1

INTRODUCTION

Unit Structure

1.0 OBJECTIVES

The objective of this module is to learn about history of Natural Language Processing (NLP), Generic NLP system, Levels of NLP. After that we will discuss the knowledge in language processing. This is followed by a discussion of the Ambiguity in Natural language, Stages in NLP, and challenges of NLP. Finally we will discuss the applications of NLP.

1.1 HISTORY OF NLP

As we all know Natural Language refers to the language spoken by people e.g. English, Hindi, Marathi as opposed to artificial/programming languages, like C, C++, Java, etc.

Natural Language Processing (NLP) is a field of research which determines the way computers can be used to understand and manage natural language text or speech to do useful things.

Natural Language Processing (NLP) a subset technique of Artificial Intelligence which is used to narrow the communication gap between the Computer and Human. It is originated from the idea of Machine Translation (MT) which came to existence during the 1950. The primary aim was to translate one human language to another human language, for example, Russian language to English language using the brain of the Computers but after that, the thought of conversion of human language to computer language and vice-versa emerged, so that communication with the machine became easy.

NLP is a process where input provided in a human language and converts this input into a useful form of representation. The field of NLP is primarily concerned with getting computers to perform interesting and useful tasks with human languages. The field of NLP is secondarily concerned with helping us come to a better understanding of human language.

As stated above the idea had emerged from the need for Machine Translation in the 1950s. Then the original language was English and Russian. But the use of other words such as Chinese also came into existence in the initial period of the 1960s.

In the 1960s, the NLP got a new life when the idea and need of Artificial Intelligence emerged.

In 1978 LUNAR is developed by W.A woods; it could analyze, compare and evaluate the chemical data on a lunar rock and soil composition that was accumulating as a result of Apollo moon missions and can answer the related question.

In the 1980s the area of computational grammar became a very active field of research which was linked with the science of reasoning for meaning and considering the user‘s beliefs and intentions.

In the period of 1990s, the pace of growth of NLP increased. Grammars, tools and Practical resources related to NLP became available with the parsers. Probabilistic and data-driven models had become quite standard by then.

In 2000, Engineers had a large amount of spoken and textual data available for creating systems. Today, large amount of work is being done in the field of NLP using Machine Learning or Deep Neural Networks in general, where we are able to create state-of-the-art models in text classification, Question and Answer generation, Sentiment Classification, etc.

1.2 GENERIC NLP SYSTEM

Natural language Processing should start with some input and ends with effective and accurate output. The possible inputs to an NLP are quite broad. Language can be in a variety of forms such as the paragraphs of text, commands that are typed directly to a computer system, etc. The input language might be given to the system a sentence at a time or it might be multiple sentences all at once. The inputs for natural language processor can be typed input, message text or speech. NLP systems have some kind of pre-processor. Data preprocessing involves preparing and cleaning text data for machines to be able to analyze it. Preprocessing puts data in workable form and highlights features in the text that an algorithm can work with. Preprocessing does dictionary lookup, morphological analysis, lexical substitutions, and part-of-speech assignment.

There are variety of output can be generated by the system. The output from a system that incorporates an NLP might be an answer from a database, a command to change some data in a database, a spoken response, Semantics, Part of speech, Morphology of word, Semantics of the word or some other action on the part of the system. Remember these are the output of the system as a whole, not the output of the NLP component of the system.

Figure 1.2.1 shows a generic NLP system and its input-output variety. Figure 1.2.2 shows a typical view of what might be inside the NLP box of Figure 1.2.1. Each of the boxes in Figure 1.2.2 represents one of the types of processing that make up an NLP analysis.

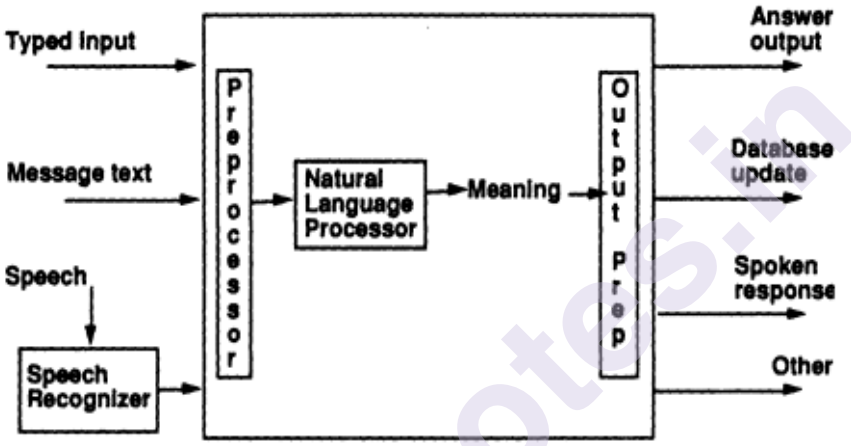


Figure 1.2.1 Generic NLP system.

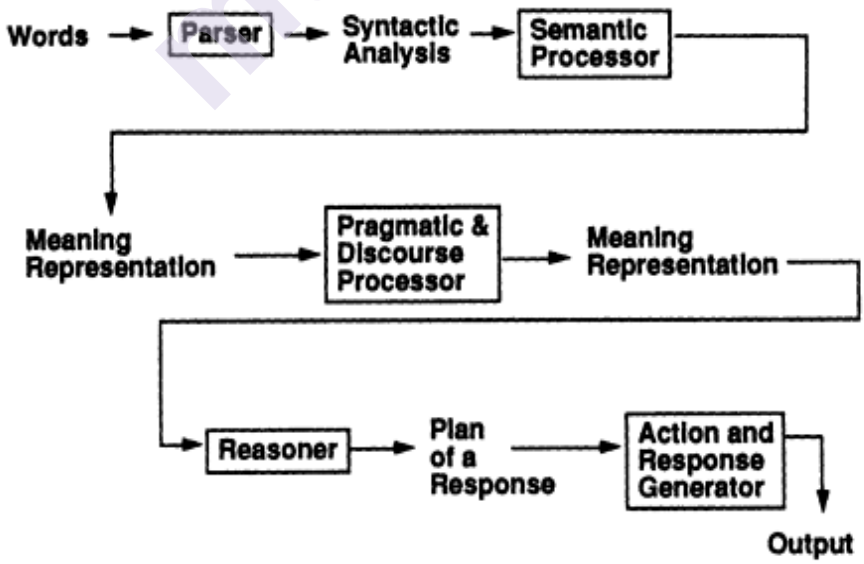


Figure 1.2.2 Pipeline view of the components of a generic NLP system.

1.3 LEVELS OF NLP

There are two components of Natural Language Processing

a. Natural Language Understanding (NLU)

- NLU takes some spoken/typed sentence and working out what it means.
- Here different level of analysis required such as morphological analysis, syntactic analysis, semantic analysis, discourse analysis, …

b. Natural Language Generation (NLG)

- NLG takes some formal representation of what you want to say and working out a way to express it in a natural (human) language (e.g., English)
- Here different level of synthesis required: deep planning (what to say), syntactic generation

Difference between NLU and NLG

NLU	NLG
It is the process of reading and interpreting language.	It is the process of writing or generating language.
NLU explains the meaning behind the written text or speech in natural language	NLG generates the natural language using machines.
NLU draws facts from the natural language using various tools and technologies such as parsers, POS taggers, etc,	NLG uses the insights generated from parsers, POS tags, etc. to generate the natural language.
NLU understands the human language and converts it into data	NLG uses the structured data and generates meaningful narratives out of it

The NLP can broadly be divided into various levels as shown in Figure 1.3.1.

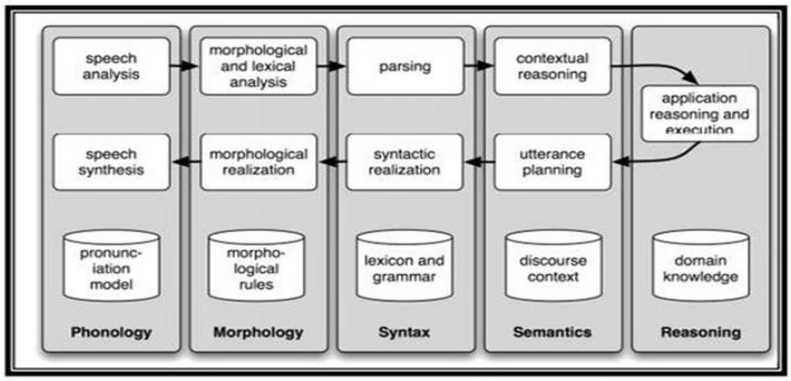


Figure 1.3.1 Levels of NLP

1. **Phonology:**
   It concerned with interpretation of speech sound within and across words.

2. **Morphology:**
   It deals with how words are constructed from more basic meaning units called morphemes. A morpheme is the primitive unit of meaning in a language. For example, "truth+ful+ness".

3. **Syntax:**
   It concerns how words can be put together to form correct sentences and determines what structural role each word plays in the sentence and what phrases are subparts of other phrases.

   For example, "the dog ate my homework"

4. **Semantics:**
   It is a study of the meaning of words and how these meaning combine in sentences to form sentence meaning. It is study of context- independent meaning. For example, plant: industrial plant/ living organism

   Pragmatics concerns with how sentences are used in different situations and how it affects the interpretation of the sentence. Discourse context deals with how the immediately preceding sentences affect the interpretation of the next sentence. For example, interpreting pronouns and interpreting the temporal aspects of the information.

5. **Reasoning:**
   To produce an answer to a question which is not explicitly stored in a database; Natural Language Interface to Database (NLIDB) carries out reasoning based on data stored in the database. For example, consider the database that holds the student academic information, and user posed a query such as: 'Which student is likely to fail in Science subject?' To answer the query, NLIDB needs a domain expert to narrow down the reasoning process.

1.4 KNOWLEDGE IN LANGUAGE PROCESSING

A natural language understanding system must have detailed information about what the words mean, how words combine to form sentences, how word meanings combine to form sentence meanings and so on.

What distinguishes these language processing applications from other data processing systems is their use of knowledge of language.

For example consider the Unix wc program, which is used to count the total number of bytes, words, and lines in a text file. When used to count lines and bytes, wc is an ordinary data processing application. However, when it is used to count the words in a file it requires information about what it means to be a word, and thus becomes a language processing system. Of course, wc is an extremely simple system with an extremely limited and impoverished knowledge of language.

The different forms of knowledge required for natural language understanding are given below:

Phonetic and Phonological Knowledge

Phonetics is the study of language at the level of sounds while phonology is the study of combination of sounds into organized units of speech, the formation of syllables and larger units. Phonetic and phonological knowledge are necessary for speech based systems as they are concerned with how words are related to the sounds that realize them.

Morphological Knowledge

Morphology concerns with word formation. It's a study of the patterns of formation of words by the combination of sounds into minimal distinctive units of meaning called mophemes. Morphological knowledge deals with how words are constructed from morphemes.

Syntactic Knowledge

Syntax is the level at which we study how words combine to form phrases, phrases combine to form clauses and clauses join to make sentences. Syntactic analysis concerns sentence formation. It's concerned with how words can be put together to form correct sentences. It also determines what structural role each word plays in the sentence and what phrases are subparts of what other phrases.

Semantic Knowledge

It deals with meanings of the words and sentences. This is the study of context independent meaning that is the meaning a sentence has, no matter in which context it is used. Defining the meaning of a sentence is very difficult due to the ambiguities involved.

Pragmatic Knowledge

Pragmatics is the extension of the meaning or semantics. Pragmatics concerned with the contextual aspects of meaning in particular situations. It concerned with how sentences are used in different situations and how use affects the interpretation of the sentence.

Discourse Knowledge

Discourse concerns connected sentences. It's a study of chunks of language which are bigger than a single sentence. Discourse language concerned with inter-sentential links means how the immediately preceding sentences affect the interpretation of the next sentence. Discourse knowledge is important for interpreting pronouns and temporal aspects of the information conveyed.

World Knowledge

Word knowledge is nothing but everyday knowledge that all speakers share about the world. It includes the general knowledge about the structure of the world and what each language user must know about the other user's beliefs and goals. This is essential to make the language understanding much better.

1.5 AMBIGUITY IN NATURAL LANGUAGE

Ambiguity can occur at all NLP levels. It is a property of linguistic expressions. If an expression (word/phrase/sentence) has more than one meaning we can refer it as ambiguous.

For eg: Consider the sentence,

"The chicken is ready to eat."

The meaning in the above phrase can be,

The chicken (food) is ready to be eaten or the chicken (bird) is ready to be feeder

Consider another sentence,

"There was not a single man at the party."

The meaning in this case can be Lack of men altogether or Lack of bachelors at the party.

There are different forms of ambiguity that are relevant in natural language processing.

1. **Lexical Ambiguity:** It's the ambiguity of a single word. A word can be ambiguous with respect to its syntactic class. Eg: book, study.

   For eg: The word "silver" can be used as, an adjective, a noun or a verb.

   - She made a silver speech.
   - She bagged two silver medals.
   - His worries had silvered his hair.

Lexical ambiguity can be resolved by Lexical category disambiguation ie, parts-of speech tagging. As many words may belong to more than one lexical category. Part-of speech tagging is the process of assigning a part-of-speech or lexical category such as a noun, pronoun, verb, preposition, adverb, adjective etc. to each word in a sentence.

Lexical Semantic Ambiguity:

The type of lexical ambiguity, which occurs when a single word is associated with multiple interpretations. Eg: fast, bat, bank, pen, cricket etc.

For eg: 1. The tank was full of water.

2. I saw a military tank.

Words have multiple meanings for such sentences. Consider the sentence "I saw a bat."

Possible meaning of the words which changes the contexts of the sentence are:

- bat flying mammal / wooden club?
- saw past tense of "see" /present tense of "saw " (to cut with a saw.)

The occurrence of tank in both sentences corresponds to the syntactic category noun, but their meanings are different. Lexical Semantic ambiguity resolved using word sense disambiguation (WSD) techniques, where WSD aims at automatically assigning the meaning of the word in the context in a computational manner.

2. **Syntactic Ambiguity:** A sentence can be parsed in different ways. For example, "He lifted the beetle with red cap" or "Did he use cap to lift the beetle".

The structural ambiguities were syntactic ambiguities

Structural ambiguity is of two kinds: i) Scope Ambiguity ii) Attachment Ambiguity

– **Scope Ambiguity**: Scope ambiguity involves operators and quantifiers.

For example: Old men and women were taken to safe locations.

The scope of the adjective (i.e., the amount of text it qualifies) is ambiguous. That is, whether the structure (old men and women) or ((old men) and women)?

The scope of quantifiers is often not clear and creates ambiguity.

Every man loves a woman

The meaning can be, For every man there is a woman and also it can be there is one particular woman who is loved by every man.

- **Attachment Ambiguity**

A sentence has attachment ambiguity if a constituent fits more than one position in a parse tree. Attachment ambiguity arises from uncertainty of attaching a phrase or clause to a part of a sentence.

Consider the example:

The man saw the girl with the telescope.

It is ambiguous whether the man saw her through his telescope or he saw a girl carrying a telescope. The meaning is dependent on whether the preposition 'with' is attached to the girl or the man.

Consider the example:

Buy books for children and attach to the verb buy

Preposition Phrase 'for children' can be either adverbial or adjectival and attach to the object noun books.

3. **Semantic Ambiguity**: This type of ambiguity is typically related to the interpretation of sentence. Even after the syntax and the meanings of the individual words have been resolved, there are two ways of reading the sentence.

Consider example: "Seema loves mother and Sriya does too"

The interpretations can be Sriya loves Seema's mother Sriya likes her own mother.

Semantic ambiguities born from the fact that generally a computer is not in a position to distinguishing what is logical from what is not.

Consider the example: "The car hit the pole while it was moving"

The interpretations can be

- The car, while moving, hit the pole
- The car the pole while the pole was moving.

The interpretation is preferred than the second because we have model of the world that helps distinguish what logical (or possible) from what is not. To supply to computer model the world is not so easy.

Consider the example: "We saw his duck"

Duck can refer person's bird or to a motion he made.

Semantic ambiguity happens when a sentence contains an ambiguous word or phrase.

4. **Discourse Ambiguity:** Discourse level processing needs shared world or shared knowledge and interpretation is carried out using this context. Anaphoric ambiguity under discourse level.

- **Anaphoric Ambiguity:** Anaphora's are the entities that have been previously introduced into the discourse.

  For example, The horse ran up the hill. It was very steep. It soon got tired.

  The anaphoric reference of 'it' in the two situation cause ambiguity. Steep applies to surface hence 'it' can be hill. Tired applies to animate object hence 'it' can be horse.

5. **Pragmatic Ambiguity:** Pragmatic ambiguity refers to a situation where the context of a phrase gives it multiple meanings. One of the hardest tasks in NLP. The problem involves processing user intention, sentiment, belief world etc. all of which are highly complex tasks.

   Consider the example,

   "Tourist (checking out of the hotel): Waiter, go upstairs to myroom and see it my sandals are there; do not be late; I have to catch the train in 15 minutes."

   Waiter (running upstairs and coming back panting) Yes sir, they are there.

   Clearly, the waiter is falling short of the expectation of the tourist, since he does not understand the pragmatics of the situation.

   Pragmatic ambiguity arises when the statement is not specific, and the context does not provide the information needed to clarify the statement. Information is missing, and must be inferred. Consider the example: "I love you too."

   This can be interpreted as:

   - I love you (as well as liking you)
   - I love you (just like someone else does)
   - I love you (just like you love me)
   - I love you (and I love someone else)

   It is a highly complex task to resolve all these kinds of ambiguities, especially in the upper levels of NLP. The meaning of a word, phrase, or sentence cannot be understood in isolation and contextual knowledge is needed to interpret the meaning, pragmatic and world knowledge is required in higher levels. It is not easy job to create a world model for disambiguation tasks. Linguistic tools and lexical resources are needed for the development of disambiguation techniques. Resourceless languages are lagging behind in these fields compared to resourceful languages in implementation of these techniques. Rule based methods are language specific where as

stochastic or statistical methods are language independent. Automatic resolution of all these ambiguities contains several long standing problems but again development towards full-fledged disambiguation techniques is required which takes care of all the ambiguities. It is very much necessary for the accurate working of NLP applications such as Machine Translation, Information Retrieval, Question Answering etc.

Statistical Approaches of Ambiguity Resolution in Natural Language Processing are:

1. Probabilistic model
2. Part of Speech Tagging

- Rule-Based Approaches
- Mak Model Approaches
- Maxman Entropy Approaches
- HMI Based Taggers

3. Machine Learning Approaches

1.6 STAGES IN NLP

There are general five steps in natural language processing

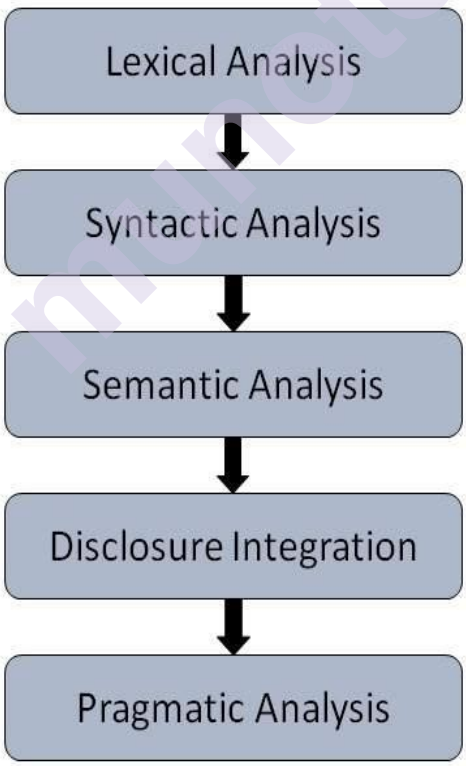


Figure 1.6.1 General five steps of NLP

1) Lexical Analysis:

It is the first stage in NLP. It is also known as morphological analysis. It consists of identifying and analyzing the structure of words. Lexicon of a language means the collection of phrases and words in a language. Lexical analysis is dividing the whole chunk of text into words, sentences, and paragraphs

2) Syntactic Analysis:

Syntactic analysis consists of analysis of words in the sentence for grammar and ordering words in a way that shows the relationship among the words. For example the sentence such as "The school goes to boy" is rejected by English syntactic analyzer.

3) Semantic Analysis:

Semantic analysis is a structure created by the syntactic analyzer which assigns meanings. This component transfers linear sequences of words into structures. It shows how the words are associated with each other. Semantics focuses only on the literal meaning of words, phrases, and sentences. This only draws the dictionary meaning or the real meaning from the given text. The structures assigned by the syntactic analyzer always have assigned meaning

The text is checked for meaningfulness. It is done by mapping syntactic structure and objects in the task domain. E.g. "colorless green idea". This would be rejected by the Symantec analysis as colorless here; green doesn't make any sense.

4) Discourse Integration:

The meaning of any sentence depends upon the meaning of the sentence just before it. Furthermore, it also brings about the meaning of immediately succeeding sentence. For example, "He wanted that", in this sentence the word "that" depends upon the prior discourse context.

5) Pragmatic Analysis:

Pragmatic analysis concerned with the overall communicative and social content and its effect on interpretation. It means abstracting or deriving the meaningful use of language in situations. In this analysis, what was said is reinterpreted on what it truly meant. It contains deriving those aspects of language which necessitate real world knowledge.

E.g., "close the window?" should be interpreted as a request instead of an order.

