
7. **end for**
8. **end for output :** The updated weight parameter vector w

Fig. 2.4.1 : Algorithm 2.4.1

1. Weight parameter w is initialized to 0.
2. Iteration is carried out on m training examples.
3. Set of candidates GEN(x) is generated for each x.
4. The most feasible candidate i.e. the candidate with maximum score according to w is selected.
5. w is updated by increasing weight values of features in truth and decreasing weight value for features appearing in top candidate.

- The problem faced by this algorithm is of overfitting during incremental weight update due to which unseen data is not classified properly.
- The algorithm is not suitable for linearly inseparable training data.

Voted perceptron algorithm for ambiguity resolution in parsing

- As shown in Fig. 2.4.2 Algorithm 2.4.2 the voted perceptron algorithm, proposed by freund and Schapire and works as follows :
- Instead of a single weight vector W, the learning process considers all intermediate weight vectors.
- In classification phase-these intermediate vectors are used to vote for the answer.
- Good prediction vector generally service for long time and have larger weight in the vote.

- In training phase count C_i counts the survival of weight parameter vector (w_i, C_i) in training.
- If top candidate is not in truth C_{i+1} is initialized 1, to generate an updated weight vector (w_{i+1}, C_{i+1}).
- While this original C_i and weight vector (W_i, C_i) are stored.
- This algorithm is more stable - than original perceptron.

Algorithm 2.4.2 : The voted perception algorithm

Training phase

Input : Training data $\{ (x_1, y_1), \ldots\ldots (x_m, y_m) \}$; number of iterations T

Initialization : $k = 0$, $w_0 = 0$, $c_1 = 0$

Algorithm :

for t = 1, ..., T do

  for i = 1, ... , m do

    Calculate y_i , where $y_i' = \underset{y \in GEN(x)}{\text{argmax}} \Phi(x_i, y) \cdot w_k$

    if $y_i' \neq y_i$ then

      $c_k = c_k + 1$

    else

      $w_{k+1} = w_k + \Phi(x_i, y_i) - \Phi(x_i, y_i')$

      $c_{k+1} = 1$

      $k = k + 1$

    end if

  end for

end for

output : A list of weight vectors $((w_1, c_1)), \ldots\ldots, (w_k, c_k)$

Prediction Phase

Input : The list of weight vectors $((w_1, c_1), \ldots\ldots, (w_k, c_k))$, an unsegemented sentence x.

Calculate :

$$y^* = \underset{y \in GEN(x)}{\text{argmax}} \left(\sum_{i=1}^{k} c_i \Phi(x, y) \cdot w_i \right)$$

Output : The voted top ranked candidate y*.

Fig. 2.4.2 : Algorithm 2.4.2

Averaged perceptron algorithm for ambiguity resolution in parsing

- The averaged perception algorithm reduces space and time complexities.
- As shown in Fig. 2.4.3 Algorithm 2.4.3 instead of w-he averaged weight parameter vector γ on m training examples is used.
- γ can be defined as

$$\gamma = \frac{1}{mT} \sum_{i = 1, ..., m , t = 1.... T} w^{i, t}$$

- Accumulating parameter vector σ is maintained and updated using w.

Algorithm 2.4.3 : The averaged perceptron learning algorithm

Input : Training Data < (x_1, y_1), ... (x_m , y_m) }; number of iterations T

Initialization : Set $w = 0, \gamma = 0, \sigma = 0$

Algorithm :

for t = 1, ..., T do

for i = 1, ... , m do

Calculate y_i , where $y_{i'} = \text{argmax}_{y \in GEN(x)} \Phi(x_i , y) \cdot w$

if $y_{i'} \neq y_i$ then

$w = w + \Phi(x_i , y_i) - \Phi(x_i , y_{i'})$

end if

$\sigma = \sigma + w$

end for

end for

output : The averaged weight parameter vector $\gamma = \sigma /(mT)$ words, are used as heads and dependent

Fig. 2.4.3 : Algorithm 2.4.3

Review Questions

1. *Explain probabilistic context free passing and statistical passing.*
2. *Explain generative models for parsing for ambiguity resolution.*
3. *Explain global linear model.*
4. *Explain voted perceptron algorithm for ambiguity resolution in passing.*

2.5 Semantic Analysis : Lexical Semantic

- Word is the smallest entity of any document.
- To know the similarity of two words, context in which they appear plays a major role.
- Similarity of context plays important role in finding out semantic similarity.
- Finding out semantic similarity is beneficial in the applications like summarization, question answering, text classification, plagiausm detection, etc.
- Let's first understand the meaning of lexical semantics with the help of following terms :
  - **Lexeme :** It is a basic unit of meaning. It can be considered as a word in its most basic form. It can be a group of word forms belonging to same word class with same basic meaning.
  - **For ex :** The words is, was, will belongs to one lexeme. The words "come, came" belongs to same lexeme.
  - **Lexicon :** It is a finite set of lexemes.
  - **Lemma :** It is grammatical form used to represent lexeme for ex : Lemma form of sing, sang, sung is sing.
  - **Lemmatization :** It is the process of mapping a wordform to Lemma.
- With this background lexical semantics is defined as the study of meaning of words and the systematic meaning related connections between words.
- The computation lexical semantics deals with computational tasks related with word, senses, relations among words and structure of predicate bearing words.