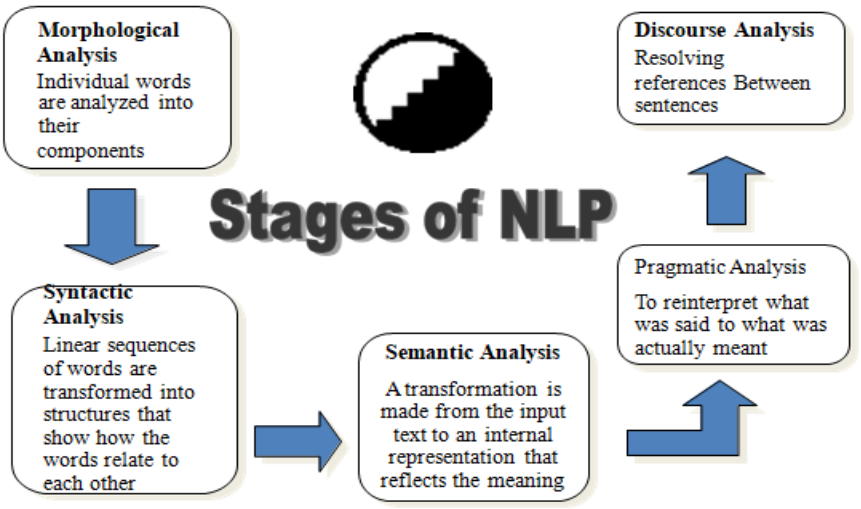


Figure 1.6.2 Stages of NLP

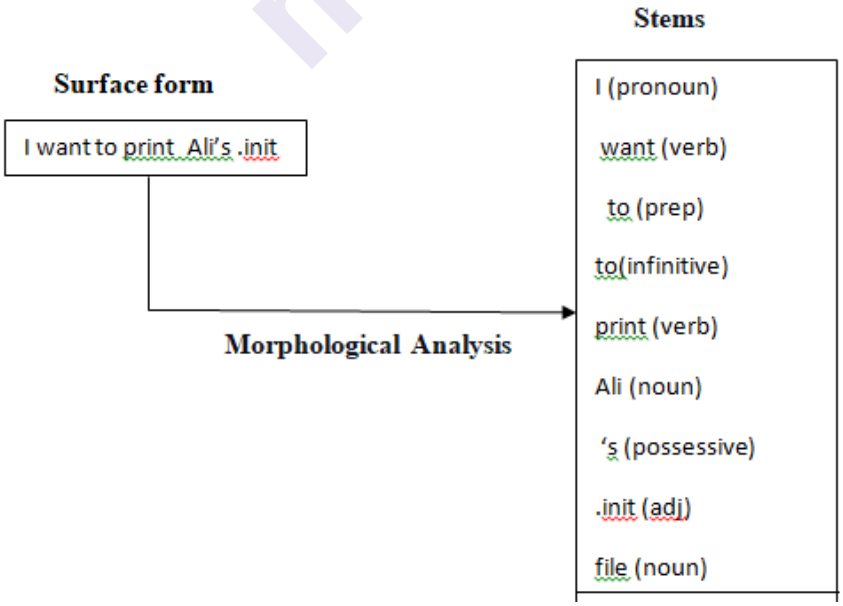
Morphological Analysis:

Consider we have an English interface to an operating system and the following sentence is typed: "I want to print Bill's .init file ".

Morphological analysis must do the following things:

- Pull apart the word "Bill's" into proper noun "Bill" and the possessive suffix "'s".
- Recognize the sequence ".init" as a file extension that is functioning as an adjective in the sentence.

This process will usually assign syntactic categories to all the words in the sentence. Consider the word "prints". This word is either a plural noun or a third person singular verb (he prints).



Syntactic analysis:

This method examines the structure of a sentence and performs detailed analysis of the sentence and semantics of the statement. In order to perform this, the system is expected to have through knowledge of the grammar of the language. The basic unit of any language is sentence, made up of group of words, having their own meanings and linked together to present an idea or thought. Apart from having meanings, words fall under categories called parts of speech. In English languages, there are eight different parts of speech. They are nouns, pronoun, adjectives, verb, adverb, prepositions, conjunction and interjections.

In English language, a sentence S is made up of a noun phrase (NP) and a verb phrase (VP), i.e.

S=NP+VP

The given noun phrase (NP) normally can have an article or delimiter (D) or an adjective (ADJ) and the noun (N), i.e.

NP=D+ADJ+N

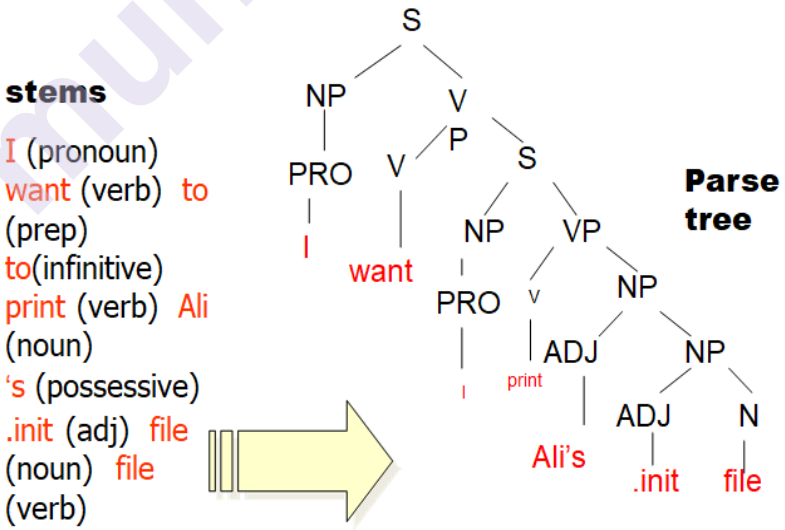
Also a noun phrase may have a prepositional phrase (PP) which has a preposition (P), a delimiter (D) and the noun (N), i.e.

PP=D+P+N

The verb phrase (VP) has a verb (V) and the object of the verb. The object of the verb may be a noun (N) and its determiner, i.e.

VP=V+N+D

These are some of the rules of the English grammar that helps one to construct a small parser for NLP.



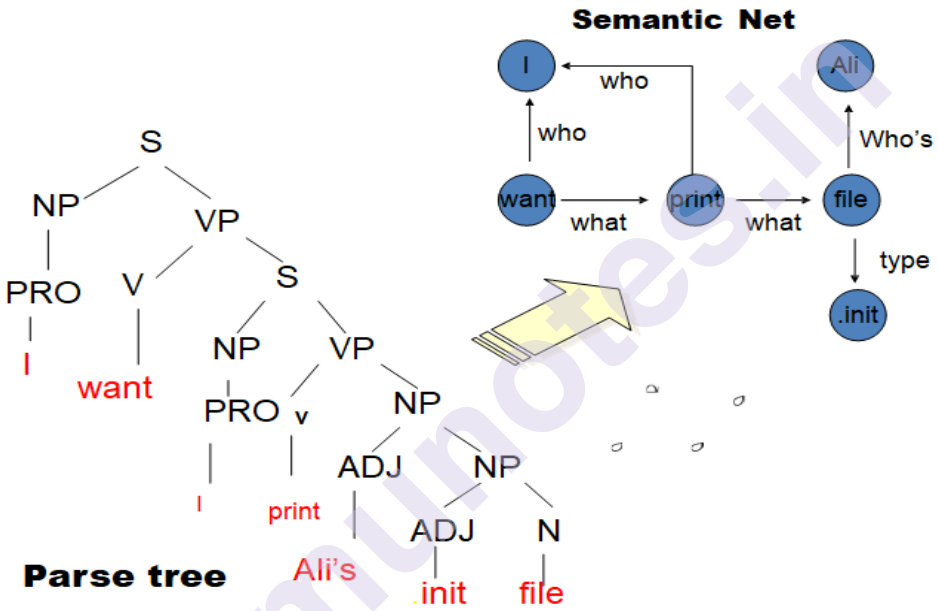
Syntactic analysis must exploit the results of morphological analysis to build a structural description of the sentence. The goal of this process, called parsing, is to convert the flat list of words that forms the sentence into a structure that defines the units that are represented by that flat list. The important thing here is that a flat sentence has been converted into a hierarchical structure and that the structures correspond to meaning units

when semantic analysis is performed. Reference markers are shown in the parenthesis in the parse tree. Each one corresponds to some entity that has been mentioned in the sentence.

Semantic Analysis:

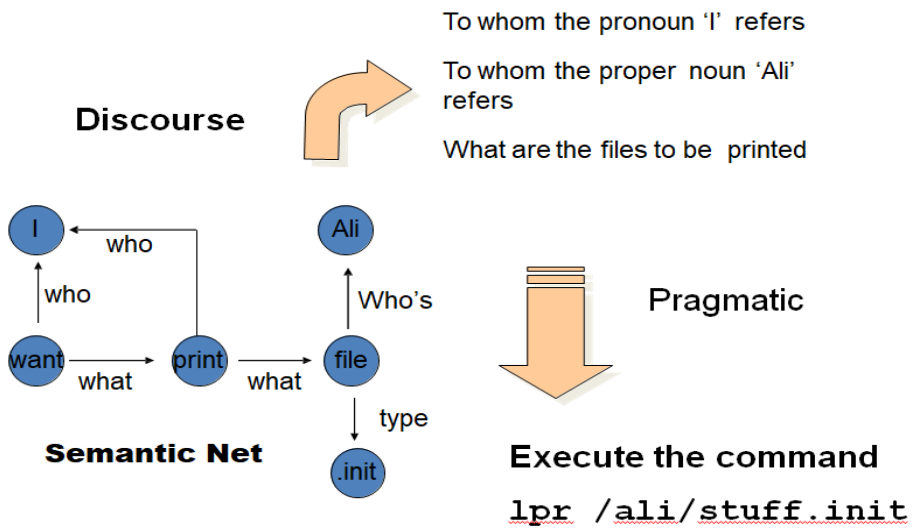
The structures created by the syntactic analyzer assigned meanings. Also, a mapping made between the syntactic structures and objects in the task domain. Moreover, Structures for which no such mapping possible may reject

Semantic analysis must do two important things:
It must map individual words into appropriate objects in the knowledge base or database It must create the correct structures to correspond to the way the meanings of the individual words combine with each other.



Discourse Integration:

The meaning of an individual sentence may depend on the sentences that precede it. And also, may influence the meanings of the sentences that follow it
Specifically we do not know whom the pronoun "I" or the proper noun "Bill" refers to. To pin down these references requires an appeal to a model of the current discourse context, from which we can learn that the current user is USER068 and that the only person named "Bill" about whom we could be talking is USER073.Once the correct referent for Bill is known, we can also determine exactly which file is being referred to.



Pragmatic Analysis

Moreover, The structure representing what said reinterpreted to determine what was actually meant. The final step toward effective understanding is to decide what to do as a results. One possible thing to do is to record what was said as a fact and be done with it. For some sentences, whose intended effect is clearly declarative, that is precisely correct thing to do. But for other sentences, including this one, the intended effect is different. We can discover this intended effect by applying a set of rules that characterize cooperative dialogues. The final step in pragmatic processing is to translate, from the knowledge based representation to a command to be executed by the system.

Results of each of the main processes combine to form a natural language system.

The results of the understanding process are lpr /ali/stuff.init. All of the processes are important in a complete natural language understanding system. Not all programs are written with exactly these components. Sometimes two or more of them collapsed. Doing that usually results in a system that is easier to build for restricted subsets of English but one that is harder to extend to wider coverage

1.7 CHALLENGES IN NLP

There are a number of challenges of natural language processing because natural language is ever-evolving and always somewhat ambiguous.

The major challenges around NLP that one must be aware of are as follows:

1. **Language differences.**
   Different languages have not only different sets of vocabulary, but also different types of phrasing, different modes of inflection, and different cultural norms. You can resolve this issue with the help of

"universal" models that can move at least part of what you've learned to other languages. However, you have to spend time updating your NLP system for each additional language.

2. Training data

NLP is all about studying language to better understand it. For a person to be proficient, it is essential to spend a substantial amount of time listening, reading, and understanding it, Even the best AI must also spend a significant amount of time reading, listening to, and using a language. The training data given to an NLP system determines its capabilities. If you feed the system bad or questionable data, it's going to learn the wrong things, or learn in an inefficient way.

3. Development time.

You also need to think about the development time for an NLP system. The total time taken to develop an NLP system is higher. AI evaluates the data points to process them and use them accordingly. With a shared deep network and many GPUs working together, the training time can be reduced by a few hours. The pre-existing NLP technologies can help in developing the product from scratch.

4. Phrasing ambiguities.

Many times it's hard even for another human being to parse out what someone means when they say something ambiguous. There may not be a clear, concise meaning to be found in analysis of their words. In order to resolve this, an NLP system must be able to seek context that can help it understand the phrasing. It may also need to ask the user for clarity.

5. Misspellings.

It is common for humans to make spelling mistakes but it can be easily interpreted by human being. But for a machine, misspellings can be harder to detect it, hence it is essential to employ NLP technology to progress and identify the misspellings.

6. Words with multiple meanings.

No language is perfect, and most languages have words that can have multiple meanings, depending on the situation. For example, a user who asks, "how are you?" has a totally different aim than a user who asks something like "how do I add a new debit card?" Good NLP tools should be able to discriminate between these utterances with the help of context.

7. Phrases with multiple intentions.

Some words and questions actually have multiple meaning, so your NLP system can't oversimplify the situation by interpreting only one of them. For example, a user may say to your chatbot with something like, "I need to cancel my previous order and update my card on file." Your AI needs to be able to distinguish these intentions separately.

8. **False positives and uncertainty.**
   A false positive occurs when an NLP notices a word that should be intelligible or addressable, but cannot be sufficiently answered. The idea is to develop an NLP system that can recognize its own limitations, and use questions to clear up the ambiguity.

9. **Keeping a conversation moving.**
   Many current NLP applications are built on communication between a human and a machine. Accordingly, your NLP AI needs to be able to keep the conversation moving, providing more questions to collect more information or data and always pointing toward a solution.

1.8 APPLICATIOS OF NLP

- **Machine Translation**

  Machine translation is basically used to convert text or speech from one natural language to another natural language. Machine translation, an integral part of Natural Language Processing where translation is done from source language to target language preserving the meaning of the sentence. Example: Google Translator

- **Information Retrieval**

  It refers to the human-computer interaction (HCI) that happens when we use a machine to search a body of information for information objects (content) that match our search query. A Person's query is matched against a set of documents to find a subset of 'relevant' document. Examples: Google, Yahoo, Altavista, etc.

- **Text Categorization**

  Text categorization (also known as text classification or topic spotting) is the task of automatically sorting a set of documents into categories (clusters).

Uses of Text Categorization

- Filtering of content
- Spam filtering Identification of document content
- Survey coding

- **Information Extraction** -

Identify specific pieces of information in unstructured or semi-structured textual document. Transform unstructured information in a corpus of documents or web pages into a structured database.

Applied to different types of text:

- Newspaper articles
- Web pages
- Scientific articles
- Newsgroup messages
- Classified ads
- Medical notes

- **Grammar Correction-**

In word processor software like MS-word, NLP techniques are widely used for spelling correction & grammar check.

- **Sentiment Analysis-**
  Sentiment Analysis is also known as opinion mining. It is mainly used on the web to analyse the behaviour, attitude, and emotional state of the sender. This application is implemented through a combination of NLP) and statistics by assigning the values to the text (natural, positive or negative), identify the mood of the context (sad, happy, angry, etc.)

- **Question-Answering systems-**
  Question Answering focuses on constructing systems that automatically answer the questions asked by humans in a natural language. It presents only the requested information instead of searching full documents like search engine. The basic idea behind the QA system is that the users just have to ask the question and the system will retrieve the most appropriate and correct answer for that question.

E.g.

Q. "What is the birth place of Shree Krishna?
A. Mathura

- **Spam Detection**
  To detect unwanted e-mails getting to a user's inbox, spam detection is used

- **Chatbot**
  Chatbot is one of the most important applications of NLP. It is used by many companies to provide the customer's chat services.

- **Speech Recognition-**
  Speech recognition is used for converting spoken words into text. It is used in applications, such as mobile, home automation, video recovery, dictating to Microsoft Word, voice biometrics, voice user interface, and so on.

- **Text summarization**
  This task aims to create short summaries of longer documents while retaining the core content and preserving the overall meaning of the text.

1.9 SUMMARY

In this chapter we studied what is natural language processing (NLP), history of NLP, generic NLP system, levels of NLP, and knowledge and ambiguity in NLP. Here we also discussed the stages in NLP, challenges and applications of NLP.

1.10 MULTIPLE CHOICE QUSETION ANSWERS

i. What is the field of Natural Language Processing (NLP)?
   a) Computer Science
   b) Artificial Intelligence
   c) Linguistics
   d) All of the mentioned

Ans : d

ii. What is the main challenge/s of NLP?
   a) Handling Ambiguity of Sentences
   b) Handling Tokenization
   c) Handling POS-Tagging
   d) All of the mentioned

Ans : a

iii. Choose form the following areas where NLP can be useful.
   a) Automatic Text Summarization
   b) Automatic Question-Answering Systems
   c) Information Retrieval
   d) All of the mentioned

Ans : d

iv. Which of the following includes major tasks of NLP?
   a) Automatic Summarization
   b) Discourse Analysis
   c) Machine Translation
   d) All of the mentioned

Ans : d

v. Natural language processing is divided into the two subfields of -
   a) symbolic and numeric
   b) understanding and generation
   c) algorithmic and heuristic
   d) None of the above.

Ans : b

1.11 TRUE OR FALSE

i. NLP is concerned with the interactions between computers and human (natural) languages.

True

ii. Machine Translation converts human language to machine language

False

iii. Modern NLP algorithms are based on machine learning, especially statistical machine learning.

True

1.12 SAMPLE QUESTIONS

1. What is NLP? Discuss the different stages in NLP
2. What are the different types of ambiguities present in NLP? How to remove it
3. Explain generic NLP system with a diagram
4. State and explain the different applications and challenges in
5. Explain Components of NLP

LIST OF REFERENCES

1. https://www.guru99.com/nlp-tutorial.html
2. https://www.datascienceprophet.com/different-levels-of-nlp/
3. https://www.slideshare.net/HareemNaz/natural-languageprocessing
4. "Natural Language Processing", Staredu Solutions.
5. Dan Jurafsky and James Martin. "Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition", Prentice Hall, Second Edition, 2009.
6. http://www.deepsky.com/~merovech/voynich/voynich_manchu_reference_materials/PDFs/jurafsky_martin.pdf
7. Christopher D.Manning and Hinrich Schutze, — Foundations of Statistical Natural Language Processing —, MIT Press, 1999.
8. https://www.cs.vassar.edu/~cs366/docs/Manning_Schuetze_StatisticalNLP.pdf
9. https://slideplayer.com/slide/4211188/

❄❄❄❄❄❄❄

2

WORD LEVEL ANALYSIS - I

Unit Structure

2.1 OBJECTIVES

The objective of this module is to learn about morphology, morphology analysis, concentrating mainly on survey of English Morphology. After that we will discuss the Inflectional Morphology & Derivational Morphology. This is followed by a discussion of the difference of Inflectional and Derivational Morphology. At last we will discuss the Stemming and Lemmatization.

2.0.1 MORPHOLOGY ANALYSIS

Word is a letter or group of letters that has meaning when spoken, signed or written. Words are orthographic tokens separated by white space. Many languages the distinction between words and sentences is less clear. In Chinese, Japanese languages no white space between words

Example: nowhitespace ~ no white space/no whites pace/now hit esp ace

In Turkish language words could represent a complete “sentence”

Example: uygarlastiramadiklarimizdanmissinizcasina

Morphology is a study of word formation. Some words can be divided into two parts but still have meaning. Example: handsome (hand + some). Many words have meaning in themselves but some words have meaning only when used with other words. Example: sleeping, colourful. Some words can stand alone. Example: free, cat, run tree, table, local etc. Words parts must be combined in the correct way and systematically.

Morphology deals with the syntax of complex words and parts of words, also called morphemes, as well as with the semantics of their lexical meanings. Understanding how words are formed and what semantic

properties they convey through their forms enables human beings to easily recognize individual words and their meanings in discourse.

Morpheme is defined as “minimal meaning-bearing unit in a language"
Morphology handles the formation of words by using morphemes
base form (stem , lemma), e.g., believe
affixes (suffixes, prefixes, infixes), e.g., un-, -able, -ly

Morphological parsing is the task of recognizing the morphemes inside a word e.g., hands, foxes, children and it’s important for many tasks machine translation, information retrieval, ,in parsing, and text simplification, etc

2.0.2 SURVEY OF ENGLISH MORPHOLOGY

Morphology concerns the structure of words. Words are assumed to be made up of morphemes, which are the minimal information carrying unit. So for example the word dog consists of a single morpheme (the morpheme dog) while the word cats consists of two: the morpheme cat and the morpheme-s.

Consider a word like: "unhappyness". This has three parts as shown in figure 2.1.2

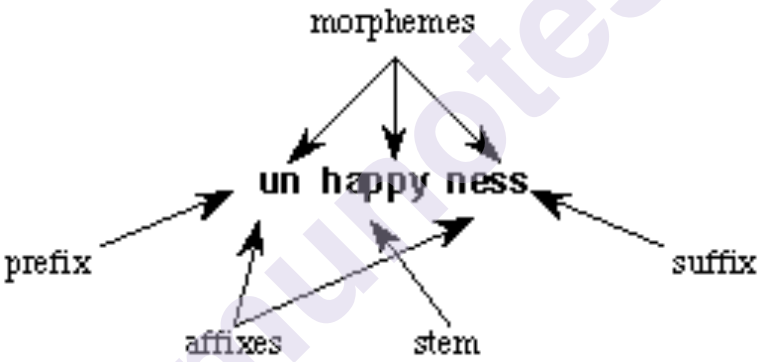
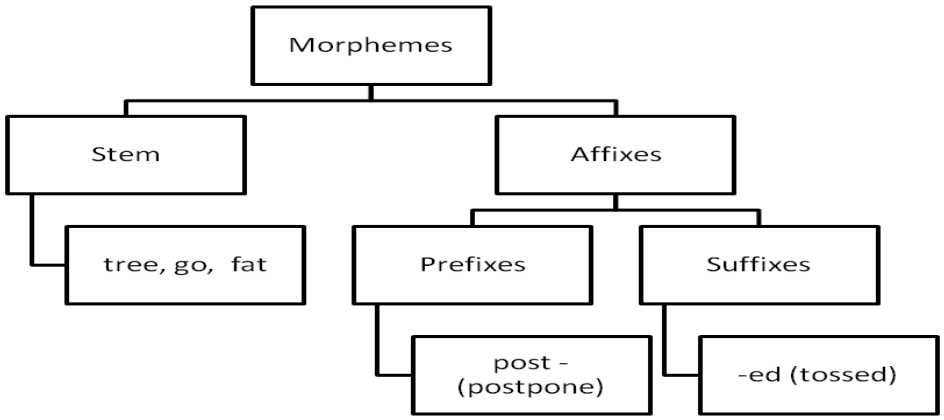


Figure 2.1.2

There are three morphemes, each carrying a certain amount of meaning. un means "not", while ness means "being in a state or condition". Happy is a free morpheme because it can appear on its own (as a "word" in its own right). Bound morphemes have to be attached to a free morpheme, and so cannot be words in their own right. Thus you can't have sentences in English such as "Jason feels very un ness today".

We can usefully divide morphemes into two broad classes of morphemes: **stems and affixes.**

- The stem is the “main” morpheme of the word, supplying the main meaning
- The affixes add “additional” meanings of various kinds.



Affixes are further divided into prefixes, suffixes, infixes, and circumfixes.

- prefixes precede the stem,
- suffixes follow the stem,
- circumfixes do both,
- infixes are inserted inside the stem.

For example, the word cats is composed of a stem cat and the suffix -s. The word undo is composed of a stem do and the prefix un-. Good examples of circumfixes are e.g Unreachable, Unbelievable. Infixes, in which a morpheme is inserted in the middle of a word, occur very commonly for example in the Philipine language Tagalog. For example the affix um, which marks the agent of an action, is infixed to the Tagalog stem hingi "borrow" to produce humingi (ask for or to demand).

A word can have more than one affix. For example, the word rewrites has the prefix re-, the stem write, and the suffix -s. The word unbelievably has a stem (believe) plus three affixes (un-, -able, and -ly). There are many ways to combine morphemes to create words.

Four of these methods are common and play important roles in speech and language processing: inflection, derivation, compounding, and cliticization.

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. For example, English has the inflectional morpheme -s for marking the plural on nouns, and the inflectional morpheme –ing for marking the present tense on verbs. e.g. clean (verb), clean-ing (verb)

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 the verb delight can take the

derivational suffix -ful to produce the adjective delightful. e.g delight (verb), delight-ful (adj)

Compounding is the combination of multiple words stems together. In English compounds can be written together, for example, notebook, bookstore, fireman etc. For example the noun doghouse is the concatenation of the morpheme dog with the morpheme house.

Finally, cliticization is the combination of a word stem with a clitic. A clitic is a morpheme that acts syntactically like a word, but is reduced in form and attached (phonologically and sometimes orthographically) to another word. For example the English morpheme've in the word I've is a clitic. That is different words from different syntactic categories, e.g. I've = I + have.

2.0.3 INFLECTIONAL MORPHOLOGY

Inflection is a morphological process that adapts existing words so that they function effectively in sentences without changing the category of the base morpheme.

Inflectional affixes are only suffixes which express grammatical features such as singular/plural, past/present forms of verbs. The study of such morphemes is called inflectional morphemes. Examples: bigger (big+er), loves (love+s).

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.

English nouns have only two kinds of inflection: an affix that marks plural and an affix that marks possessive. For example, many English nouns can either appear in the singular form, or take a plural suffix. Here are examples of the regular plural suffix -s (also spelled -es), and irregular plurals and sometimes -x (box/boxes). Nouns ending in -y preceded by a consonant change the -y to -i (butterfly/butterflies).

Table 2.1.3.1

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

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 a lone apostrophe after regular plural nouns (llamas') and some names ending in -s or -z (Euripides' comedies).

English verbal inflection is more complicated than nominal inflection.
First, English has three kinds of verbs;

- main verbs, (eat, sleep, impeach),
- modal verbs (can, will, should),
- primary verbs (be, have, do).

Here we will mostly be concerned with the main and primary verbs, because it is these that have inflectional endings. Of these verbs a large class are regular, that is to say all verbs of this class have the same endings marking the same functions. These regular verbs (e.g. walk, or inspect) have four morphological forms. 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. As discussed earlier, a productive class is one that automatically includes any new words that enter the language. For example the recently-created verb fax (My mom faxed me the note from cousin Everett) takes the regular endings -ed, -ing, -es. (Note that the -s form is spelled faxes rather than faxs).

The table 2.0.3.2 shows the morphological forms for regular verbs.

Table 2.0.3.2

Morphological Form Classes	**Regularly Inflected Verbs**			
stem	**talk**	**merge**	**cry**	**map**
-s form	**talks**	**merges**	**cries**	**maps**
-ing participle	**talking**	**merging**	**crying**	**mapping**
Past form or -ed participle	**talked**	**merged**	**cried**	**mapped**

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 there are only about 250 irregular verbs, not counting auxiliaries), this class includes most of the very frequent verbs of the language. The table 2.4.3 shows some sample irregular forms. Note that 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 change at all (cut/cut).

The table 2.1.3.3 shows the morphological forms for irregular verbs.

Table 2.1.3.3

Morphological Form Classes	**Irregularly Inflected Verbs**		
stem	**eat**	**catch**	**put**
***-s* form**	**eats**	**catches**	**puts**
***-ing* participle**	**eating**	**catching**	**putting**
Past form	**ate**	**caught**	**put**
***-ed* participle**	**eaten**	**caught**	**put**

The -s form is used in the "habitual present" form to distinguish the third-person singular ending (She jogs every Tuesday) from the other choices of person and number (I/you/we/they jog every Tuesday). The stem form is used in the infinitive form, and also after certain other verbs (I'd rather walk home, I want to walk home). The -ing participle is used in the progressive construction to mark present or ongoing activity (It is raining), or when the verb is treated as a noun; this particular kind of nominal use of a verb is called a gerund use: Fishing is fine if you live near water. The -ed participle is used in the perfect construction (He's eaten lunch already) or the passive construction (The verdict was overturned yesterday).

In addition to noting which suffixes can be attached to which stems, we need to capture the fact that a number of regular spelling changes occur at these morpheme boundaries. For example, a single consonant letter is doubled before adding the -ing and -ed suffixes (beg/begging/begged). If the final letter is "c", the doubling is spelled "ck" (picnic/picnicking/picnicked). If the base ends in a silent -e, it is deleted before adding -ing and -ed (merge/merging/merged). Just as for nouns, the -s ending is spelled -es after verb stems ending in -s (toss/tosses), -z, (waltz/waltzes) -sh, (wash/washes) -ch, (catch/catches) and sometimes -x (tax/taxes). Also like nouns, verbs ending in –y preceded by a consonant change the -y to -i (try/tries).

2.0.4 DERIVATIONAL MORPHOLOGY

Derivation is concerned with the way morphemes are connected to existing lexical forms as affixes.

English inflection is relatively simple compared to other languages, derivation in English is quite complex. Recall that 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.

The 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 (computer-ize → computerization).

The table 2.0.4.1 shows the formation of new nouns, often from verbs or adjectives.

Table 2.0.4.1

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

Adjectives can also be derived from nouns and verbs. The table 2.0.4.2 shows adjectives derived from the verbs or noun.

Table 2.0.4.2

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 a number of reasons. One is that 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. Thus we can't say *eatation or *spellation (we use an asterisk (*) to mark "non-examples" of English). Another is that there are subtle and complex meaning differences among nominalizing suffixes. For example sincerity has a subtle difference in meaning from sincereness.

2.0.5 DIFFERENCE OF INFLECTIONAL AND DERIVATIONAL MORPHOLOGY

1) Inflectional morphemes never change the grammatical category (part of speech) of a word.

   For example, tall and taller are both adjectives. The inflectional morpheme -er (comparative marker) simply produces a different version of the adjective tall.

   derivational morphemes often change the part of speech of a word. Thus, the verb read becomes the noun reader when we add the derivational morpheme -er.

   It is simply that read is a verb, but reader is a noun

   example:-For example, such derivational prefixes as re- and un- in English generally do not change the category of the word to which they are attached. Thus, both happy and unhappy are adjectives, and both fill and refill are verbs, for example. The derivational suffixes -hood and -dom, as in neighborhood and kingdom, are also the typical examples of derivational morphemes that do not change the grammatical category of a word to which they are attached.

2) When a derivational suffix and an inflectional suffix are added to the same word, they always appear in a certain relative order within the word

   That is, inflectional suffixes follow derivational suffixes. Thus, the derivational (-er) is added to read, then the inflectional (-s) is attached to produce readers.

   Similarly, in organize– organizes the inflectional -s comes after the derivational -ize. When an inflectional suffix is added to a verb, as with organizes, then we cannot add any further derivational suffixes It is impossible to have a form like organizesable, with inflectional -

s after derivational - able because inflectional morphemes occur outside derivational morphemes and attach to the base or stem.

3) A third point worth emphasizing is that certain derivational morphemes serve to create new base forms or new stems to which we can attach other derivational or inflectional affixes. For example, we use the derivational -atic to create adjectives from nouns, as in words like systematic and problematic.

2.0.6 STEMMING AND LEMMATIZATION

In natural language processing, there may come a time when you want your program to recognize that the words "ask" and "asked" are just different tenses of the same verb. This is the idea of reducing different forms of a word to a core root. Words that are derived from one another can be mapped to a central word or symbol, especially if they have the same core meaning.

"When we are running a search, we want to find relevant results not only for the exact expression we typed on the search bar, but also for the other possible forms of the words we used. For example, it's very likely we will want to see results containing the form "shoot" if we have typed "shoots" in the search bar."

This can be achieved through two possible methods: stemming and lemmatization. The aim of both processes is the same: reducing the inflectional forms of each word into a common base or root. However, these two methods are not exactly the same.

Example: Affectation, affects, affections, affected, affection, affecting -> all of these came from single root work affect.

Stemming

Stemming algorithms work by cutting off the end or the beginning of the word, taking into account a list of common prefixes and suffixes that can be found in an inflected word.

This indiscriminate cutting can be successful on some occasions, but not always, and that is why we affirm that this approach presents some limitations.

Form	Suffix	Stem
studies	-es	studi
studying	-ing	study
niñas	-as	niñ
niñez	-ez	niñ

Ex: writing □ writ + ing (or write + ing)

Lemmatization

Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma

Example: Lemmatization will map gone, going and went->Go.

- Output of Lemmatization is proper word
- If confronted with the token saw, stemming might return just s, whereas lemmatization would attempt to return either see or saw depending on whether the use of the token was as a verb or a noun.

Form	Morphological information	Lemma
studies	Third person, singular number, present tense of the verb study	study
studying	Gerund of the verb study	study
niñas	Feminine gender, plural number of the noun niño	niño
niñez	Singular number of the noun niñez	niñez

How do they work?

- **Stemming:** there are different algorithms that can be used in the stemming process, but the most common in English is Porter stemmer. The rules contained in this algorithm are divided in five different phases numbered from 1 to 5. The purpose of these rules is to reduce the words to the root.
- **Lemmatization:** the key to this methodology is linguistics. To extract the proper lemma, it is necessary to look at the morphological analysis of each word. This requires having dictionaries for every language to provide that kind of analysis.

Difference between Stemming and Lemmatization:

Sr.no	Stemming	Lemmatization
1	Stemming is used to group words with a similar basic meaning together	Lemmatization concept is used to make dictionary or WordNet kind of dictionary
2	Accuracy is less	Accuracy is more as compared to stemming.
3	Stemming is preferred when the meaning of the word is not important for analysis. Example: Spam detection	Lemmatization would be recommended when the meaning of the word is not important for analysis. Example: Question Answer

4	Stemming usually operates on single word without knowledge of the context	Lemmatization usually considers words and the context of the word in the sentence
5	When we convert any word into root form then stemming may create the non-existence meaning of word.	Lemmatization always gives the dictionary meaning word while converting into root form
6	Stemming for the word "studies" is studi	Lemma for the word "studies" is study

2.1 SUMMARY

In this chapter we studied concept of morphology, morphology analysis in detailed, and survey of English Morphology. Morphological parsing is the process of finding the constituent morphemes in a word. We discussed the Inflectional morphology & Derivational morphology, English mainly uses prefixes and suffixes to express inflectional and derivational morphology. English inflectional morphology is relatively simple and includes person and number agreement (-s) and tense markings (-ed and -ing). English derivational morphology is more complex and includes suffixes like -ation, -ness, -able as well as prefixes like co- and re-. Here we also discussed stemming and Lemmatization.

2.2 MULTIPLE CHOICE QUSETION ANSWERS

i. In linguistic morphology, _____ is the process for reducing inflected words to their root form.

a) Rooting
b) Stemming
c) Text-Proofing
d) Both a & b

Ans : b

ii. _____ techniques looks at the meaning of the word

a) Stemming
b) Stop Word
c) Leematization
d) Marphological Analysis

Ans : c

iii. _____is the study of the internal structure of words, and the rules by which words are formed
a) Syntax
b) Semantics
c) Pragmatics
d) Morphing

Ans : d

iv. Lemma for the word "studies" is_____
a) studies
b) ies
c) studi
d) study

Ans : d

v. Stemming for the word "studies" is_____
a) studies
b) ies
c) studi
d) study

Ans : c

TRUE OR FALSE

i. Derivational morphemes form new lexemes conveying new meanings
True

ii. Lemmatization involves resolving words to their dictionary form.
True

iii. Inflectional morphemes change the grammatical category (part of speech) of a word.
False

SAMPLE QUESTIONS

1. Define morpheme, stem and affixes.
2. Explain stemming and lemmatization with example
3. Explain with examples Inflectional morphology & Derivational morphology
4. State differences between inflectional and derivational morphemes

2.3 LIST OF REFERENCES

1. "Natural Language Processing", Staredu Solutions.
2. Dan Jurafsky and James Martin. "Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition", Prentice Hall, Second Edition, 2009.
3. http://www.deepsky.com/~merovech/voynich/

   voynich_manchu_reference_materials/PDFs/jurafsky_martin.pdf
4. Christopher D.Manning and Hinrich Schutze, — Foundations of Statistical Natural Language Processing —, MIT Press, 1999.
5. https://cl-illc.github.io/nlp1/resources/slides/Morphology-notes.pdf
6. https://www.cs.vassar.edu/~cs395/docs/3.pdf

❄❄❄❄❄❄❄

3

WORD LEVEL ANALYSIS - II

Unit Structure

3.0 OBJECTIVES

The objective of this module is to learn about Regular expression (RE), Finite Automata, Finite state morphological parsing, how to build finite state lexicon, and Finite State Transducers (FST). After that we will discuss the Morphological Parsing with FST. This is followed by a discussion of Lexicon free FST Porter Stemmer. Finally we will discuss N –Grams- N-gram language model.

3.1 REGULAR EXPRESSION

Regular expressions are defined as a sequence of characters that are mainly used to find substrings in a given string. A regular expression (RE) is a language for specifying text search strings. It helps us to match or extract other strings or sets of strings, with the help of a specialized syntax present in a pattern. For Example, extracting all hashtags from a tweet, getting phone numbers, email ID etc from large unstructured text content. Regular Expressions are used in various tasks such as Data pre-processing, Text feature Engineering, Data Extraction, Rule-based information Mining systems, Pattern Matching, Web scraping, etc.

Properties of Regular Expressions

Some of the important properties of Regular Expressions are as follows:

1. Regular Expression (RE) is a formula in a special language, which can be used for specifying a sequence of symbols, simple classes of strings. In simple words, we can say that regular Expression is an algebraic notation for characterizing a set of strings.
2. Regular expression requires two things, one is the pattern that we want to search and the other is a corpus of text or a string from which we need to search the pattern. For example, the Unix command-line tool grep takes a regular expression and returns every line of the input document that matches the expression

Regular Expression can be defined in the following manner:

1. ε is a Regular Expression, which indicates that the language is having an empty string.
2. φ is a Regular Expression which denotes that it is an empty language.
3. If X and Y are Regular Expressions, then the following expressions are also regular.

    - X, Y
    - X.Y(Concatenation of XY)
    - X+Y (Union of X and Y)
    - X*, Y* (Kleen Closure of X and Y)

4. If a string is derived from the above rules then that would also be a regular expression.

How can Regular Expressions be used in NLP?

In NLP, we can use Regular expressions at many places such as,

1. To validate data fields. For example, URLs, dates, email address, abbreviations, etc.
2. To filter a particular text from the whole corpus. For example, disallowed websites, spam etc.
3. To identify particular strings in a text. For example, token boundaries
4. To convert the output of one processing component into the format required for a second component

Basic Regular Expression Patterns

Brackets ([ ]):

They are used to specify a disjunction of characters.

For Examples,

/[A-Z]/ → matches an uppercase letter
/[a-z]/ → matches a lowercase letter
/[0–9]/ → matches a single digit

RE	Match	Example Patterns Matched
/ [A-Z] /	an uppercase letter	"we should call it 'Drenched Blossoms'"
/ [a-z] /	a lowercase letter	"my beans were impatient to be hoed!"
/ [0-9] /	a single digit	"Chapter 1: Down the Rabbit Hole"

For Examples,

/[cC]hirag/ → Chirag or chirag
/[xyz]/ → 'x', 'y', or 'z'
/[1234567890]/ → any digit

Here slashes represent the start and end of a particular expression.

Dash (-):

They are used to specify a range.

RE	Match	Example Patterns
/ [wW] oodchuck/	Woodchuck or woodchuck	"Woodchuck"
/ [abc] /	'a', 'b', *or* 'c'	"In uomini, in soldati"
/ [1234567890] /	any digit	"plenty of 7 to 5"

Caret (^):

They can be used for negation or just to mean ^.

For Examples,

/[^a-z]/ → not an lowercase letter
/[^Cc]/ → neither 'C' nor 'c'
/[^.]/ → not a period
/[c^]/ → either 'c' or '^'
/x^y/ → the pattern 'x^y'

RE	Match (single characters)	Example Patterns Matched
[^A-Z]	not an uppercase letter	"Oyfn pripetchik"
[^Ss]	neither 'S' nor 's'	"I have no exquisite reason for't"
[^\.]	not a period	"our resident Djinn"
[e^]	either 'e' or '^'	"look up ^ now"
a^b	the pattern 'a^b'	"look up a^ b now"

Question mark (?):

It marks the optionality of the previous expression.

For Examples,

/maths?/ → math or maths
/colou?r/ → color or colour

RE	Match	Example Patterns Matched
woodchucks?	woodchuck or woodchucks	"woodchuck"
colou?r	color or colour	"colour"

Period (.):

Used to specify any character between two expressions.

/beg.n/ → Match any character between which fits between beg and n such as begin, begun

RE	Match	Example Patterns
/beg.n/	any character between *beg* and *n*	begin, beg'n, begun

Anchors :

These are special characters that help us to perform string operations either at the beginning or at the end of text input. They are used to assert something about the string or the matching process. The most common anchors are the caret ^ and the dollar sign $.

i) Caret character '^'

It specifies the start of the string. For a string to match the pattern, the character followed by the '^' in the pattern should be the first character of the string.

For Examples,
^The: Matches a string that starts with 'The'

ii) Dollar character '$'

It specifies the end of the string. For a string to match the pattern, the character that precedes the '$' in the pattern should be the last character in the string.

For Examples,
end$: Matches a string that ends with 'end'
^The end$': Exact string match (starts and ends with 'The end')
roar: Matches a string that has the text roar in it.

Quantifiers:

They allow us to mention and control over how many times a specific character(s) pattern should occur in the given text.
The most common Quantifiers are: *, +, ? and { }

RE	Match
*	zero or more occurrences of the previous char or expression
+	one or more occurrences of the previous char or expression
?	exactly zero or one occurrence of the previous char or expression
{n}	*n* occurrences of the previous char or expression
{n,m}	from *n* to *m* occurrences of the previous char or expression
{n,}	at least *n* occurrences of the previous char or expression

For Examples,

abc*: matches a string that has 'ab' followed by zero or more 'c'.
abc+: matches 'ab' followed by one or more 'c'
abc?: matches 'ab' followed by zero or one 'c'
abc{2}: matches 'ab' followed by 2 'c'
abc{2, }: matches 'ab' followed by 2 or more 'c'
abc{2, 5}: matches 'ab' followed by 2 upto 5 'c'
a(bc)*: matches 'a' followed by zero or more copies of the sequence 'bc'

3.2 FINITE AUTOMATA

The term automata, derived from the Greek word "αὐτόματα. Automaton may be defined as an abstract self-propelled computing device that follows a predetermined sequence of operations automatically. An automaton having a finite number of states is called a Finite Automaton (FA) or Finite State automata (FSA).

The finite-state automaton is not only the mathematical device used to implement regular expressions, but also one of the most significant tools of computational linguistics. Variations of automata such as finite- state transducers, Hidden Markov Models, and *N*-gram grammars are important components of the speech recognition and synthesis, spell-checking, and information-extraction applications.

Mathematically, an automaton can be represented by a 5-tuple (Q, Σ, δ, q0, F),

where −

- Q is a finite set of states.
- Σ is a finite set of symbols, called the alphabet of the automaton.
- δ is the transition function
- q0 is the initial state from where any input is processed (q0 ∈ Q).
- F is a set of final state/states of Q (F ⊆ Q).

An RE is one way of describing finite state automata (FSA). An RE is one way of characterizing a particular kind of formal language called a regular language. Regular grammar is the equivalent ways of describing regular languages.

Relation between Finite Automata, Regular Grammars and Regular Expressions

- As we know that finite state automata are the theoretical foundation of computational work and regular expressions is one way of describing them.
- We can say that any regular expression can be implemented as FSA and any FSA can be described with a regular expression.
- On the other hand, regular expression is a way to characterize a kind of language called regular language. Hence, we can say that regular language can be described with the help of both FSA and regular expression.
- Regular grammar, a formal grammar that can be right-regular or left-regular, is another way to characterize regular language.

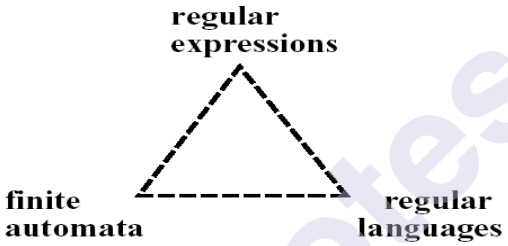


Figure 3.2.1 Relation between Finite Automata, Regular languages and Regular Expressions

Finite state automation is of two types:

- Deterministic Finite automation (DFA)
- Non-deterministic Finite Automation (NDFA)

1. Deterministic Finite automation (DFA)

It may be defined as the type of finite automation wherein, for every input symbol we can determine the state to which the machine will move. It has a finite number of states that is why the machine is called Deterministic Finite Automaton (DFA).

Mathematically, a DFA can be represented by a 5-tuple (Q, Σ, δ, q0, F), where −

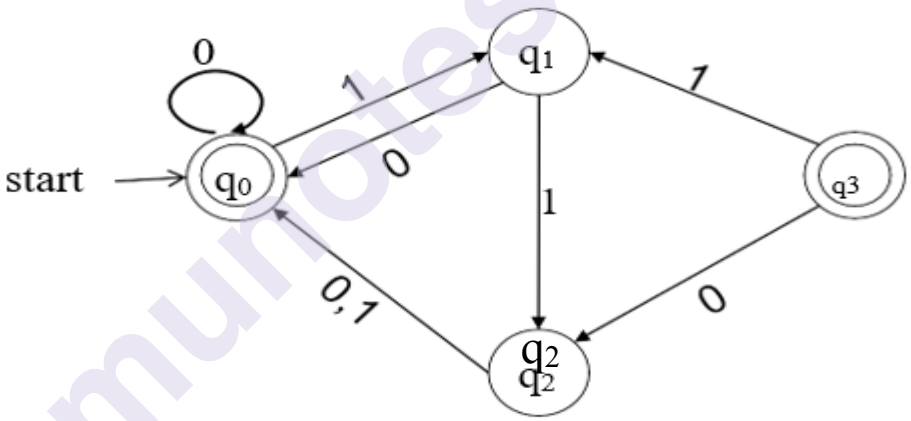
- Q is a finite set of states.
- Σ is a finite set of symbols, called the alphabet of the automaton.
- δ is the transition function where $\delta: Q \times \Sigma \rightarrow Q$.
- q0 is the initial state from where any input is processed ($q0 \in Q$).
- F is a set of final state/states of Q ($F \subseteq Q$).