2.6 WordNet

- In this world people communicate with each other in more than 7000 languages.
- Written communication is the most popular form of communication.
- Any text comprises sentences which are formed by basic text units i.e words.
- Automatic understanding, analysis and preprocessing of text and in turn words is a challenging task for the NLP system.
- Typically this is done with the help of lexicons.
- A lexicon is a vocabulary of a person, language or branch of knowledge.
- Text in the textual data is mapped to the lexicon which helps us to understand the relationships between the words.

- WordNet is the lexical resource which helps us find word relations, synonyms, grammers, etc.
- WordNet is useful in solving many NLP applications like machine translation, sentiment analysis, etc.
- WordNet is a network of words linked by lexical and semantic relations and is freely and publicly available
- The words in WordNet are related by synonymy, for example the words shut and close or car and automobile possess the same concept and can be interchanged as per the context. All such words are grouped into unordered sets or synsets.
- Under synsets nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms.
- Synsets are interlinked by means of conceptual-semantic and lexical relations.
- WordNet can be confused with thesaurus as, just like thesaurus it is collection words based on their meaning but following are the unique points of WordNet :
  - Along with word forms WordNet also interlinks specific senses of words. Due to this, words that are found in close proximity to one another in the network are semantically disambiguated.
  - WordNet labels the semantic relations among words, whereas the groupings of words in a thesaurus does not follow any explicit pattern other than meaning similarity.
- The major relations which are covered in WordNet are shown in below table :

Semantic Relation	Syntactic Category	Examples
Synonymy (similar)	N, V, Aj, Av	Pipe, tube rise, ascend sad, unhappy rapidly, speedily.
Antonymy (opposite)	Aj, Av, (N, V)	Wet, dry powerful, powerless friendly, unfriendly rapidly, slowly.
Hyponumy (subordinate)	N	Sugar maple, maple maple, tree tree, plant
Meronymy (part)	N	brim, hat gin, martini ship, fleet.
Troponomy (manner)	V	March, walk whisper, speak.
Entailment	V	Drive, ride divorce, marry.
Note : N = Nouns Aj = Adjectives V = Verbs Av = Adverbs		

Reference : Semantic relations in WordNet. Source: Miller 1995, table 1.

- Following are meanings of the relations mentioned in the table :
  - Synonymy : Words with the same meaning.
  - Polysemy : Words with more than one sense.
  - Hyponymy/Hypernymy : Words which are having is-a relation.
    For example : Peacock is a type of bird. Likewise, bird is a hypernym of peacock.
  - Meronymy/Holonymy : Words with Part-whole relation. For example: Beak is meronym of bird since beak is part of a bird's anatomy. Likewise, bird is holonym of beak.
  - Antonymy : Words which are lexically opposite. For example : Hot, cold.
  - Troponymy : Applies to verbs. For example, whisper is a troponym of talk since whisper elaborates on the manner of talking.
- Many WordNets are built across the world for various languages. Few of them are listed in the below table :

Language	Resource name	Developer(s)
Multilingual (South African languages/isiZulu, isiXhosa, Setswana, Sesotho sa Leboa, Tshivenḓa, Sesotho, isiNdebele, Siswati & Xitsonga	African Wordnet (AfWN)	University of South Africa & South African Centre for Digital Language Resources, Pretoria, South Africa
Multilingual (Hindi/ Indonesian/ Japanese/ Lao/ Mongolian/ Burmese/ Nepali/ Sinhala/ Thai/ Vietnamese)	Asian WordNet	National Electronics and Computer Technology Center (NECTEC) – Thai Computational Linguistics Laboratory (TCL) – NICT, Kyoto, Japan
Multilingual (English/ Spanish/ Catalan/ Basque/ Italian)	Multilingual Central Repository	University of the Basque Country – Department of Software, Technical University of Catalonia (UPC)
English	WordNet 3.01	Princeton University
Hindi, Marathi, Sanskrit	Hindi WordNet	Indian Institute of Technology Bombay Powai, Mumbai
	Marathi WordNet	
	Sanskrit Wordnet	