Whereas graphically, a DFA can be represented by diagraphs called state diagrams where −

The states are represented by vertices.
The transitions are shown by labeled arcs.
The initial state is represented by an empty incoming arc.
The final state is represented by a double circle.

Construct the state diagram for the finite-state automata M= (Q, Σ, δ, q0, F),, where Q = {q_0, q_1, q_2, q_3}, Σ = {0,1}, F= {q_0, q_3} and the transition function δ is given in the following table.

State	0	1
q0	q_0	q_1
q1	q_0	q_2
q2	q_0	q_0
q3	q_2	q_1



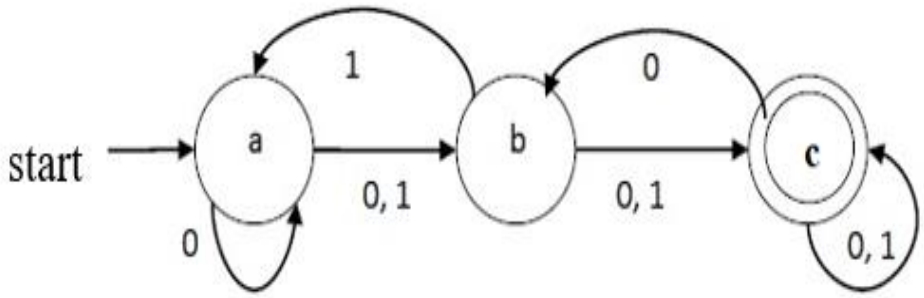
2. Non-deterministic Finite Automation (NDFA)

It may be defined as the type of finite automation where for every input symbol we cannot determine the state to which the machine will move i.e. the machine can move to any combination of the states. It has a finite number of states that is why the machine is called Non-deterministic Finite Automation (NDFA).

Mathematically, NDFA can be represented by a 5-tuple (Q, Σ, δ, q0, F), where −

- Q is a finite set of states.
- Σ is a finite set of symbols, called the alphabet of the automaton.
  δ :-is the transition function where δ: $Q \times \Sigma \rightarrow 2^Q$.
- q0 :-is the initial state from where any input is processed (q0 ∈ Q).
- F :-is a set of final state/states of Q (F ⊆ Q).

The finite state automaton M= (Q, Σ, q0, δ, F), where Q= {a, b, c }, Σ ={0,1}, F={c}, and δ is depicted graphically as follows:



State	0	1
a	a, b	b
b	c	a, c
c	b, c	c

3.3 FINITE STATE MORPHOLOGICAL PARSING

To know the structure about a word when we perform the morphological parsing for that word. Given a surface form (input form), e.g., "going" we might produce the parsed form: verb-go + gerund-ing. Morphological parsing can be done with the help of finite-state transducer. A morpheme is a meaning bearing unit of any language. For example, cats: has two morphemes, cat, -s.

The objective of the morphological parsing is to produce output lexicons for single input lexicon. Our aim is to take input forms like in the first column of Table 3.3.1 and produce the output like in second column. The second column contains the stem of each word as well as mixed morphological features. These features include, like N (noun), V (verb), specify additional information about the word stem, e.g., +N means that word is noun, +SG means singular, +PL for plural, etc.

Table 3.3.1 Mapping of input word to Morphemes.

Input Words	Morphological parsed output
cats	cat *+N +PL*
cat	cat *+N +SG*
cities	city *+N +PL*
geese	goose *+N +PL*
goose	goose *+V +3SG*
caught	catch *+V +PAST-Part*

To build morphological parser, we need at least following database:

1. Lexicon: List of stems, and affixes, plus additional information about them, like +N, +V.
2. Morphotactics rules: Rules about ordering of morphemes in a word, e.g. -ed is followed after a verb (e.g., worked, studied), un (undo) precede a verb, for example, unlock, untie, etc.
3. Orthographic rules (spelling): For combining morphemes, e.g., city+ -s gives cities and not citys.

3.4 BUILDING A FINITE-STATE LEXICON

A lexicon is a repository for words. They are grouped according to their main categories like noun, adjective, verb, adverb etc. They may be divided in to sub categories like regular-nouns, irregular-singular nouns, irregular-plural nouns. The simplest possible lexicon would consist of an explicit list of every word of the language (every word, i.e., including abbreviations ("AAA") and proper names ("Joy" or "Bebo")) as follows:

a, AAA, AA, Aachen, aardvark, aardwolf, aba, abaca, aback,

The simplest way to create a morphological parser put all possible words in to lexicon. As it's often impossible, for the various reasons, to list every word in the language, computational lexicons are usually structured with a list of each of the stems and affixes of the language together with a representation of the morphotactics that tells us how they can fit together. There are many ways to model morphotactics; one of the most common is the finite state automaton. A very simple finite-state model for English nominal inflection might look like Fig. 3.4.1.

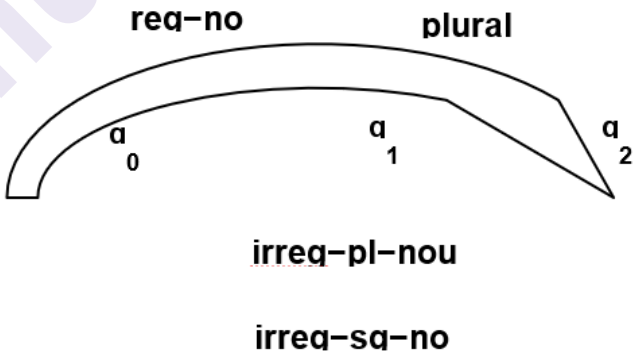


Figure 3.4.1 A finite-state automaton for English nominal inflection.

The FSA in Fig. 3.4.1 assumes that the lexicon includes regular nouns (reg-noun) that take the regular -s plural (e.g. dog, fox, cat, and aardvark). The lexicon also includes irregular noun forms that don't take -s, both singular irreg-sg-noun (goose, mouse) and plural irreg-pl-noun (geese, mice).

reg-noun	irreg-pl-noun	irreg-sg-noun	plural
fox cat aardvark	geese sheep mice	goose sheep mouse	-s

A similar model for English verbal inflection might look like Fig. 3.4.2

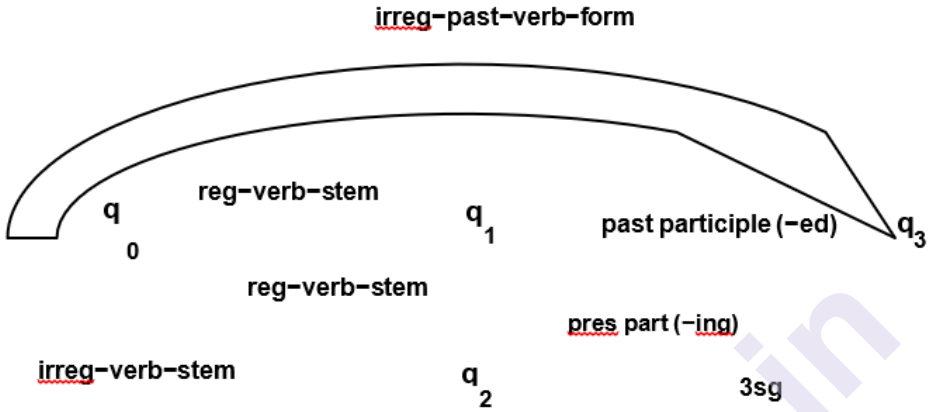


Figure 3.4.2 A finite-state automaton for English verbal inflection

This lexicon has three stem classes (reg-verb-stem, irreg-verb-stem, and irreg-past- verb-form), plus four more affix classes (-ed past, -ed participle, -ing participle, and third singular -s):

reg-verb-stem	irreg-verb-stem	irreg-past-verb	past	past-part	pres-part	3sg
walk fry talk impeach	cut speak sing	caught ate eaten sang	-ed	-ed	-ing	-s

English derivational morphology is significantly more complex than English inflectional morphology, and so automata for modeling English derivation tend to be quite complex. An initial hypothesis might be that adjectives can have an optional prefix (un-), an obligatory root (big, cool, etc.) and an optional suffix (-er, -est, or -ly). This might suggest the FSA in Fig. 3.4.3.

big, bigger, biggest, cool, cooler, coolest, coolly
happy, happier, happiest, happily red, redder, reddest
unhappy, unhappier, unhappiest, unhappily real, unreal, really
clear, clearer, clearest, clearly, unclear, unclearly

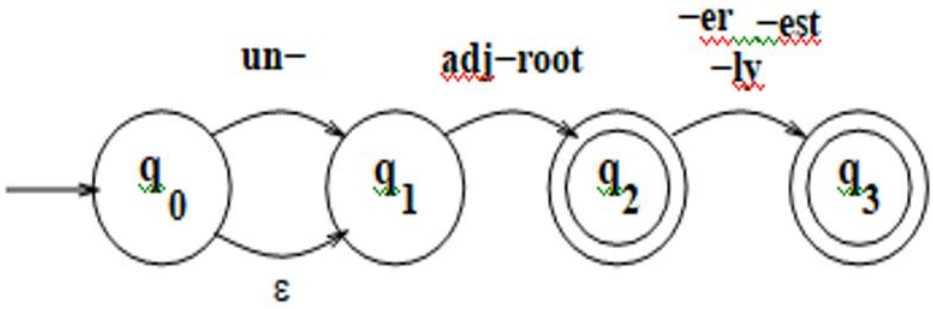


Figure 3.4.3 An FSA for a fragment of English adjective morphology

An FSA for another fragment of English adjective morphology is shown in Fig. 3.4.4.

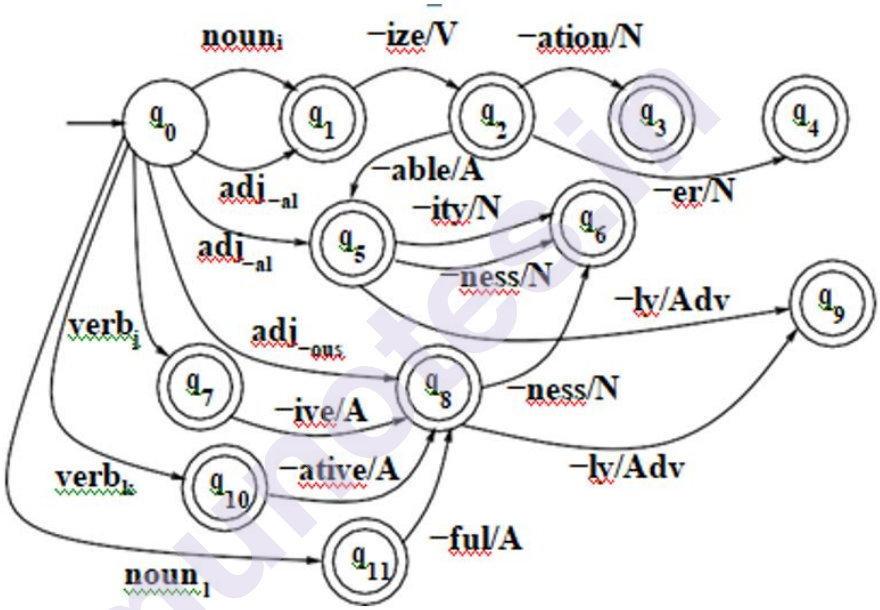


Figure 3.4.4 An FSA for another fragment of English derivational morphology.

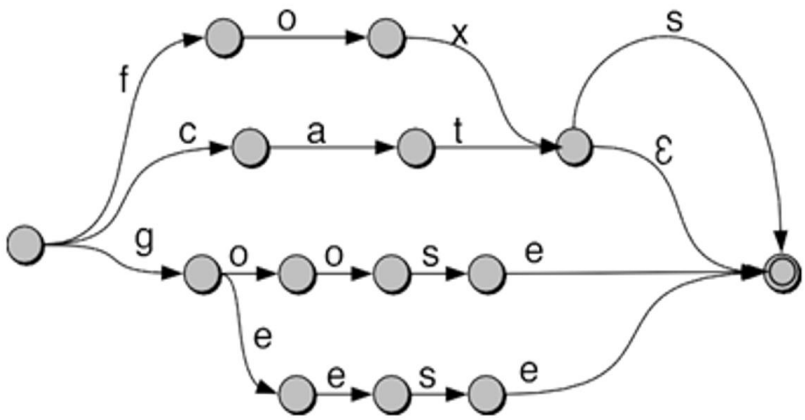


Figure 3.4.5 Expanded FSA for a few English nouns with their inflection

Expanded FSA for a few English nouns with their inflection is shown in Fig, 3.4.5. Note that this automaton will incorrectly accept the input foxs.

3.5 FINITE STATE TRANSDUCERS (FST)

FSAs can be used to recognise particular patterns but don't, by themselves, allow for any analysis of word forms. Hence for morphology, we use finite state transducers (FSTs) which allow the surface structure to be mapped into the list of morphemes. FSTs are useful for both analysis and generation, because the mapping is bidirectional. This approach is known as two-level morphology.

FST is a type of finite automaton which maps between two sets of symbols. We can visualize an FST as a two-tape automaton which recognizes or generates pairs of strings. We can do this by labelling each arc in the finite-state machine with two symbol strings, one from each tape. Fig 3.5.1 shows an example of an FST where each arc is labelled by an input and output string, separated by a colon.

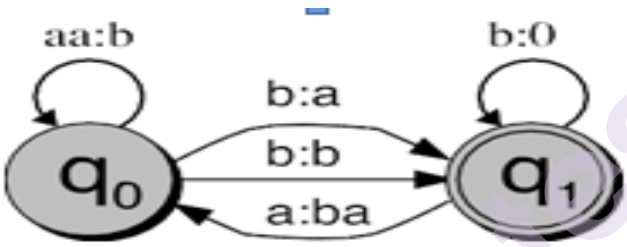


Figure 3.5.1 A Finite-state transducer

FSA defines a formal language by defining a set of strings, an FST defines a relation between sets of strings. In short, FST is as a machine that reads one string and generates another.

Four-fold way of thinking about transducer:

i. **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, and reject if it is not.

ii. **FST as generator:** a machine that outputs pairs of strings of the language. Means the output is a yes or no, and a pair of output strings.

iii. **FST as translator:** a machine that reads a string and outputs another string

iv. **FST as set relater:** a machine that computes relations between sets.

For morphological parsing (and for many other NLP applications), we can apply the FST as translator metaphor, taking as input a string of letters and producing as output a string of morphemes.

Formal definition an FST can be defined with 7 parameters:

Q: a finite set of N states q0, q1, . . . , qN−1

Σ: a finite set corresponding to the input alphabet

Δ: a finite set corresponding to the output alphabet

$q_0 \in Q$: start state

$F \subseteq Q$: set of final states

δ(q, w): the transition function between states; Given a state $q \in Q$ and a string $w \in \Sigma^*$,

δ(q, w) returns a set of new states $Q' \in Q$. δ is thus a function from $Q \times \Sigma*$ to 2^Q (because there are 2^Q possible subsets of Q). δ returns a set of states rather than a single state because a given input may be ambiguous in which state it maps to.

σ(q, w): the output function giving the set of possible output strings for each state and

input. Given a state $q \in Q$ and a string $w \in \Sigma^*$, σ(q, w) gives a set of output

strings, each a string $o \in \Delta*$. σ is thus a function from $Q \times \Sigma*$ to $2^{\Delta}*$

Where FSAs are isomorphic to regular languages, FSTs are isomorphic to regular relations. Regular relations are sets of pairs of strings, a natural extension of the regular languages, which are sets of strings. Like FSAs and regular languages, FSTs and regular relations are closed under union, although in general they are not closed under difference, complementation and.

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 alphabet O, T^{-1} maps from O to I.

  Inversion is useful because it makes it easy to convert a FST-as-parser into an FST- as-generator.

  T = {(a, a1), (a, a2), (b, b1), (c, c1), (c, a)}

  T^{-1} = {(a1, a), (a2, a), (b1, b), (c1, c), (a, c)

- **Composition**: If T1 is a transducer from I1 to O1 and T2 a transducer from O1 to O2, then T1 ◦ T2 maps from I1 to O2.

Composition is useful because it allows us to take two transducers that run in series and replace them with one more complex transducer. Composition works as in algebra; applying T1 ◦ T2 to an input sequence S is identical to applying T1 to S and then T2 to the result; thus T1 ◦ T2(S) = T2 (T1(S)).

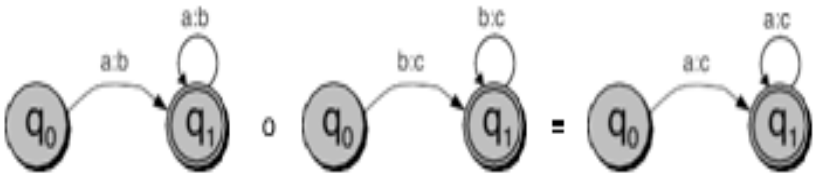


Fig. 3.5.2, for example, shows the composition of [a:b]+ with [b:c]+ to produce [a:c]+.

Figure 3.5.2 The composition of [a: b]+ with [b:c]+ to produce [a:c]+.

The projection of an FST is the FSA that is produced by extracting only one side of the relation. We can refer to the projection to the left or upper side of the relation as the upper or first projection and the projection to the lower or right side of the relation as the lower or second projection

3.6 MORPHOLOGICAL PARSING WITH FST

Now we see the task of morphological parsing. If we give the input cats, we'd like to output cat +N +Pl, telling us that cat is a plural noun

In the finite-state morphology paradigm, we represent a word as a correspondence between a lexical level (which represents a concatenation of morphemes making up a word) and the surface level (which represents the concatenation of letters which make up the actual spelling of the word). Fig. 3.6.1 shows these two levels for (English) cats.

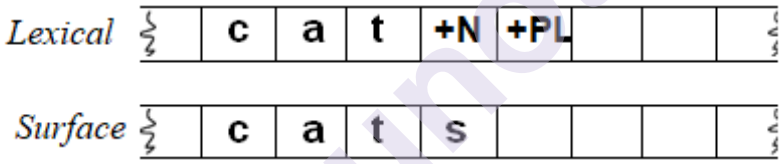


Figure 3.6.1 Schematic examples of the lexical and surface tapes

The actual transducers consist of intermediate tapes also. For finite-state morphology it's convenient to view an FST as having two tapes. The lexical tape is composed from characters from one alphabet Σ. The surface tape is composed of characters from another alphabet Δ. In the two-level morphology, we allow each arc only to have a single symbol from each alphabet. We can then combine the two symbol alphabets Σ and Δ to create a new alphabet, Σ′, which makes the relationship to FSAs quite clear. Σ′ is a finite alphabet of complex symbols. Each complex symbol is composed of an input-output pair i : o; one symbol i from the input alphabet Σ, and one symbol o from an output alphabet Δ, thus $\Sigma' \subseteq \Sigma \times \Delta$. Thus where an FSA accepts a language stated over a finite alphabet of single symbols,

such as the alphabet of our sheep language:
$\Sigma = \{b, a, !\}$

an FST defined this way accepts a language stated over pairs of symbols, as in:

$\Sigma' = \{a : a, b : b, ! : !, a : !, a : ϱ, ϱ : !\}$

In two-level morphology, the pairs of symbols in Σ' are also called feasible pairs. Thus each feasible pair symbol a : b in the transducer alphabet Σ' expresses how the symbol a from one tape is mapped to the symbol b on the other tape. For example a : ϱ means that an a on the lexical tape will correspond to nothing on the surface tape. Just as for an FSA, we can write regular expressions in the complex alphabet Σ'. Since it's most common for symbols to map to themselves, in two-level morphology we call pairs like a : a default pairs, and just refer to them by the single letter a.

We can now build an FST morphological parser out of our earlier morphotactic FSAs and lexica by adding an extra "lexical" tape and the appropriate morphological features. Fig. 3.6.2 shows an augmentation of Fig. 3.4.1 with the nominal morphological features (+Sg and +Pl) that correspond to each morpheme. The symbol ^ indicates a morpheme boundary, while the symbol # indicates a word boundary. The morphological features map to the empty string ϱ or the boundary symbols since there is no segment corresponding to them on the output tape.

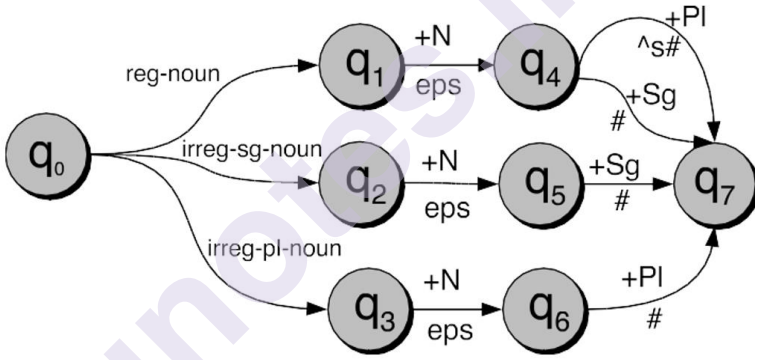


Figure 3.6.2 A schematic transducer for English nominal number inflection Tnum.

The symbols above each arc represent elements of the morphological parse in the lexical tape; the symbols below each arc represent the surface, using the morpheme-boundary symbol ^ and word-boundary marker #. The arcs need to be expanded by individual words in the lexicon.

In order to use Fig. 3.6.2 as a morphological noun parser, it needs to be expanded with all the individual regular and irregular noun stems, replacing the labels reg-noun etc. In order to do this we need to update the lexicon for this transducer, so that irregular plurals like geese will parse into the correct stem goose +N +Pl. We do this by allowing the lexicon to also have two levels. Since surface geese maps to lexical goose, the new lexical entry will be "g:g o:e o:e s:s e:e". Regular forms are simpler; the two-level entry for fox will now be "f:f o:o x:x", but by relying on the orthographic convention that f stands for f:f and so on, we can simply refer to it as fox and the form for geese as "g o:e o:e s e". Thus the lexicon will look only slightly more complex

reg-noun	irreg-pl-noun	irreg-sg-noun
fox cat dog	g o:e o:e se sheep m o:i u:ǫ s:c e	goose sheep mouse

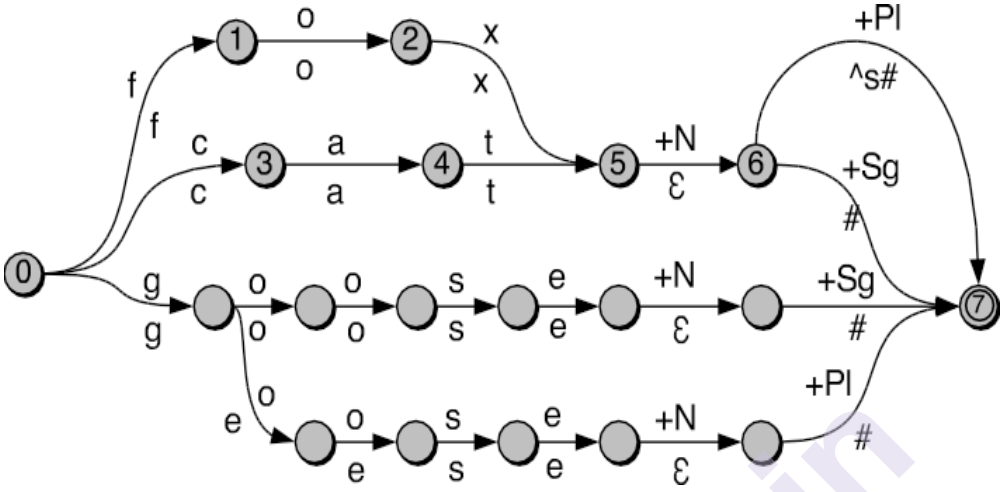


Figure 3.6.3 A fleshed-out English nominal inflection FST Tlex, expanded from Tnum by replacing the three arcs with individual word stems (only a few sample word stems are shown).

The resulting transducer, shown in Fig 3.6.3 will map plural nouns into the stem plus the morphological marker +Pl, and singular nouns into the stem plus the morphological marker +Sg. Thus surface cats will map to cat +N +Pl. This can be viewed in feasible-pair format as follows:

c:c a:a t:t +N:ǫ +Pl:^s#

Since the output symbols include the morpheme and word boundary markers ^ and #, the lower labels do not correspond exactly to the surface level. Hence we refer to tapes with these morpheme boundary markers in Fig. 3.6.4 as intermediate tapes; the next section will show how the boundary marker is removed.

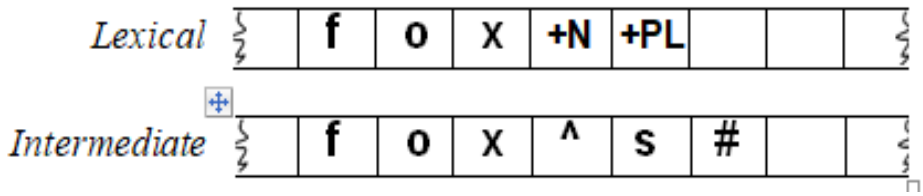


Figure 3.6.4 A schematic view of the lexical and intermediate tapes

3.7 LEXICON FREE FST PORTER STEMMER

While building a transducer from a lexicon plus rules is the standard algorithm for morphological parsing, there are simpler algorithms that don't require the large on-line lexicon demanded by this algorithm. These are used especially in Information Retrieval (IR) tasks like web search (a

user needs some information, and is looking for relevant documents) User gives the system a query with some important characteristics of documents she desires, and the IR system retrieves what it thinks are the relevant documents. Morphological information in IR is thus only used to determine that two words have the same stem; the suffixes are thrown away.

One of the most widely used such stemming algorithms are the simple and efficient Porter (1980) algorithm, which is based on a series of simple cascaded rewrite rules. Since cascaded rewrite rules are just the sort of thing that could be easily implemented as an FST, we think of the Porter algorithm as a lexicon-free FST stemmer. The algorithm contains rules like these:

ATIONAL → ATE (e.g., relational → relate)

ING → ǫ if stem contains vowel (e.g., motoring → motor)

Do stemmers really improve the performance of information retrieval, especially with smaller documents? Nonetheless, not all IR engines use stemming, partly because of stemmer errors. Following kinds of errors of omission and commission in the Porter algorithm:

Errors of Commission		**Errors of Omission**	
Organization-	organ	European-	Europe
doing-	doe	analysis-	analyzes
generalization-	generic	matrices-	matrix
numerical-	numerous	noise-	noisy
policy-	police	sparse-	sparsity
university-	universe	explain-	explanation

3.8 N-GRAMS

N-gram is simply a sequence of N words. For instance, a 2-gram (or bigram) is a two-word sequence of words like “please turn”, “turn your”, or ”your homework”, and a 3-gram (or trigram) is a three-word sequence of words like “please turn your”, or “turn your homework.”

In Natural Language Processing, n-grams are used for a variety of things. Some examples include like Auto completion of sentences, Auto spell check, Speech recognition and Machine translation.

How do N-grams work?

Let us take a look at the following examples.

- San Francisco (is a 2-gram)
- The Three Musketeers (is a 3-gram)
- She stood up slowly (is a 4-gram)

Now which of these three N-grams have you seen quite frequently? Probably, “San Francisco” and “The Three Musketeers”. On the other hand, you might not have seen “She stood up slowly” that frequently. Basically, “She stood up slowly” is an example of an N-gram that does not occur as often in sentences as Examples 1 and 2.

Now if we assign a probability to the occurrence of an N-gram or the probability of a word occurring next in a sequence of words, it can be very useful. Why?

First of all, it can help in deciding which N-grams can be chunked together to form single entities (like “San Francisco” chunked together as one word, “high school” being chunked as one word).

It can also help make next word predictions. Say you have the partial sentence “Please hand over your”. Then it is more likely that the next word is going to be “test” or “assignment” or “paper” than the next word being “school”.

It can also help to make spelling error corrections. For instance, the sentence “drink cofee” could be corrected to “drink coffee” if you knew that the word “coffee” had a high probability of occurrence after the word “drink” and also the overlap of letters between “cofee” and “coffee” is high.

3.9 N –GRAM LANGUAGE MODEL

An N-gram model is a type of Language Model (LM), that focuses on finding the probability distribution over word sequences. The model is built by counting how often word sequences occur in corpus text and then estimating the probabilities. Due to the fact that a simple N-gram model has limitations, improvements are often made through the means of smoothing, interpolation and backoff. A model that just focuses on how frequently a word occurs without looking at previous words is called unigram. If a model only considers just the previous word to predict the current word, then that model is called a bigram.

When there is a sequence of N-1 words, an N-gram model is used to predict the most probable word that would follow that sequence. The N-gram model is a probabilistic model that is trained on a corpus of text. Formal grammars (e.g. regular, context free) give a hard “binary” model of the legal sentences in a language. For NLP, a probabilistic model of a language that gives a probability that a string is a member of a language is more useful. To specify a correct probability distribution, the probability of all sentences in a language must sum to 1.

A language model also supports predicting the completion of a sentence.

- Please turn off your cell_____.
- Your program does not _____.

Predictive text input systems can guess what you are typing and give choices on how to complete it.

Here N-Gram Models ,Estimate probability of each word given prior context.

- P (phone | Please turn off your cell)

Number of parameters required grows exponentially with the number of words of prior context. An N-gram model uses only N-1 words of prior context.

- Unigram: P(phone)
- Bigram: P(phone | cell)
- Trigram: P(phone | your cell)

The Markov assumption is the presumption that the future behaviour of a dynamical system only depends on its recent history. In particular, in a kth-order Markov model, the next state only depends on the k most recent states, therefore an N-gram model is a (N-1)-order Markov model.

N-Gram Model Formulas

Word sequences

$$w_1^n = w_1 \ldots w_n$$

Chain rule of probability

$$P(w_1^n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2)...P(w_n \mid w_1^{n-1}) = \prod_{k=1}^{n} P(w_k \mid w_1^{k-1})$$

Bigram approximation

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k \mid w_{k-1}) \quad w_1^n = w_1 \ldots w_n$$

N-gram approximation

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k \mid w_{k-N+1}^{k-1})$$

Estimating Probabilities

N-gram conditional probabilities can be estimated from raw text based on the relative frequency of word sequences.

Bigram:

$$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

N-gram:

$$P(w_n \mid w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)}{C(w_{n-N+1}^{n-1})}$$

- To have a consistent probabilistic model, append a unique start (<s>) and end (</s>) symbol to every sentence and treat these as additional words.
- An N-gram model can be seen as a probabilistic automata for generating sentences.

Initialize sentence with N-1 <s> symbols
Until </s> is generated do:
Stochastically pick the next word based on the conditional probability of each word given the previous N -1 words.

Example:

Let's work through an example using a mini-corpus of three sentences

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

In general, we can say that this probability is (the number of times the previous word 'wp' occurs before the word 'wn') / (the total number of times the previous word 'wp' occurs in the corpus) =

(Count (wp wn))/(Count (wp))

To find the probability of the word "I" following the word "<s>",we can write this as
P (I | <s>) which is a conditional probability.
This becomes equal to:
= (No. of times "I" occurs) / (No. of times "<s>" occurs)
= 2/3
=0.67
Here are the calculations for some of the bigram probabilities from this corpus
P (Sam | <s>) = 1/ 3 = 0.33
P (am | I) =2/3=0.67
P (</s> | Sam)=1/2 = 0.5
P (Sam | am =1/2 = 0.5
P (do | I) =1/3 = 0.33

3.10 SUMMARY

In this chapter we studied concept of Regular expression (RE), Finite Automata, and Finite State Transducers (FST). Finite-state transducers are an extension of finite-state automata that can generate output symbols. Important operations for FSTs include composition, projection, and intersection. Finite-state morphology and two-level morphology are applications of finite- state transducers to morphological representation and parsing

After that we learned the Morphological Parsing with FST, Lexicon free FST Porter Stemmer. Finally we studied N –Grams and N-gram language model. An N-gram model is one type of a Language Model (LM), which is about finding the probability distribution over word sequences.

3.11 MULTIPLE CHOICE QUSETION ANSWERS

i. FST cannot work as _____

a) recognizer
b) generator
c) translator
d) lexicon

Ans : d

ii. N-Gram language models cannot be used for -------.

a) Spelling Correction
b) Predicting the completion of a sentence
c) Removing semantic ambiguity
d) Speech Recognition

Ans : c

iii. Both _____ and finite-state automata can be used to describe regular languages

a) Language model
b) Deterministic Finite Automata
c) Regular expressions
d) Finite State Translators (FSTs)

Ans : c

iv. _____ is as a machine that reads one string and generates another

a) FSA
b) FST
c) FSM
d) FSI

Ans : b

v. _____is used to remove the suffixes from an English word and obtain its stem which becomes very useful in the field of Information Retrieval (IR).

a) HMM Stemmer
b) Porter Stemmer
c) Markov Stemmer
d) Bert Stemmer

Ans : b

3.12 TRUE OR FALSE

i. Regular expressions are case sensitive--- **True**
ii. FSA define Regular Language while FST define Regular Relation--- **True**
iii. FST used in semantic analysis ----**False**

3.13 SAMPLE QUESTIONS

1. Explain the different forms of regular expressions used in NLP. Also state the uses of Regular expressions.
2. What is Lexicon free FST Porter stemmer?
3. Write a note on Morphological parsing with FST
4. What are bi-grams, tri-grams and N-grams? Give examples.
5. Explain Finite state Automata. What are deterministic and non-deterministic FSA?

3.14 LIST OF REFERENCES

1. "Natural Language Processing", Staredu Solutions.
2. Dan Jurafsky and James Martin. "Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition", Prentice Hall, Second Edition, 2009.
3. http://www.deepsky.com/~merovech/voynich/voynich_manchu_reference_materials/PDFs/jurafsky_martin.pdf
4. Christopher D.Manning and Hinrich Schutze, — Foundations of Statistical Natural Language Processing —, MIT Press, 1999.
5. https://www.analyticsvidhya.com/blog/2021/06/part-13-step-by-step-guide-to-master-nlp-regular-expressions/
6. https://www.cs.vassar.edu/~cs395/docs/3.pdf

❄❄❄❄❄❄❄

4

SYNTAX ANALYSIS - I

Unit Structure

4.0 Syntax analysis

- Check if the code is valid grammatically
- The syntactical analyser helps you to apply rules to the code
- Helps you to make sure that each opening brace has a corresponding closing balance
- Each declaration has a type and that the type must be exists.

4.1 Objective

This chapter would make you to give the idea of the goals of NLP.

Syntax Analysis is a second phase of the compiler design process in which the given input string is checked for the confirmation of rules and structure of the formal grammar. It analyses the syntactical structure and checks if the given input is in the correct syntax of the programming language or not.

- Syntax Analysis in Compiler Design process comes after the Lexical analysis phase. It is also known as the Parse Tree or Syntax Tree. The Parse Tree is developed with the help of pre-defined grammar of the language. The syntax analyser also checks whether a given program fulfills the rules implied by a context-free grammar. If it satisfies, the parser then creates the parse tree of that source program. Otherwise, it will display error messages.

- Why do you need Syntax Analyser?
- Important Syntax Analyser Terminology
- Why do we need Parsing?
- Parsing Techniques
- Error – Recovery Methods
- Grammar:
- Notational Conventions
- Context Free Grammar
- Grammar Derivation
- Syntax vs. Lexical Analyser
- Disadvantages of using Syntax Analysers

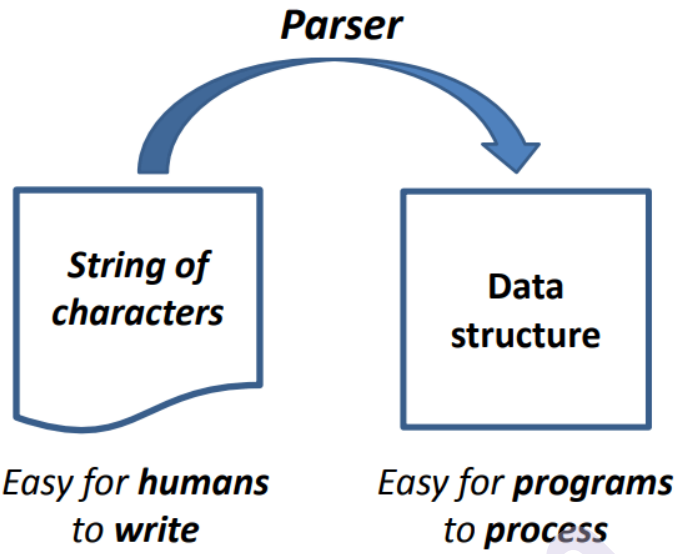
Why do you need Syntax Analyser?

- Check if the code is valid grammatically
- The syntactical analyser helps you to apply rules to the code
- Helps you to make sure that each opening brace has a corresponding closing balance
- Each declaration has a type and that the type must be exists

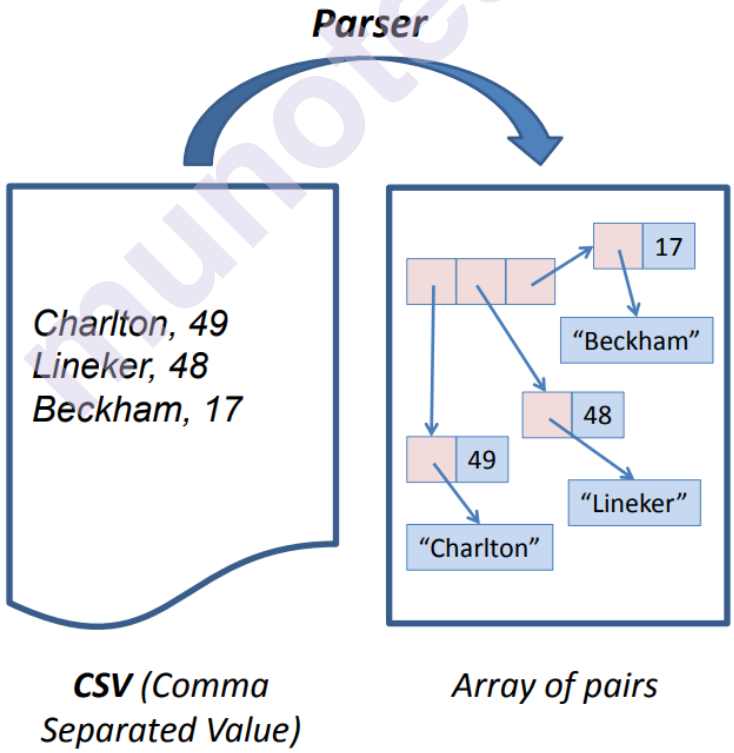
Important Syntax Analyser Terminology

Important terminologies used in syntax analysis process:

- **Sentence:** A sentence is a group of character over some alphabet.
- **Lexeme:** A lexeme is the lowest level syntactic unit of a language (e.g., total, start).
- **Token:** A token is just a category of lexemes.
- **Keywords and reserved words** – It is an identifier which is used as a fixed part of the syntax of a statement. It is a reserved word which you can't use as a variable name or identifier.
- **Noise words** – Noise words are optional which are inserted in a statement to enhance the readability of the sentence.
- **Comments** – It is a very important part of the documentation. It mostly display by, /* */, or//Blank (spaces)
- **Delimiters** – It is a syntactic element which marks the start or end of some syntactic unit. Like a statement or expression, "begin"…"end", or {}.
- **Character set** – ASCII, Unicode
- **Identifiers** – It is a restrictions on the length which helps you to reduce the readability of the sentence.
- **Operator symbols** – + and – performs two basic arithmetic operations.
- Syntactic elements of the Language



A parse also checks that the input string is well-formed, and if not, reject it.



Following are important tasks perform by the parser in compiler design:

- Helps you to detect all types of Syntax errors
- Find the position at which error has occurred
- Clear & accurate description of the error.

- Recovery from an error to continue and find further errors in the code.
- Should not affect compilation of "correct" programs.
- The parse must reject invalid texts by reporting syntax errors

Parsing Techniques

Parsing techniques are divided into two different groups:

- Top-Down Parsing,
- Bottom-Up Parsing

Top-Down Parsing:

In the top-down parsing construction of the parse tree starts at the root and then proceeds towards the leaves.

Two types of Top-down parsing are:

1. **Predictive Parsing:**

   Predictive parse can predict which production should be used to replace the specific input string. The predictive parser uses look-ahead point, which points towards next input symbols. Backtracking is not an issue with this parsing technique. It is known as LL(1) Parser

2. **Recursive Descent Parsing:**

   This parsing technique recursively parses the input to make a prase tree. It consists of several small functions, one for each nonterminal in the grammar.

Bottom-Up Parsing:

In the bottom up parsing in compiler design, the construction of the parse tree starts with the leave, and then it processes towards its root. It is also called as shift-reduce parsing. This type of parsing in compiler design is created with the help of using some software tools.

Error – Recovery Methods

Common Errors that occur in Parsing in System Software

- **Lexical**: Name of an incorrectly typed identifier
- **Syntactical**: unbalanced parenthesis or a missing semicolon
- **Semantical**: incompatible value assignment
- **Logical**: Infinite loop and not reachable code

A parser should able to detect and report any error found in the program. So, whenever an error occurred the parser. It should be able to handle it and carry on parsing the remaining input. A program can have following

types of errors at various compilation process stages. There are five common error-recovery methods which can be implemented in the parser

Statement mode recovery

- In the case when the parser encounters an error, it helps you to take corrective steps. This allows rest of inputs and states to parse ahead.
- For example, adding a missing semicolon is comes in statement mode recover method. However, parse designer need to be careful while making these changes as one wrong correction may lead to an infinite loop.

Panic-Mode recovery

- In the case when the parser encounters an error, this mode ignores the rest of the statement and not process input from erroneous input to delimiter, like a semi-colon. This is a simple error recovery method.
- In this type of recovery method, the parser rejects input symbols one by one until a single designated group of synchronizing tokens is found. The synchronizing tokens generally using delimiters like or.

Phrase-Level Recovery:

- Compiler corrects the program by inserting or deleting tokens. This allows it to proceed to parse from where it was. It performs correction on the remaining input. It can replace a prefix of the remaining input with some string this helps the parser to continue the process.

Error Productions

- Error production recovery expands the grammar for the language which generates the erroneous constructs. The parser then performs error diagnostic about that construct.

Global Correction:

- The compiler should make less number of changes as possible while processing an incorrect input string. Given incorrect input string a and grammar c, algorithms will search for a parse tree for a related string b. Like some insertions, deletions, and modification made of tokens needed to transform an into b is as little as possible.

Grammar:

A grammar is a set of structural rules which describe a language. Grammars assign structure to any sentence. This term also refers to the study of these rules, and this file includes morphology, phonology, and syntax. It is capable of describing many, of the syntax of programming languages.

Rules of Form Grammar

- The non-terminal symbol should appear to the left of the at least one production
- The goal symbol should never be displayed to the right of the::= of any production
- A rule is recursive if LHS appears in its RHS

Notational Conventions

Notational conventions symbol may be indicated by enclosing the element in square brackets. It is an arbitrary sequence of instances of the element which can be indicated by enclosing the element in braces followed by an asterisk symbol, { … }*.

It is a choice of the alternative which may use the symbol within the single rule. It may be enclosed by parenthesis ([,]) when needed.

Two types of Notational conventions area Terminal and Non-terminals

1. Terminals:

- Lower-case letters in the alphabet such as a, b, c,
- Operator symbols such as +,-, *, etc.
- Punctuation symbols such as parentheses, hash, comma
- 0, 1, …, 9 digits
- Boldface strings like id or if, anything which represents a single terminal symbol

2.Nonterminals:

- Upper-case letters such as A, B, C
- Lower-case italic names: the expression or some

Context Free Grammar

A CFG is a left-recursive grammar that has at least one production of the type. The rules in a context-free grammar are mainly recursive. A syntax analyser checks that specific program satisfies all the rules of Context-free grammar or not. If it does meet, these rules syntax analysers may create a parse tree for that programme.

expression -> expression -+ term
expression -> expression – term
expression-> term
term -> term * factor
term -> expression/ factor
term -> factor factor
factor -> (expression)
factor -> id

Grammar Derivation

Grammar derivation is a sequence of grammar rule which transforms the start symbol into the string. A derivation proves that the string belongs to the grammar's language.

Left-most Derivation

When the sentential form of input is scanned and replaced in left to right sequence, it is known as left-most derivation. The sentential form which is derived by the left-most derivation is called the left-sentential form.

Right-most Derivation

Rightmost derivation scan and replace the input with production rules, from right to left, sequence. It's known as right-most derivation. The sentential form which is derived from the rightmost derivation is known as right-sentential form.

Syntax vs. Lexical Analyser

Syntax Analyser

The syntax analyser mainly deals with recursive constructs of the language.

The syntax analyser works on tokens in a source program to recognize meaningful structures in the programming language.

It receives inputs, in the form of tokens, from lexical analysers.

Lexical Analyser

The lexical analyser eases the task of the syntax analyser.

The lexical analyser recognizes the token in a source program.