Review Question

1. *Explain WordNet.*

2.7 Word Sense Disambiguation (WSD)

- To understand the concept of word sense disambiguation let's consider following example.
  1. I went to bank to withdraw money.
  2. I went to bank to fetch water.
- In the above example the same word 'bank' has appeared in two different senses and contexts.
- In computational lexical semantics it is required to examine the contextual word tokens and to figure out which sense of word is used.
- This task is known as Word Sense Disambiguation (WSD).
- WSD is the task of selecting the correct sense for a word.
- WSD is essential part of many important NLP applications like question answering, information retrieval and text classification where the use of wrong senses of words can create disasters.
- Typically any WSD algorithms, input is a word in context and fixed inventory of possible word senses. The output is a correct word sense.
- The task for which WSD is done decides nature of input and inventory of senses used.
- For example : If the task is of machine translation from English to Spanish, sense tag inventory for an English word can be set of Spanish translations. But if the task is automatic indexing of medical articles inventory can be MeSH (Medical Subject Headings) thesaurus entries.
- The main requirement of supervised WSD is we need labeled data.
- If we have hand labeled data with proper word senses, then using supervised WSD can be used to extract features from the text.
- These features can be used to predict the senses and a classifier can be learned for assigning correct senses from the features.
- These are two broad steps in supervised WSD :
  1. Creative extraction for supervised learning.
  2. Training a classifier.

1. Feature extraction for supervised learning :

- In this step features, predicting word sense are extracted.
- First preprocessing is performed which includes POS tagging, stemming and lemmatization.
- A feature vector is generated, which contains numeric values for this linguistic information.
- This feature vector can be used as an input to machine learning algorithms.
- Features are classified in two classes :

a) Collocational features

b) Bag of words features.

a) Collocational features :

- Collocation refers to a group of words that often go together or occur together.
- Some examples of collocation arc
  - To feel free → please feel free to take a seat
  - To deposit a check → I would like to deposit this check
  - Deeply regret the loss of someone → I deeply target the loss of your loved one.
- Collocational features takes into account the information about specific positions present to the left or right of the target word.
- Consider the example :

  An electric guitar and **bass** player stand off to one side not really part of the scene, just as a sort of nod to gringo expectational perhaps. Consider 'bass' as a target word the collocation feature vector can be written considering two words to the left and to the right with their respective parts of speech.
- The collocation feature vector can be written for the above examples POS.

  $[w_{i-2}, POS_{i-2}, w_{i-1}, POS_{i-1}, w_{i+1}, POS_{i+2}]$

  [guitar, NN and CC, player, NN, Stand, VB]

b) Bag-Of-Words :

- Concept of bag of words is explained in detail in section 3.9.
- In WSD task using BOW concept a target word is kept as center and the context region surrounding the target word is small, symmetric, fixed size window.
- Stop words are not used as features and the vector is limited to BOW considering small number of frequently occurring words.

- Again consider the same bass example in BOW vector with 12 most frequent words feature set in written as [fishing, big, sound, player, fly, rod, pound, double, runs, playing, guitar, band] with window size 10 the above feature set is written as binary vector

  [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0]

2. Training a classifier :

- Once the features are extracted, along with the training data is used to train sense classifier.
- Among the classifiers, Naive Bayes and decision list approaches are used mostly for the task of WSD.
- Naive Bayes classifier for WSD : Naive Bayes classifier is used to find the best sense $\hat{s}$ out of all possible senses S for feature vector $\vec{f}$, i.e.

$$\hat{s} = \underset{s \leftarrow S}{\text{argmax}} \; P(s | \vec{f})$$

- To understand formulation of Naive Bayes classifier, let's consider BOW vector for 20 words having 2^{20} possible feature vectors.
- To solve this problem, first the problem in reformulated in Bayesian manner as,

$$\hat{s} = \underset{s \in S}{\text{argmax}} \frac{P(\vec{f} | s) \, P(s)}{P(\vec{f})}$$

- In this equation vector $\vec{f}$ for available data with each sense S is very sparse. But in tagged training set the information about individual feature value pair is having greater presence.
- So an independent assumption is made naively that features are independent of each other.
- Considering this the approximation for $P(\vec{f} | s)$ is written as,

$$P(\vec{f} | s) \approx \prod_{j=1}^{n} P(f_i | s)$$

- $P(\vec{f})$ is same for all the senses, it is condensed that it will not affect final ranking.
- With this assumption Naive Bayes classifier for WSD in given as,

$$\hat{s} = \underset{S \in s}{\text{argmax}} \; P(s) \prod_{j=1}^{n} P(f_j | s)$$

- To train naive bayes classifier each of the above probability is estimated.
- The prior probability of each sense P(s), the maximum likelihood estimate of the probability from sense-tagged training corpus in calculated by number of times sense s_i occurrance and dividing it by total count of target word w_j.