It is responsible for the validity of a token supplied by the syntax analyser

Disadvantages of using Syntax Analysers

- It will never determine if a token is valid or not
- Not helps you to determine if an operation performed on a token type is valid or not
- You can't decide that token is declared & initialized before it is being used

Summary

- Syntax analysis is a second phase of the compiler design process that comes after lexical analysis
- The syntactical analyser helps you to apply rules to the code
- Sentence, Lexeme, Token, Keywords and reserved words, Noise words, Comments, Delimiters, Character set, Identifiers are some important terms used in the Syntax Analysis in Compiler construction

- Parse checks that the input string is well-formed, and if not, reject it
- Parsing techniques are divided into two different groups: Top-Down Parsing, Bottom-Up Parsing
- Lexical, Syntactical, Semantical, and logical are some common errors occurs during parsing method
- A grammar is a set of structural rules which describe a language
- Notational conventions symbol may be indicated by enclosing the element in square brackets
- A CFG is a left-recursive grammar that has at least one production of the type
- Grammar derivation is a sequence of grammar rule which transforms the start symbol into the string
- The syntax analyser mainly deals with recursive constructs of the language while the lexical analyser eases the task of the syntax analyser in DBMS
- The drawback of Syntax analyser method is that it will never determine if a token is valid or not

4.2 Introduction-The main problems of language processing will be discussed with examples from NLP applications. This chapter will also introduce some methodological distinctions and place the applications and methodology like what is NLP?

Natural language processing (NLP) can be defined as the computer-based modeling of human language. The term 'NLP' is sometimes used more narrowly and often excludes information retrieval and sometimes even machine translation.

Natural language processing (NLP) is the intersection of computer science, linguistics and **machine learning.** The field focuses on communication between computers and humans in natural language and NLP is all about making computers understand and generate human language. **Applications of NLP** techniques include voice assistants like Amazon's Alexa and Apple's Siri, but also things like machine translation and text-filtering.

NLP has heavily benefited from recent advances in machine learning, especially from deep learning techniques. The field is divided into the three parts:

- **Speech Recognition—**The translation of spoken language into text.
- **Natural Language Understanding—**The computer's ability to understand what we say.
- **Natural Language Generation—**The generation of natural language by a computer.

II. Why NLP is difficult

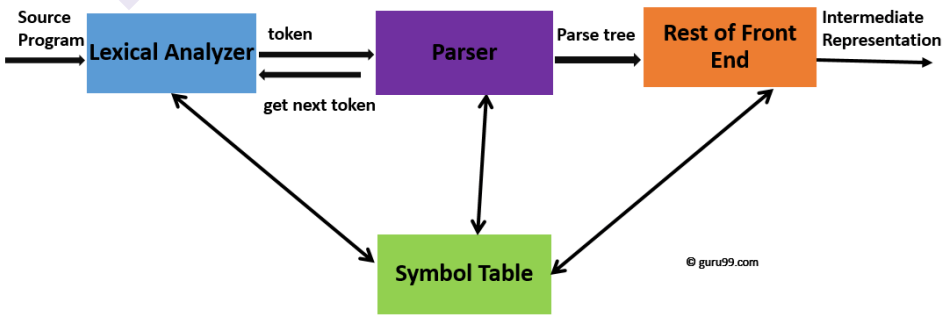
Human language is special for several reasons. It is specifically constructed to convey the speaker/writer's meaning. It is a complex system, although little children can learn it pretty quickly.

Another remarkable thing about human language is that it is all about symbols. According to Chris Manning, a machine learning professor at Stanford, it is a discrete, symbolic, categorical signaling system. This means we can convey the same meaning in different ways (i.e., speech, gesture, signs, etc.) The encoding by the human brain is a continuous pattern of activation by which the symbols are transmitted via continuous signals of sound and vision.

Understanding human language is considered a difficult task due to its complexity. For example, there is an infinite number of different ways to arrange words in a sentence. Also, words can have several meanings and contextual information is necessary to correctly interpret sentences. Every language is more or less unique and ambiguous. Just take a look at the following newspaper headline "The Pope's baby steps on gays." This sentence clearly has two very different interpretations, which is a pretty good example of the challenges in NLP.

Note that a perfect understanding of language by a computer would result in an AI that can process the whole information that is available on the internet, which in turn would probably result in artificial general intelligence.

NLP is sometimes compared to 'computational linguistics', although NLP is considered more applied. Nowadays, alternative terms are often preferred, such as 'Language Technology' or 'Language Engineering'. The term 'language' is often used in contrast to 'speech' (e.g. Speech and Language Technology). However, I will simply refer to NLP and use the term in a broader sense.



4.3 Define Syntax analysis

Syntax Analysis in Compiler Design process comes after the Lexical analysis phase. It is also known as the Parse Tree or Syntax Tree. The Parse Tree is developed with the help of pre-defined grammar of the language. The syntax analyser also checks whether a given program

fulfills the rules implied by a context-free grammar. If it satisfies, the parser then creates the parse tree of that source program. Otherwise, it will display error messages.

Tagging is a kind of classification that may be defined as the automatic assignment of description to the tokens. Here the descriptor is called tag, which may represent one of the part-of-speech, semantic information and so on.

What Is Natural Language Processing? A Gentle Introduction to NLP

An introduction to Natural Language Processing: what it is, how do we use it, what challenges we face and what tools we have.

Natural Language Processing, or **NLP** is a subfield of Artificial Intelligence research that is focused on developing models and points of interaction between humans and computers based on natural language. This includes text, but also speech-based systems.

Computer scientists and researchers have been studying this topic for entire decades, but only recently has it become a hot topic again, a situation made possible by recent breakthroughs in the research community.

Having this said, here's what you'll learn by the end of this article.

- A basic overview of how Natural Language Processing works
- Who uses Natural Language Processing and for what kind of problems
- What are the challenges of using Natural Language Processing for your app or for your business
- What are some basic tools that can get you started working with Natural Language Processing

How Natural Language Processing works

A generally accepted truth in computer science is that every complex problem becomes easier to solve if we break it into smaller pieces. That is especially true in the Artificial Intelligence field. For a given problem, we build several small, highly specialized components that are good at solving one and only one problem. We then align all this components, we pass our input through each component and we get our output at the end of the line. This is what we call a **pipeline**.

In the NLP context, a basic problem would be that for a given paragraph, the computer understands exactly the meaning of it and then possibly it acts accordingly. For this to work, we need to go through a few steps.

Sentence boundary segmentation

For a given text, we need to correctly identify every sentence, so that each sentence resulted from this will have its meaning extracted in the next steps. It seems that extracting the meaning from every sentence in a text

and then putting it all together is more accurate than trying to identify the meaning of the whole text. After all, when we speak (or write) we don't necessarily mean only one thing. We often tend to convey more ideas into one and the beauty of natural language (and the curse for NLP) is that we actually can.

A naive approach for this would be to only search for periods in a chunk of text and define that as the end of a sentence. The problem is that periods can also be used for other purposes(for example, abbreviations) so in practice machine learning models have been defined to correctly identify the punctuation marks that are used for ending sentences.

Word tokenization

This part involves taking a sentence from the previous step and breaking it into a list of all the words (and punctuation marks) it contains. This will be used in the next steps to perform an analysis on each word.

Part of Speech Tagging

This step involved taking each word from the previous step and classify it as to what part of speech it represents. This is an essential step for identifying the meaning behind a text. Identifying the nouns allows us to figure out who or what the given text is about. Then the verbs and adjectives let us understand what entities do, or how they are described, or any other meaning we can get from a text. PoS Tagging is a difficult problem but it has mostly been solved and implementations for this can be found in most of the modern Machine Learning libraries and tools.

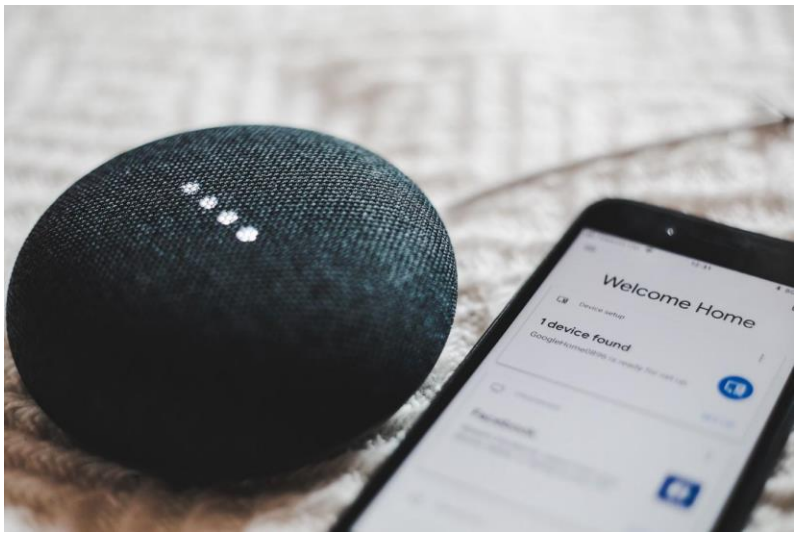
Named Entity Recognition — NER

This task refers to identifying the names in a sentence and correctly classify it against a list of predefined categories. Such categories may involve: Persons, Organizations, Locations, Time, Quantities and so on. Lists of categories may be tailored for your own particular use case, but in general, almost everybody needs at least these categories to be correctly identified.

There are many implementations for this kind of problem and models build recently have achieved near-human performance. Basically, this step is also divided in two subtasks: correctly identify the names in a sentence and then classify each name according to your list of categories.

There are of course many other tasks which NLP solves in order to be used in real world applications, but the next steps are tailored to every use case and every business need. Having all these said, the steps previously presented are the basic steps in almost every use case we can think of.

How Natural Language Processing is used



NLP is used in a variety of software and various use cases have been identified as being solvable by deploying NLP models. Some of these examples are:

- **Virtual assistants:** Most modern, powerful personal assistants employ a large suite of NLP techniques in order to help users accomplish their tasks. Siri, the Google Assistant and many others have become very efficient and highly skilled in helping their users by using the latest breakthroughs in the NLP field. This means that the companies behind them invest large amounts of resources in further research in the race to developing the perfect virtual assistant. While until a few years ago, these assistants were more fancy than useful, nowadays millions of users are using them and are able to take advantage of their solutions.

- **Machine translation:** Have you noticed that in the recent years, Google Translate has become more accurate in translating texts for you? This is thanks to latest advances achieved by the research in this field, with Google Translate powering machine translation in hundreds of languages for hundreds of millions of users.

- **Speech to Text:** In the fast-paced society that we live in today, we often don't have time to record in writing everything that we discuss, be it business notes, phone calls, speeches and so on. There are quite a handful of startups nowadays which help us with these kinds of problems, namely taking sound as an input and providing us with the text, from which we can carry on and take actions based on that.

Challenges of using Natural Language Processing

There are, of course, quite a few challenges when using NLP techniques. The most important are related to extracting context from the text. Humans are very good at understanding the context of a sentence, but

computers only use statistical models to represent these sentences, so it is very hard for them to understand exactly what we mean by our words.

For example, when we say “bush” we may refer to the plant or to the former US President. Identifying the difference may be very easy for you, but a computer will have to go through several steps from the pipeline until deciding which meaning you are using.

A further challenge of NLP is related to the first one and regards finding enough data to train our model. As with any other Machine Learning models, training an NLP models takes a lot of data and a lot of time, meaning there are only a handful of large enough companies that have the resources to build truly powerful applications involving NLP. Other, smaller companies are employing machine learning models that are highly specialized, meaning they solve only a subset of all the NLP problems, for which they need considerably less data.

Having considered all of these, it is important at the end of the day that we correctly identify the value that an NLP model brings to our business or our app. We need to see if the model we’ve managed to build can truly help us and our customers, or if it’s just a fancy feature so that we can say we are a machine learning company.

Natural Language Processing tools

There are quite a few libraries available for developers in order to start learning about and developing NLP models. And the good thing is most of them are open-source.

NLTK — Natural Language Toolkit is a Python library which is mainly used in research and education. It includes many steps in the pipeline with models which are ready to test and use.

Stanford CoreNLP is another Python library which provides a wide range of tools for understanding natural language.

spaCy is yet another Python library which employs machine learning and deep learning to help you with a lot of powerful feature.

❄❄❄❄❄❄❄

5

SYNTAX ANALYSIS - II

Unit Structure

5.0 Part-Of-Speech Tagging

Tagging is a kind of classification that may be defined as the automatic assignment of description to the tokens. Here the descriptor is called tag, which may represent one of the part-of-speech, semantic information and so on.

Now, if we talk about Part-of-Speech (PoS) tagging, then it may be defined as the process of assigning one of the parts of speech to the given word. It is generally called POS tagging. In simple words, we can say that POS tagging is a task of labelling each word in a sentence with its appropriate part of speech. We already know that parts of speech include nouns, verb, adverbs, adjectives, pronouns, conjunction and their sub-categories.

Most of the POS tagging falls under Rule Base POS tagging, Stochastic POS tagging and Transformation based tagging.

5.1 What are the parts of Speech?

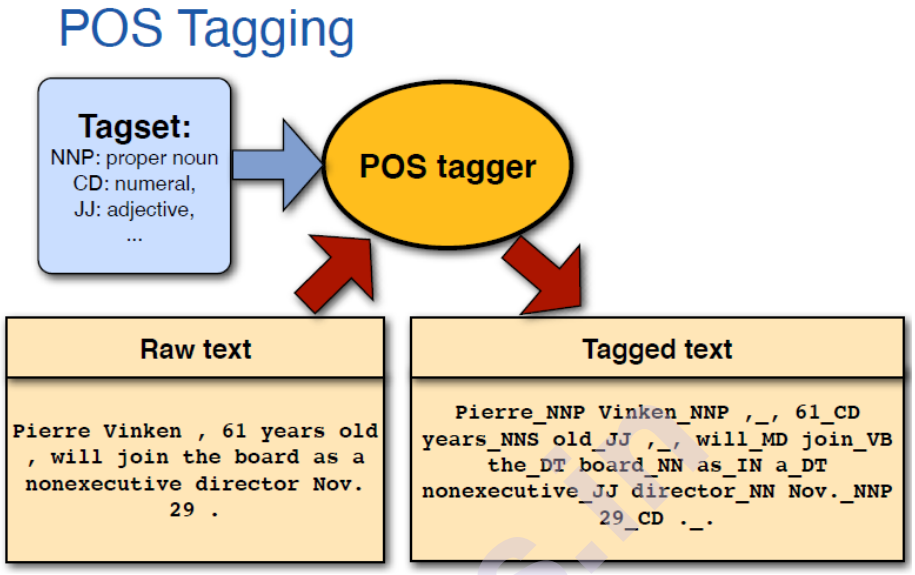
POS tagging words often have more than one POS: - The back door (adjective) - On my back (noun) - Winning back the voters (particle) - Promised to support the bill (verb) The POS tagging task: determine the POS tag for all tokens in a sentence. Due to ambiguity (and unknown words), we cannot rely on a dictionary to look up the correct POS tags.

Why POS tagging?

POS Tagging is one of the first steps in the NLP pipeline (right after tokenization, segmentation). POS tagging is traditionally considered a prerequisite for further analysis:

Syntactic parsing: what words are in the sentence?

Information extraction: finding names, dates, relationships, etc. NB: Although many neural models do not use POS tagging, it is still important to understand what makes POS tagging difficult (or easy) and how the basic models and algorithms work.



Creating a POS Tagger :To deal with ambiguity and coverage, POS taggers rely on learned models. For a new language (or domain) Step 0: Define a POS tag set Step 1: Annotate a corpus with these tags For a well-researched language (and domain):

Step 1: Obtain a POS -tagged corpus For any language....:

Step 2: Choose a POS tagging model (e.g., an HMM) Step 3: Train your model on your training corpus Step 4: Evaluate your model on your test corpus

Define a tag set We need to define an inventory of labels for the word classes (i.e., the tag set) - Most taggers are based on models that need to be trained on annotated (tagged) corpora.

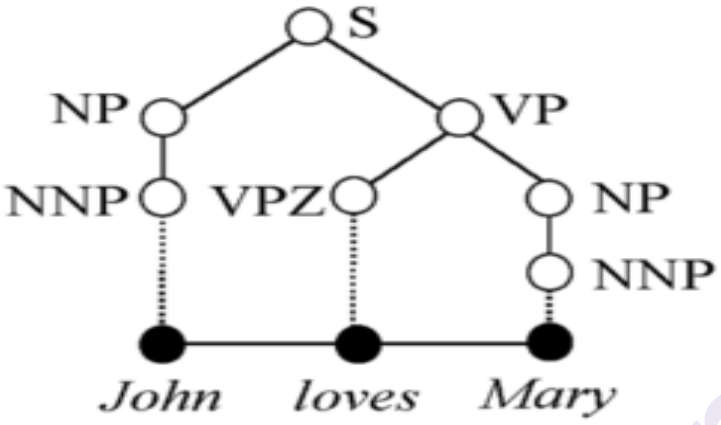
- Evaluation also requires annotated corpora. - Since annotation by humans is expensive and time-consuming, tag sets used in a few existing corpora are becoming the de facto standard. - Tag sets must capture semantically or syntactically important distinctions that can be easily made by trained human annotators.

5.2 Tag set for English (Penn Treebank)

In linguistics, a Treebank is a parsed text corpus that annotates syntactic or semantic sentence structure. The construction of parsed corpora in the early 1990s revolutionized computational linguistics, which benefitted from large-scale empirical data.

What is Treebank NLP?

A Treebank is **a collection of syntactically annotated sentences in which the annotation has been manually checked** so that the Treebank can serve as a training corpus for natural language parsers, as a repository for linguistic research, or as an evaluation corpus for NLP systems.



Tag	Description
ADJ	Adjective
ADV	Adposition
ADP	Adverb
AUX	Auxiliary
CCONJ	Coordinating Conjunction
DET	Determiner
INTJ	Interjection
NOUN	Noun
NUM	Numeral
PART	Particle
PRON	Pronoun
PROPN	Proper Noun
PUNCT	Punctuation
SCONJ	Subordinating Conjunction
SYM	Symbol
VERB	Verb
X	Other

5.3 Rule-based POS Tagging

One of the oldest techniques of tagging is rule-based POS tagging. Rule-based taggers use dictionary or lexicon for getting possible tags for tagging each word. If the word has more than one possible tag, then rule-based taggers use hand-written rules to identify the correct tag. Disambiguation can also be performed in rule-based tagging by analyzing the linguistic features of a word along with its preceding as well as following words. For example, suppose if the preceding word of a word is article then word must be a noun.

As the name suggests, all such kind of information in rule-based POS tagging is coded in the form of rules. These rules may be either −

- Context-pattern rules
- Or, as Regular expression compiled into finite-state automata, intersected with lexically ambiguous sentence representation.

We can also understand Rule-based POS tagging by its two-stage architecture −

- First stage − In the first stage, it uses a dictionary to assign each word a list of potential parts-of-speech.
- Second stage − In the second stage, it uses large lists of hand-written disambiguation rules to sort down the list to a single part-of-speech for each word.

Properties of Rule-Based POS Tagging

Rule-based POS taggers possess the following properties −

- These taggers are knowledge-driven taggers.
- The rules in Rule-based POS tagging are built manually.
- The information is coded in the form of rules.
- We have some limited number of rules approximately around 1000.
- Smoothing and language modeling is defined explicitly in rule-based taggers.

5.3.1 Stochastic POS Tagging

Another technique of tagging is Stochastic POS Tagging. Now, the question that arises here is which model can be stochastic. The model that includes frequency or probability (statistics) can be called stochastic. Any number of different approaches to the problem of part-of-speech tagging can be referred to as stochastic tagger.

The simplest stochastic tagger applies the following approaches for POS tagging −

Word Frequency Approach

In this approach, the stochastic taggers disambiguate the words based on the probability that a word occurs with a particular tag. We can also say that the tag encountered most frequently with the word in the training set is the one assigned to an ambiguous instance of that word. The main issue with this approach is that it may yield inadmissible sequence of tags.

Tag Sequence Probabilities

It is another approach of stochastic tagging, where the tagger calculates the probability of a given sequence of tags occurring. It is also called n-gram approach. It is called so because the best tag for a given word is determined by the probability at which it occurs with the n previous tags.

Properties of Stochastic POST Tagging

Stochastic POS taggers possess the following properties −

- This POS tagging is based on the probability of tag occurring.
- It requires training corpus
- There would be no probability for the words that do not exist in the corpus.
- It uses different testing corpus (other than training corpus).
- It is the simplest POS tagging because it chooses most frequent tags associated with a word in training corpus.

Transformation-based Tagging

Transformation based tagging is also called Brill tagging. It is an instance of the transformation-based learning (TBL), which is a rule-based algorithm for automatic tagging of POS to the given text. TBL, allows us to have linguistic knowledge in a readable form, transforms one state to another state by using transformation rules.

It draws the inspiration from both the previous explained taggers − rule-based and stochastic. If we see similarity between rule-based and transformation tagger, then like rule-based, it is also based on the rules that specify what tags need to be assigned to what words. On the other hand, if we see similarity between stochastic and transformation tagger then like stochastic, it is machine learning technique in which rules are automatically induced from data.

Working of Transformation Based Learning(TBL)

In order to understand the working and concept of transformation-based taggers, we need to understand the working of transformation-based learning. Consider the following steps to understand the working of TBL

- **Start with the solution** − The TBL usually starts with some solution to the problem and works in cycles.

- **Most beneficial transformation chosen** − In each cycle, TBL will choose the most beneficial transformation.
- **Apply to the problem** − The transformation chosen in the last step will be applied to the problem.

The algorithm will stop when the selected transformation in step 2 will not add either more value or there are no more transformations to be selected. Such kind of learning is best suited in classification tasks.

Advantages of Transformation-based Learning (TBL)

The advantages of TBL are as follows −

- We learn small set of simple rules and these rules are enough for tagging.
- Development as well as debugging is very easy in TBL because the learned rules are easy to understand.
- Complexity in tagging is reduced because in TBL there is interlacing of machinelearned and human-generated rules.
- Transformation-based tagger is much faster than Markov-model tagger.

Disadvantages of Transformation-based Learning (TBL)

The disadvantages of TBL are as follows −

- Transformation-based learning (TBL) does not provide tag probabilities.
- In TBL, the training time is very long especially on large corpora.

5.4 Issues-Multiple tags & words

The main problem with POS tagging is ambiguity. In English, many common words have multiple meanings and therefore multiple POS . The job of a POS tagger is to resolve this ambiguity accurately based on the context of use. For example, the word "shot" can be a noun or a verb

3.4.1. Introduction to context-free grammar (CFG)

Definition − A context-free grammar (CFG) consisting of a finite set of grammar rules is a quadruple (N, T, P, S) where

- N is a set of non-terminal symbols.
- T is a set of terminals where $N \cap T = NULL$.
- P is a set of rules, P: $N \rightarrow (N \cup T)^*$, i.e., the left-hand side of the production rule P does have any right context or left context.
- S is the start symbol.

Example

- The grammar ({A}, {a, b, c}, P, A), P : A → aA, A → abc.
- The grammar ({S, a, b}, {a, b}, P, S), P: S → aSa, S → bSb, S → ε
- The grammar ({S, F}, {0, 1}, P, S), P: S → 00S | 11F, F → 00F | ε

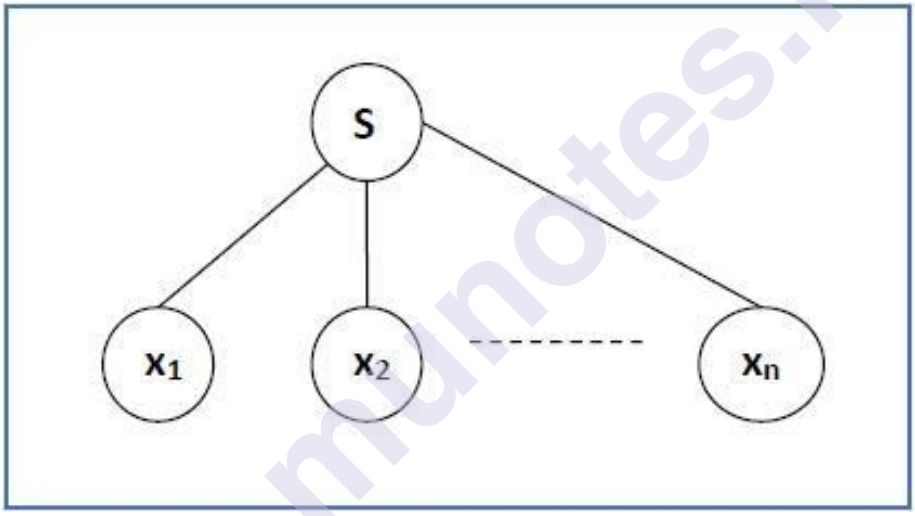
Generation of Derivation Tree

A derivation tree or parse tree is an ordered rooted tree that graphically represents the semantic information a string derived from a context-free grammar.

Representation Technique

- Root vertex − Must be labeled by the start symbol.
- Vertex − Labeled by a non-terminal symbol.
- Leaves − Labeled by a terminal symbol or ε.

If S → x_1x_2 …… x_n is a production rule in a CFG, then the parse tree / derivation tree will be as follows –



There are two different approaches to draw a derivation tree −

Top-down Approach −

- Starts with the starting symbol S
- Goes down to tree leaves using productions

Bottom-up Approach −

- Starts from tree leaves
- Proceeds upward to the root which is the starting symbol S

5.5 Sequence labeling

We have seen that language models estimate the probability of a sequence. Unlike text classification, we are not predicting any labels so it's an

example of unsupervised learning. This week we will look at supervised learning with sequences. In many NLP tasks, we want to produce a sequence *conditioned* on another sequence. Sequence labeling is perhaps the simplest form of a sequence-to-sequence task. Here our goal is to predict a label to each token in the input sequence.

Let's use part-of-speech tagging as a running example. The input a sequence of tokens (in a sentence), and we want to tag each token by its grammatical category such as nouns and verbs. For example,

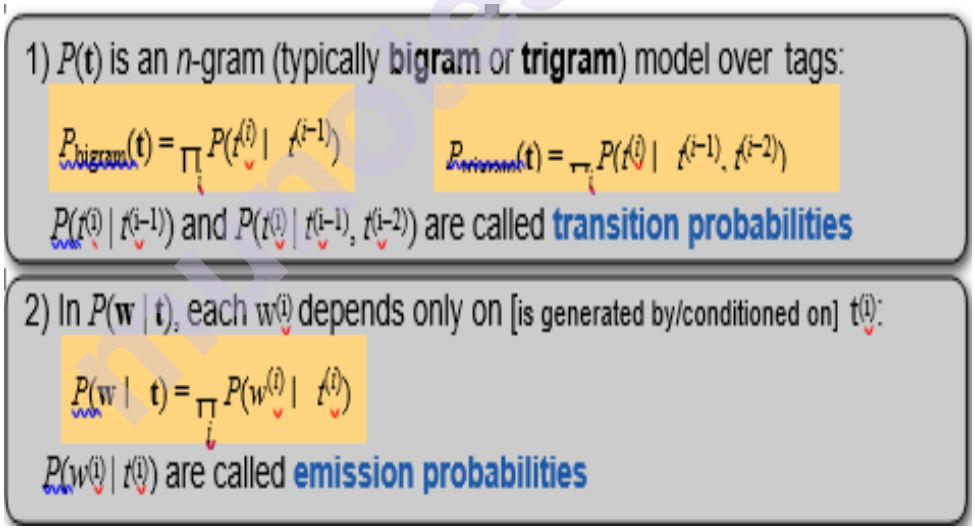
Language/NOUN is/VERB fun/ADJ ./PUNCT

A first thought might be to simply build a word-to-tag dictionary. However, many words can be assigned multiple tags depending on the context. In the above example, "fun" can be either a noun (as in "have fun") or an adjective (as in "a fun evening"). Therefore, the model must take context into consideration.

3.5.1 Hidden Markov Model (HMM)

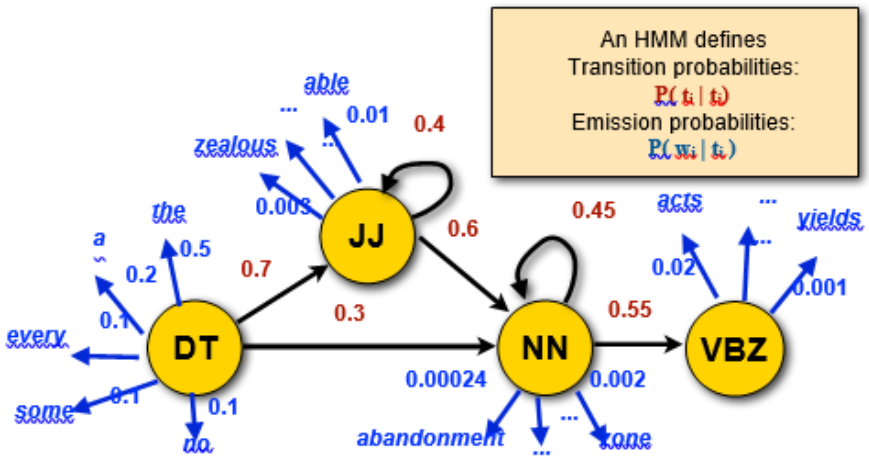
HMMs are the most commonly used generative models for POS tagging (and other tasks, e.g. in speech recognition)

HMMs make specific independence assumptions in $P(t)$ and $P(w|t)$:



These probabilities don't depend on the position in the sentence $^{(i)}$, but are defined over word and tag types.

With subscripts $_{i,j,k}$, to index word/tag types, they become $P(t_i \mid t_j)$, $P(t_i \mid t_j, t_k)$, $P(w_i \mid t_j)$



Hidden Markov Model (HMM) POS Tagging

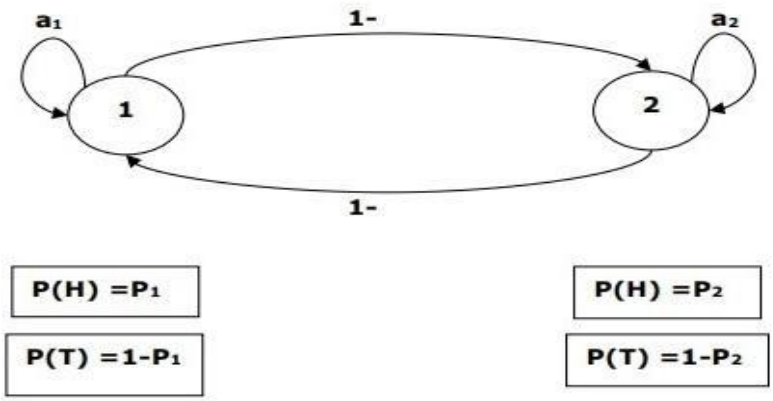
Before digging deep into HMM POS tagging, we must understand the concept of Hidden Markov Model (HMM).

Hidden Markov Model

An HMM model may be defined as the doubly-embedded stochastic model, where the underlying stochastic process is hidden. This hidden stochastic process can only be observed through another set of stochastic processes that produces the sequence of observations.

Example

For example, a sequence of hidden coin tossing experiments is done and we see only the observation sequence consisting of heads and tails. The actual details of the process - how many coins used, the order in which they are selected - are hidden from us. By observing this sequence of heads and tails, we can build several HMMs to explain the sequence. Following is one form of Hidden Markov Model for this problem −



We assumed that there are two states in the HMM and each of the state corresponds to the selection of different biased coin. Following matrix gives the state transition probabilities −

$$A=[a11a21a12a22]A=[a11a12a21a22]$$

Here,

- $\mathbf{a_{ij}}$ = probability of transition from one state to another from i to j.
- $\mathbf{a_{11}} + \mathbf{a_{12}} = 1$ and $a_{21} + a_{22} = 1$
- $\mathbf{P_1}$ = probability of heads of the first coin i.e. the bias of the first coin.
- $\mathbf{P_2}$ = probability of heads of the second coin i.e. the bias of the second coin.

We can also create an HMM model assuming that there are 3 coins or more.

This way, we can characterize HMM by the following elements −

- N, the number of states in the model (in the above example N =2, only two states).
- M, the number of distinct observations that can appear with each state in the above example M = 2, i.e., H or T).
- A, the state transition probability distribution − the matrix A in the above example.
- P, the probability distribution of the observable symbols in each state (in our example P1 and P2).
- I, the initial state distribution.

Use of HMM for POS Tagging

The POS tagging process is the process of finding the sequence of tags which is most likely to have generated a given word sequence. We can model this POS process by using a Hidden Markov Model (HMM), where **tags** are the **hidden states** that produced the **observable output,** i.e., the **words**.

Mathematically, in POS tagging, we are always interested in finding a tag sequence (C) which maximizes −

P (C|W)

Where,

$C = C_1, C_2, C_3... C_T$

$W = W_1, W_2, W_3, W_T$

On the other side of coin, the fact is that we need a lot of statistical data to reasonably estimate such kind of sequences. However, to simplify the problem, we can apply some mathematical transformations along with some assumptions.

The use of HMM to do a POS tagging is a special case of Bayesian interference. Hence, we will start by restating the problem using Bayes' rule, which says that the above-mentioned conditional probability is equal to −

(PROB (C_1, CT) * PROB (W_1, WT | C_1, CT)) / PROB (W_1, WT)

We can eliminate the denominator in all these cases because we are interested in finding the sequence C which maximizes the above value. This will not affect our answer. Now, our problem reduces to finding the sequence C that maximizes −

PROB (C_1, CT) * PROB (W_1, WT | C_1, CT) (1)

Even after reducing the problem in the above expression, it would require large amount of data. We can make reasonable independence assumptions about the two probabilities in the above expression to overcome the problem.

First Assumption

The probability of a tag depends on the previous one (bigram model) or previous two (trigram model) or previous n tags (n-gram model) which, mathematically, can be explained as follows −

PROB (C_1,..., C_T) = $\Pi_{i=1..T}$ PROB ($C_i|C_{i-n+1}…C_{i-1}$) (n-gram model)

PROB (C_1,..., CT) = $\Pi_{i=1..T}$ PROB ($C_i|C_{i-1}$) (bigram model)

The beginning of a sentence can be accounted for by assuming an initial probability for each tag.

PROB ($C_1|C_0$) = PROB $_{initial}$ (C_1)

Second Assumption

The second probability in equation (1) above can be approximated by assuming that a word appears in a category independent of the words in the preceding or succeeding categories which can be explained mathematically as follows −

PROB (W_1,..., W_T | C_1,..., C_T) = $\Pi_{i=1..T}$ PROB ($W_i|C_i$)

Now, on the basis of the above two assumptions, our goal reduces to finding a sequence C which maximizes

$\Pi_{i=1...T}$ PROB($C_i|C_{i-1}$) * PROB($W_i|C_i$)

Now the question that arises here is has converting the problem to the above form really helped us. The answer is - yes, it has. If we have a large tagged corpus, then the two probabilities in the above formula can be calculated as −

PROB ($C_{i=VERB}|C_{i-1=NOUN}$) = (# of instances where Verb follows Noun) / (# of instances where Noun appears) (2)

PROB ($W_i|C_i$) = (# of instances where W_i appears in C_i) /(# of instances where C_i appears) (3)

An example HMM

Transition Matrix A

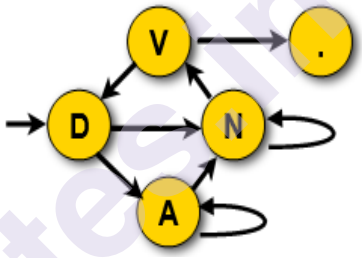
	D	N	V	A	.
D		0.8		0.2	
N		0.7	0.3		
V	0.6				0.4
A		0.8		0.2	
.					

Emission Matrix B

	the	man	ball	throw	sees	red	blue	.
D	1							
N		0.7	0.3					
V				0.6	0.4			
A						0.8	0.2	
.								1

Initial state vector π

	D	N	V	A	.
π	1				



Building an HMM tagger

To build an HMM tagger, we have to:

Train the model, i.e. estimate its parameters (the transition and emission probabilities)

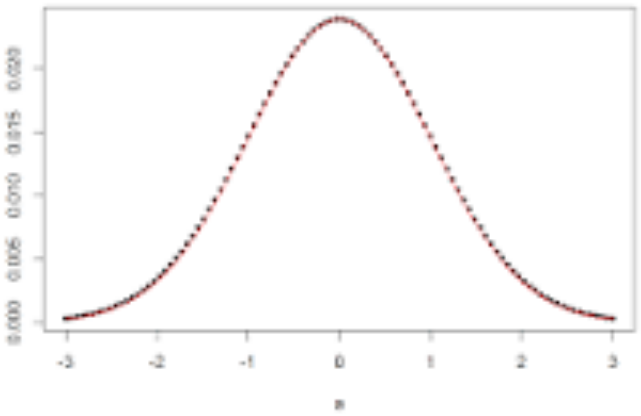
Easy case: We have a corpus labeled with POS tags (supervised learning)
Harder case: We have a corpus, but it's just raw text without tags (unsupervised learning). In that case it really helps to have a dictionary of which POS tags each word can have

Define and implement a **tagging algorithm** that finds the best tag sequence $\mathbf{t}^*$ for each input sentence $\mathbf{w}$:

$$\mathbf{t}^* = \text{argmax}_{\mathbf{t}} P(\mathbf{t})P(\mathbf{w} \mid \mathbf{t})$$

3.5.2 Maximum Entropy

What is Max entropy model?



The maximum entropy principle (MaxEnt) states that the most appropriate distribution to model a given set of data is the one with highest entropy among all those that satisfy the constrains of our prior knowledge. Usually, these constrains are given as equations regarding moments of the desired distribution.

Reference pdf :

- https://web.stanford.edu/group/cslipublications/cslipublications/pdf/1575863480.pdf
- https://www.researchgate.net/publication/270568943_Syntax_Workbook
- https://linguistics.ucla.edu/people/stabler/isat.pdf

❄❄❄❄❄❄❄

6

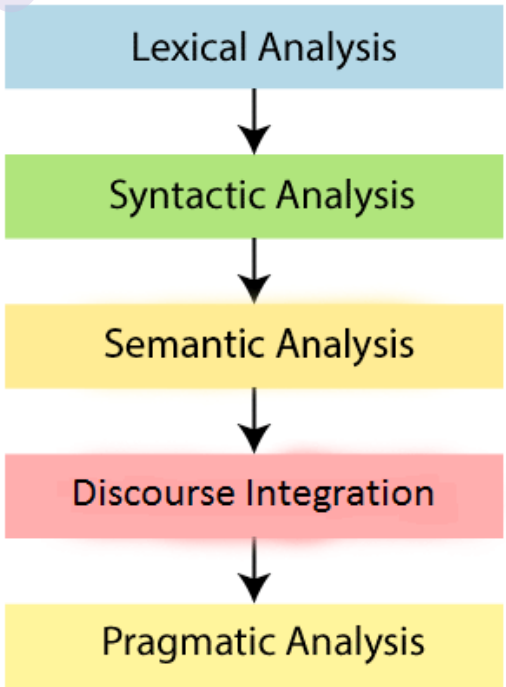
SEMANTIC ANALYSIS - I

Unit Structure

6.0 OBJECTIVE

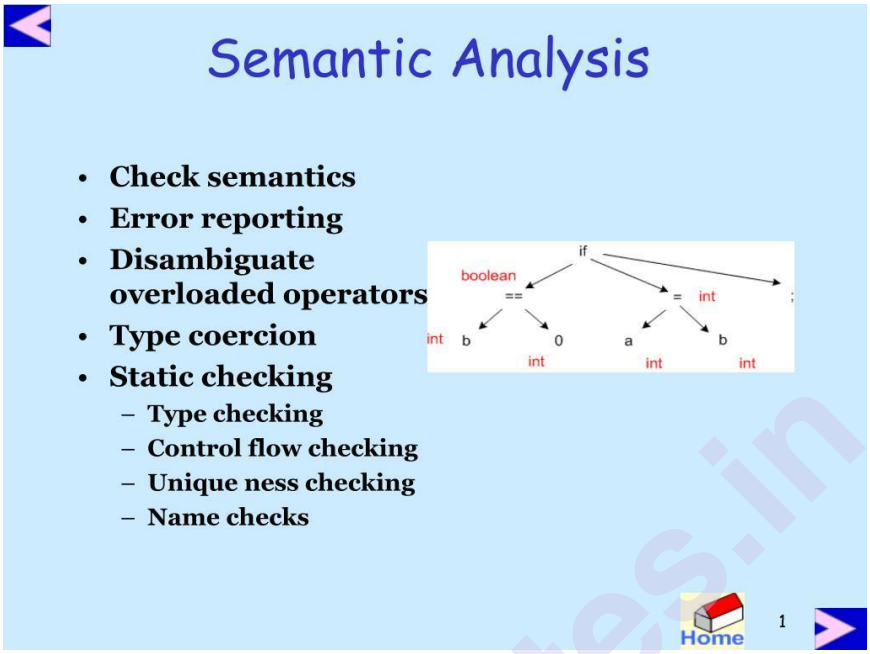
What is the role of semantic analysis in NLP?

The purpose of semantic analysis is **to draw exact meaning, or you can say dictionary meaning from the text**. The work of semantic analyzer is to check the text for meaningfulness.



What is semantic analysis used for?

Simply semantic analysis is **the process of drawing meaning from text**. It allows computers to understand and interpret sentences, paragraphs, or whole documents, by analyzing their grammatical structure, and identifying relationships between individual words in a particular context.



Semantic Analysis

Michael L. Scott, in Programming Language Pragmatics (Third Edition), 2009

One-Pass Compilers

A compiler that interleaves semantic analysis and code generation with parsing is said to be a *one-pass compiler*.[4] It is unclear whether interleaving semantic analysis with parsing makes a compiler simpler or more complex; it's mainly a matter of taste. If intermediate code generation is interleaved with parsing, one need not build a syntax tree at all (unless of course the syntax tree *is* the intermediate code). Moreover, it is often possible to write the intermediate code to an output file on the fly, rather than accumulating it in the attributes of the root of the parse tree. The resulting space savings were important for previous generations of computers, which had very small main memories. On the other hand, semantic analysis is easier to perform during a separate traversal of a syntax tree, because that tree reflects the program's semantic structure better than the parse tree does, especially with a top-down parser, and because one has the option of traversing the tree in an order other than that chosen by the parser.

Decorating a Syntax Tree

In our discussion so far we have used attribute grammars solely to decorate parse trees. As we mentioned in the chapter introduction,

attribute grammars can also be used to decorate syntax trees. If our compiler uses action routines simply to build a syntax tree, then the bulk of semantic analysis and intermediate code generation will use the syntax tree as base.

What is semantic analysis in natural language processing?

Semantic analysis describes **the process of understanding natural language**–the way that humans communicate–based on meaning and context. ... It analyzes context in the surrounding text and it analyzes the text structure to accurately disambiguate the proper meaning of words that have more than one definition.

What is semantic analysis language?

In linguistics, semantic analysis is **the process of relating syntactic structures**, from the levels of phrases, clauses, sentences and paragraphs to the level of the writing as a whole, to their language-independent meanings. ... Semantic analysis can begin with the relationship between individual words.

How important is semantics?

However, in truth, semantics is a very important subject. Semantics is **the study of the meaning of words**. Many words have very similar meanings and it is important to be able to distinguish subtle differences between them. ... ' It is also important for children to know the opposite meanings of words (antonyms).

What is meant by semantic features?

Semantic features are **theoretical units of meaning-holding components which are used for representing word meaning**. These features play a vital role in determining the kind of lexical relation which exists between words in a language.

What are examples of semantics?

semantics Add to list Share. Semantics is the study of meaning in language. It can be applied to entire texts or to single words. For example, **"destination" and "last stop"** technically mean the same thing, but students of semantics analyze their subtle shades of meaning.

We have learnt how a parser constructs parse trees in the syntax analysis phase. The plain parse-tree constructed in that phase is generally of no use for a compiler, as it does not carry any information of how to evaluate the tree. The productions of context-free grammar, which makes the rules of the language, do not accommodate how to interpret them.

For example

$E \rightarrow E + T$

The above CFG production has no semantic rule associated with it, and it cannot help in making any sense of the production.

Semantics

Semantics of a language provide meaning to its constructs, like tokens and syntax structure. Semantics help interpret symbols, their types, and their relations with each other. Semantic analysis judges whether the syntax structure constructed in the source program derives any meaning or not.

CFG + semantic rules = Syntax Directed Definitions

For example:

int a = “value”;

should not issue an error in lexical and syntax analysis phase, as it is lexically and structurally correct, but it should generate a semantic error as the type of the assignment differs. These rules are set by the grammar of the language and evaluated in semantic analysis. The following tasks should be performed in semantic analysis:

- Scope resolution
- Type checking
- Array-bound checking

Semantic Errors

We have mentioned some of the semantics errors that the semantic analyzer is expected to recognize:

- Type mismatch
- Undeclared variable
- Reserved identifier misuse.
- Multiple declaration of variable in a scope.
- Accessing an out of scope variable.
- Actual and formal parameter mismatch.

Attribute Grammar

Attribute grammar is a special form of context-free grammar where some additional information (attributes) are appended to one or more of its non-terminals in order to provide context-sensitive information. Each attribute has well-defined domain of values, such as integer, float, character, string, and expressions.

Attribute grammar is a medium to provide semantics to the context-free grammar and it can help specify the syntax and semantics of a programming language. Attribute grammar (when viewed as a parse-tree) can pass values or information among the nodes of a tree.

Example:

E → E + T { E.value = E.value + T.value }

The right part of the CFG contains the semantic rules that specify how the grammar should be interpreted. Here, the values of non-terminals E and T are added together and the result is copied to the non-terminal E.

Semantic attributes may be assigned to their values from their domain at the time of parsing and evaluated at the time of assignment or conditions. Based on the way the attributes get their values, they can be broadly divided into two categories : synthesized attributes and inherited attributes.