So,
$$P(s_i) = \frac{\text{count}(s_i, w_j)}{\text{count}(w_j)}$$

- So in our example of 'bass' if collocational feature guitar occurs 3 times for sense 'bass' and if sense "bass" occurs 60 times in training the maximum likelihood estimate is $P(f_j | s) = 0.05$.

2) Decision list classifier :

- In this, a sequence of tests is applied to each target word feature vector.
- A specific sense is shown by each test.
- A sense is returned with a successful test and in case of failure the next test is applied till end of list.
- Consider the disambiguation of the sense 'bass' from Fig. 2.7.1.
- As shown in Fig. 2.7.1 the first test indicates that if word fish occurs in input context then bass in correct.

Rule		Sense
fish within window	⇒	$bass^1$
stripped bass	⇒	$bass^1$
guitar within window	⇒	$bass^2$
bass player	⇒	$bass^2$
piano within window	⇒	$bass^2$
tenor within window	⇒	$bass^2$
sea bass	⇒	$bass^1$
play/V bass	⇒	$bass^2$
river within window	⇒	$bass^1$
violin within window	⇒	$bass^2$
salmon within window	⇒	$bass^1$

on bass	⇒	$bass^2$
bass are	⇒	$bass^1$

Fig. 2.7.1 An abbreviated decision list for disambiguating the fish sense of bass from the music sense (Yarowsky, 1997)

- In case fish doesn't occur next text is carried out till we obtain the result as true.
- The default test returning trade is mentioned at the last in the list.
- In decision tree classifier then sense in indicated by a particular feature by finding out ratio of the probabilities of senses as follows :

$$\left| \log \left(\frac{P(sense_1 | f_i)}{P(sense_2 | f_i)} \right) \right|$$

Review Question

1. *What is wordsense disambiguation.*

2.8 Latent Semantic Analysis

- Latent Semantic Analysis (LSA) is a mathematical method for computer modelling and simulation of the meaning of words and passages in natural text corpora.
- LSA is a technique in natural language processing of analyzing relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms.
- Many aspects of human language learning and comprehension are well represented by LSA.
- When complex wholes are treated as additive functions of component parts, they can be used to solve problems like information retrieval (one of the text mining techniques), educational technology and pattern recognition.
- Latent Semantic Analysis (LSA) is a type of natural language processing that looks at how documents and the terms they contain are related.
- It searches unstructured data for hidden relationships between terms and concepts using singular value decomposition, a mathematical technique.

LSA Implementation

- LSA assumes that words that are close in meaning will occur in similar pieces of text.

- A matrix containing word counts per document (rows represent unique words and columns represent each document) is constructed from a large piece of text.
- A mathematical technique called Singular Value Decomposition (SVD) is used to reduce the number of rows while preserving the similarity structure among columns.
- Documents are then compared by cosine similarity between any two columns.
  - Values close to 1 represent very similar documents
  - Values close to 0 represent very dissimilar documents.
- The SVD is typically computed using large matrix methods and computed incrementally.
- It uses a neural network-like approach to compress resources via a neural network-like approach. As a result, it does not require the large, full-rank matrix to be held in memory.
- LSA has been used to assist in performing prior art searches for patents.

LSA Process

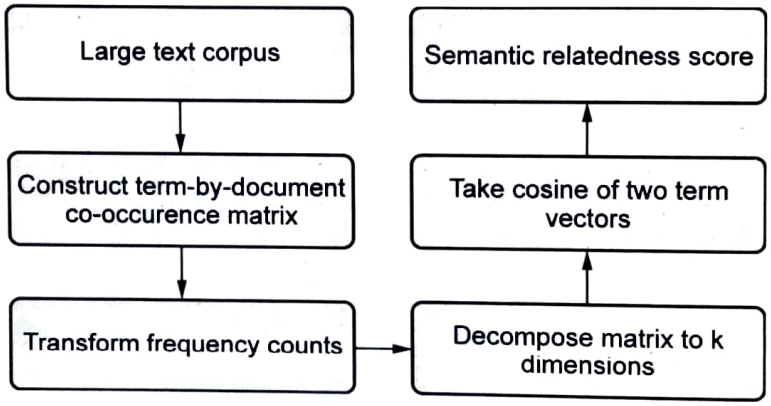


Fig. 2.8.1

Limitations of LSA

- Due to size of the resulting dimensions, it is difficult to interpret.
- LSA can only partially capture polysemy (i.e., multiple meanings of a word) because each occurrence of a word is treated as having the same meaning due to the word being represented as a single point in space.
- The probabilistic model of LSA does not match observed data : LSA assumes that words and documents form a joint Gaussian model (ergodic hypothesis), while a Poisson distribution has been observed.

Review Question

1. *What is latent semantic analysis ?*