Synthesized attributes

These attributes get values from the attribute values of their child nodes. To illustrate, assume the following production:

S → ABC

If S is taking values from its child nodes (A,B,C), then it is said to be a synthesized attribute, as the values of ABC are synthesized to S.

As in our previous example (E → E + T), the parent node E gets its value from its child node. Synthesized attributes never take values from their parent nodes or any sibling nodes.

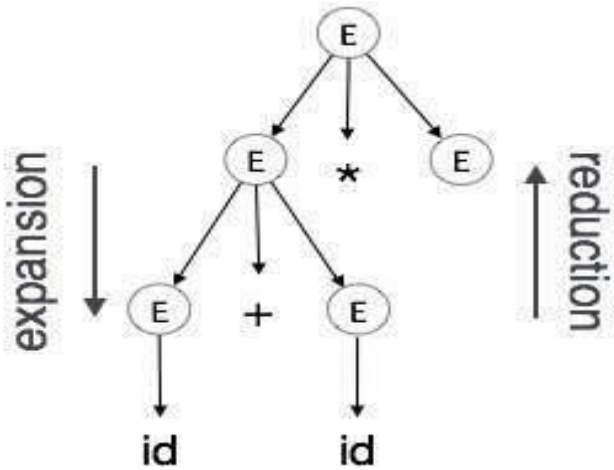
Inherited attributes

In contrast to synthesized attributes, inherited attributes can take values from parent and/or siblings. As in the following production,

S → ABC

A can get values from S, B and C. B can take values from S, A, and C. Likewise, C can take values from S, A, and B.

Expansion : When a non-terminal is expanded to terminals as per a grammatical rule



Reduction : When a terminal is reduced to its corresponding non-terminal according to grammar rules. Syntax trees are parsed top-down and left to right. Whenever reduction occurs, we apply its corresponding semantic rules (actions).

Semantic analysis uses Syntax Directed Translations to perform the above tasks.

Semantic analyzer receives AST (Abstract Syntax Tree) from its previous stage (syntax analysis).

Semantic analyzer attaches attribute information with AST, which are called Attributed AST.

Attributes are two tuple value, <attribute name, attribute value>

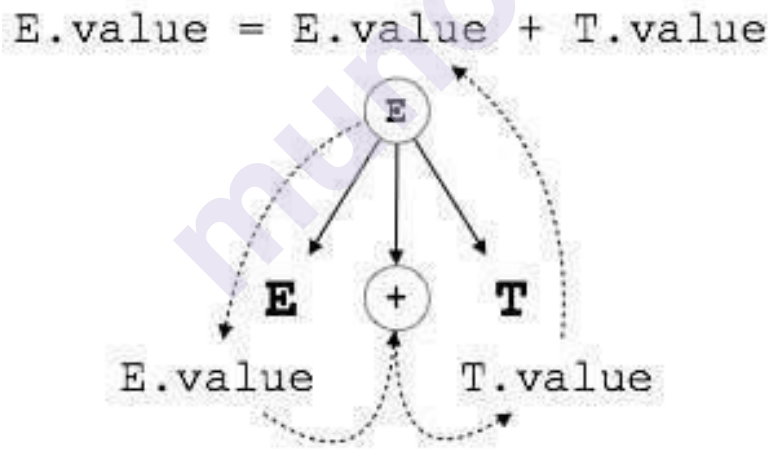
For example:

```
int value  = 5;
<type, “integer”>
<presentvalue, “5”>
```

For every production, we attach a semantic rule.

S-attributed SDT

If an SDT uses only synthesized attributes, it is called as S-attributed SDT. These attributes are evaluated using S-attributed SDTs that have their semantic actions written after the production (right hand side).



As depicted above, attributes in S-attributed SDTs are evaluated in bottom-up parsing, as the values of the parent nodes depend upon the values of the child nodes.

L-attributed SDT

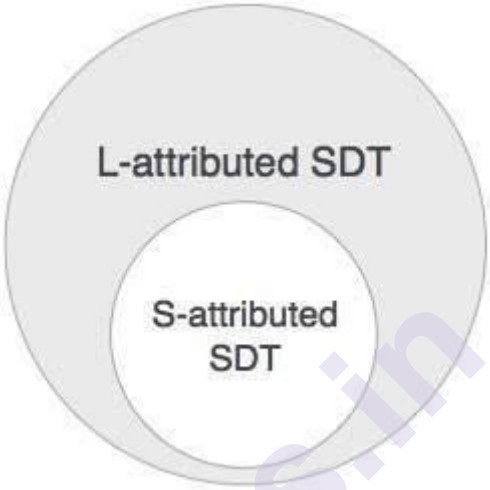
This form of SDT uses both synthesized and inherited attributes with restriction of not taking values from right siblings.

In L-attributed SDTs, a non-terminal can get values from its parent, child, and sibling nodes. As in the following production

S → ABC

S can take values from A, B, and C (synthesized). A can take values from S only. B can take values from S and A. C can get values from S, A, and B. No non-terminal can get values from the sibling to its right.

Attributes in L-attributed SDTs are evaluated by depth-first and left-to-right parsing manner.



We may conclude that if a definition is S-attributed, then it is also L-attributed as L-attributed definition encloses S-attributed definitions.

6.1 INTRODUCTION

The purpose of semantic analysis is to draw exact meaning, or you can say dictionary meaning from the text. The work of semantic analyzer is to check the text for meaningfulness.

We already know that lexical analysis also deals with the meaning of the words, then how is semantic analysis different from lexical analysis? Lexical analysis is based on smaller token but on the other side semantic analysis focuses on larger chunks. That is why semantic analysis can be divided into the following two parts −

Studying meaning of individual word

It is the first part of the semantic analysis in which the study of the meaning of individual words is performed. This part is called lexical semantics.

Studying the combination of individual words

In the second part, the individual words will be combined to provide meaning in sentences.

The most important task of semantic analysis is to get the proper meaning of the sentence. For example, analyze the sentence **"Ram is great."** In this sentence, the speaker is talking either about Lord Ram or about a person

whose name is Ram. That is why the job, to get the proper meaning of the sentence, of semantic analyzer is important.

Meaning Representation

Semantic analysis creates a representation of the meaning of a sentence. But before getting into the concept and approaches related to meaning representation, we need to understand the building blocks of semantic system.

Building Blocks of Semantic System

In word representation or representation of the meaning of the words, the following building blocks play an important role −

- **Entities** − It represents the individual such as a particular person, location etc. For example, Haryana. India, Ram all are entities.
- **Concepts** − It represents the general category of the individuals such as a person, city, etc.
- **Relations** − It represents the relationship between entities and concept. For example, Ram is a person.
- **Predicates** − It represents the verb structures. For example, semantic roles and case grammar are the examples of predicates.

Now, we can understand that meaning representation shows how to put together the building blocks of semantic systems. In other words, it shows how to put together entities, concepts, relation and predicates to describe a situation. It also enables the reasoning about the semantic world.

Approaches to Meaning Representations

Semantic analysis uses the following approaches for the representation of meaning −

- First order predicate logic (FOPL)
- Semantic Nets
- Frames
- Conceptual dependency (CD)
- Rule-based architecture
- Case Grammar
- Conceptual Graphs

Need of Meaning Representations

A question that arises here is why do we need meaning representation? Followings are the reasons for the same −

Linking of linguistic elements to non-linguistic elements

The very first reason is that with the help of meaning representation the linking of linguistic elements to the non-linguistic elements can be done.

Representing variety at lexical level

With the help of meaning representation, unambiguous, canonical forms can be represented at the lexical level.

Can be used for reasoning

Meaning representation can be used to reason for verifying what is true in the world as well as to infer the knowledge from the semantic representation.

6.1.2 Definition of Semantic Analysis

In linguistics, **semantic analysis** is the process of relating syntactic structures, from the levels of phrases, clauses, sentences and paragraphs to the level of the writing as a whole, to their language-independent meanings. It also involves removing features specific to particular linguistic and cultural contexts, to the extent that such a project is possible. The elements of idiom and figurative speech, being cultural, are often also converted into relatively invariant meanings in semantic analysis. Semantics, although related to pragmatics, is distinct in that the former deals with word or sentence choice in any given context, while pragmatics considers the unique or particular meaning derived from context or tone. To reiterate in different terms, semantics is about universally coded meaning, and pragmatics, the meaning encoded in words that is then interpreted by an audience.

Semantic analysis can begin with the relationship between individual words. This requires an understanding of lexical hierarchy, including hyponymy and hypernymy, meronomy, polysemy, synonyms, antonyms, and homonyms. It also relates to concepts like connotation (semiotics) and collocation, which is the particular combination of words that can be or frequently are surrounding a single word. This can include idioms, metaphor, and simile, like, "white as a ghost. In the current chapter we turn to the topic of semantics. Informally, syntax concerns the form of a valid program, while semantics concerns its meaning. Meaning is important for at least two reasons: it allows us to enforce rules (e.g., type consistency) that go beyond mere form, and it provides the information we need in order to generate an equivalent output program. It is conventional to say that the syntax of a language is precisely that portion of the language definition that can be described conveniently by a context-free grammar, while the semantics is that portion of the definition that cannot. This convention is useful in practice, though it does not always agree with intuition. When we require, for example, that the number of arguments contained in a call to a subroutine match the number of formal parameters in the subroutine definition, it is tempting to say that this requirement is a matter of syntax. After all, we can count arguments without knowing what they mean. Unfortunately, we cannot count them with context-free rules. Similarly, while it is possible to write a context-free grammar in which every function must contain at least one return statement, the required complexity makes this strategy very unattractive. In general, any rule that requires the compiler to compare things that are

separated by long distances, or to count things that are not properly nested, ends up being a matter of semantics. Semantic rules are further divided into static and dynamic semantics, though again the line between the two is somewhat fuzzy. The compiler enforces static semantic rules at compile time. It generates code to enforce dynamic semantic rules at run time (or to call library routines that do so). Certain errors, such as division by zero, or attempting to index into an array with an out-of-bounds subscript, cannot in general be caught at compile time, since they may occur only for certain input values, or certain behaviors of arbitrarily complex code. In special cases, a compiler may be able to tell that a certain error will always or never occur, regardless of run-time input. In these cases, the compiler can generate an error message at compile time, or refrain from generating code to perform the check at run time, as appropriate.

Semantic Analysis programs: there will inevitably be cases in which an error will always occur, but the compiler cannot tell, and must delay the error message until run time; there will also be cases in which an error can never occur, but the compiler cannot tell, and must incur the cost of unnecessary run-time checks. Both semantic analysis and intermediate code generation can be described in terms of annotation, or decoration of a parse tree or syntax tree. The annotations themselves are known as attributes. Numerous examples of static and dynamic semantic rules will appear in subsequent chapters.

In this current chapter we focus primarily on the mechanisms a compiler uses to enforce the static rules. We will consider intermediate code generation (including the generation of code for dynamic semantic checks) . we consider the role of the semantic analyzer in more detail, considering both the rules it needs to enforce and its relationship to other phases of compilation. Most of the rest of the chapter is then devoted to the subject of attribute grammars. Attribute grammars provide a formal framework for the decoration of a tree. This framework is a useful conceptual tool even in compilers that do not build a parse tree or syntax tree as an explicit data structure. We then consider various ways in which such grammars can be applied in practice. discusses the issue of attribute flow, which constrains the order(s) in which nodes of a tree can be decorated. In practice, most compilers require decoration of the parse tree (or the evaluation of attributes that would reside in a parse tree if there were one) to occur in the process of an LL or LR parse. presents action routines as an ad hoc mechanism for such “on-the-fly” evaluation. (mostly on the PLP CD) we consider the management of space for parse tree attributes. Because they have to reflect the structure of the CFG, parse trees tend to be very complicated. Once parsing is complete, we typically want to replace the parse tree with a syntax tree that reflects the input program in a more straightforward way. One particularly common compiler organization uses action routines during parsing solely for the purpose of constructing the syntax tree. The syntax tree is then decorated during a separate traversal, which can be formalized, if desired, with a separate attribute grammar.

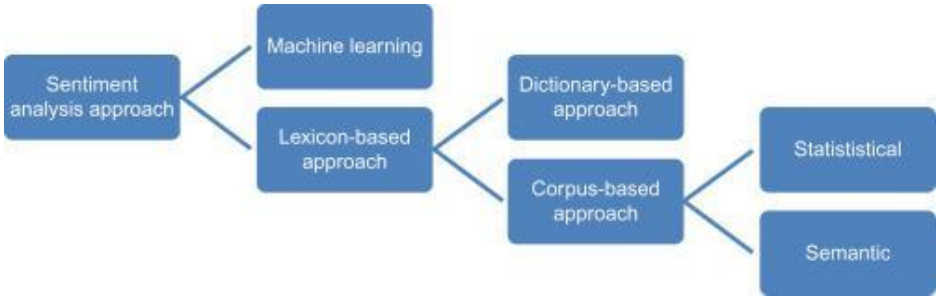
Space Management for Attributes

Any attribute evaluation method requires space to hold the attributes of the grammar symbols. If we are building an explicit parse tree, then the obvious approach is to store attributes in the nodes of the tree themselves. If we are not building a parse tree, then we need to find a way to keep track of the attributes for the symbols we have seen (or predicted) but not yet finished parsing. The details differ in bottom-up and top-down parsers. For a bottom-up parser with an S-attributed grammar, the obvious approach is to maintain an attribute stack that directly mirrors the parse stack: next to every state number on the parse stack is an attribute record for the symbol we shifted when we entered that state. Entries in the attribute stack are pushed and popped automatically by the parser driver; space management is not an issue for the writer of action routines. Complications arise if we try to achieve the effect of inherited attributes, but these can be accommodated within the basic attribute stack framework. For a top-down parser with an L-attributed grammar, we have two principal options. The first option is automatic, but more complex than for bottom-up grammars. It still uses an attribute stack, but one that does not mirror the parse stack. The second option has lower space overhead, and saves time by "shortcutting" copy rules, but requires action routines to allocate and deallocate space for attributes explicitly. In both families of parsers, it is common for some of the contextual information for action routines to be kept in global variables. The symbol table in particular is usually global. We can be sure that the table will always represent the current referencing environment, because we control the order in which action routines (including those that modify the environment at the beginnings and ends of scopes) are executed. In a pure attribute grammar, we should need to pass symbol table information into and out of productions through inherited and synthesized attributes.

6.2 LEXICAL SEMANTICS

One of the approaches or techniques of semantic analysis is the lexicon-based approach. This technique calculates the sentiment orientations of the whole document or set of sentence(s) from semantic orientation of lexicons. Semantic orientation can be positive, negative, or neutral. The dictionary of lexicons can be created manually as well as automatically generated. The WorldNet dictionary is used by many researchers. First of all, lexicons are found from the whole document and then WorldNet or any other kind of online thesaurus can be used to discover the synonyms and antonyms to expand that dictionary.

Lexicon-based techniques use adjectives and adverbs to discover the semantic orientation of the text. For calculating any text orientation, adjective and adverb combinations are extracted with their sentiment orientation value. These can then be converted to a single score for the whole value (Fig. below)



The different approaches to lexicon-based approach are:

1. Dictionary-based approach

In this approach, a dictionary is created by taking a few words initially. Then an online dictionary, thesaurus or WordNet can be used to expand that dictionary by incorporating synonyms and antonyms of those words. The dictionary is expanded till no new words can be added to that dictionary. The dictionary can be refined by manual inspection.

2. Corpus-based approach

This finds sentiment orientation of context-specific words. The two methods of this approach are:

Statistical approach: The words which show erratic behavior in positive behavior are considered to have positive polarity. If they show negative recurrence in negative text they have negative polarity. If the frequency is equal in both positive and negative text then the word has neutral polarity.

Semantic approach: This approach assigns sentiment values to words and the words which are semantically closer to those words; this can be done by finding synonyms and antonyms with respect to that word.

Lexical semantics (also known as **lexicosemantic**), as a subfield of linguistic semantics, is the study of word meanings. It includes the study of how words structure their meaning, how they act in grammar and compositionality, and the relationships between the distinct senses and uses of a word.

The units of analysis in lexical semantics are lexical units which include not only words but also sub-words or sub-units such as affixes and even compound words and phrases. Lexical units include the catalogue of words in a language, the lexicon. Lexical semantics looks at how the meaning of the lexical units correlates with the structure of the language or syntax. This is referred to as syntax-semantics interface.

The study of lexical semantics looks at:

- the classification and decomposition of lexical items
- the differences and similarities in lexical semantic structure cross-linguistically
- the relationship of lexical meaning to sentence meaning and syntax.

Lexical units, also referred to as syntactic atoms, can stand alone such as in the case of root words or parts of compound words or they necessarily attach to other units such as prefixes and suffixes do. The former are called free morphemes and the latter bound morphemes. They fall into a narrow range of meanings (semantic fields) and can combine with each other to generate new denotations

Lexical Semantics

The first part of semantic analysis, studying the meaning of individual words is called lexical semantics. It includes words, sub-words, affixes (sub-units), compound words and phrases also. All the words, sub-words, etc. are collectively called lexical items. In other words, we can say that lexical semantics is the relationship between lexical items, meaning of sentences and syntax of sentence.

Following are the steps involved in lexical semantics −

- Classification of lexical items like words, sub-words, affixes, etc. is performed in lexical semantics.
- Decomposition of lexical items like words, sub-words, affixes, etc. is performed in lexical semantics.
- Differences as well as similarities between various lexical semantic structures is also analyzed.

We understand that words have different meanings based on the context of its usage in the sentence. If we talk about human languages, then they are ambiguous too because many words can be interpreted in multiple ways depending upon the context of their occurrence.

Word sense disambiguation, in natural language processing (NLP), may be defined as the ability to determine which meaning of word is activated by the use of word in a particular context. Lexical ambiguity, syntactic or semantic, is one of the very first problem that any NLP system faces. Part-of-speech (POS) taggers with high level of accuracy can solve Word's syntactic ambiguity. On the other hand, the problem of resolving semantic ambiguity is called WSD (word sense disambiguation). Resolving semantic ambiguity is harder than resolving syntactic ambiguity.

For example, consider the two examples of the distinct sense that exist for the word ***"bass"*** −

- I can hear bass sound.
- He likes to eat grilled bass.

The occurrence of the word **bass** clearly denotes the distinct meaning. In first sentence, it means **frequency** and in second, it means **fish**. Hence, if it would be disambiguated by WSD then the correct meaning to the above sentences can be assigned as follows −

- I can hear bass/frequency sound.
- He likes to eat grilled bass/fish.

❄❄❄❄❄❄❄

7

SEMANTIC ANALYSIS - II

Unit structure

7.0 ATTACHMENT FOR FRAGMENT OF ENGLISH-SENTENCES

A sentence fragment is a group of words that cannot stand alone as a sentence because it does not express a complete thought. Learn about subordinators, fragment phrases, and fragment temptations to understand how to identify and correct sentence fragments.

Definition of a Sentence Fragment

Sentence fragments are groups of words that look like sentences, but aren't. To be a sentence, groups of words need to have at least one independent clause. An **independent clause** is any group of words that contain both a subject and a verb and can stand on its own. For example, 'I like cheeseburgers' is an independent clause.

Sentence fragments never have independent clauses, but instead are dependent clauses or phrases. Fragments can masquerade as real sentences because they begin with a capital letter and end with a period. If you read them more closely, you'll see that fragments don't form a complete thought. A sentence fragment is a little like having only half of the pieces to a puzzle. Without all the pieces, you won't have the whole picture.

Where to Find Sentence Fragments

Sentence fragments usually appear before or after the independent clauses to which they belong. For example:

When we got in the car. We rolled down the windows.

'When we got in the car' is a sentence fragment and a dependent clause. It clearly belongs to the independent clause that follows it and should be rewritten like this:

When we got in the car, we rolled down the windows.

Or like this:

We rolled down the windows when we got in the car.

Subordinators

The sentence fragment 'When we got in the car' also has the subordinator 'when'. Some other examples of **subordinators** are: 'after', 'although', 'before', 'if', 'since', 'until', 'when', 'where', 'while', and 'why'. Clauses with subordinators can be called either dependent clauses or subordinating clauses, but when those clauses appear at the beginning of a sentence, they should be followed by a comma.

Fragment Phrases

Phrases are groups of words that are missing a subject or verb, or both. Phrases can also masquerade as sentences, like dependent clauses can. Here are some examples.

Here's an example missing subject and verb:

From morning until night.

This fragment can be made a complete sentence by changing it to:

I worked from morning until night.

Adding 'I' as the subject and 'worked' as the verb corrects this fragment and makes it an independent clause and a complete thought.

Here's an example of a missing subject:

Start after the weekend.

This fragment can be made a complete sentence by changing it to:

Classes start after the weekend.

Adding the subject 'classes' corrects this fragment and makes it an independent clause and a complete thought.

Finally, here's an example of a missing verb:

Some girls in the class.

This fragment can be changed to:

Some girls in the class study together.

Adding the verb 'study' corrects this fragment and makes it an independent clause and a complete thought.

Fragment Temptations

Certain words and expressions make it easy to fall into the sentence fragment habit. Some of these words include 'also', 'for example', 'and', 'but', 'for instance', 'mainly', 'or', and 'that'. Here's how they appear in a sentence:

Harris claims that men and women have different ideas about dating. For example, that men should pay for dinner on a date.

7.1 NOUN PHRASES

Noun phrases are groups of words that function like nouns. Typically, they act as subjects, objects or prepositional objects in a sentence. While that might seem tricky to grasp, the best way to understand these useful phrases is to see them in action. Get a clear idea of a noun phrase and how it is in sentence through example.

What is a noun phrase example?

A noun phrase is either a pronoun or any group of words that can be replaced by a pronoun. For example, **'they', 'cars', and 'the cars'** are noun phrases, but 'car' is just a noun, as you can see in these sentences (in which the noun phrases are all in bold) Q: Do you like cars? A: Yes, I like them.

Noun phrases are simply nouns with modifiers. Just as nouns can act as subjects, objects and prepositional objects, so can noun phrases. Similarly, noun phrases can also work in a sentence as adjectives, participles, infinitives, and prepositional or absolute phrases. Noun phrases are important for adding more detail to a noun. Examples of simple noun phrases include:

- the little boy
- the happy puppy
- the building on the corner
- the sharp pencil
- your religion

7.2 VERB PHRASES

Verb phrase (VP): These phrases are **lexical units that have a verb acting as the head word**. Usually, there are two forms of verb phrases. One form has the verb components as well as other entities such as nouns, adjectives, or adverbs as parts of the object.

The phrase would include the verbal (participle, gerund or infinitive) and any modifiers, complements or objects. Examples of verb phrases versus verbal phrases include: **The man was texting on his phone**. (verb phrase was texting functions as the action) Texting on his phone, the man swerved into a ditch.

What is the main verb in a verb phrase?

The main verb is also called the lexical verb or the principal verb. This term refers to **the important verb in the sentence, the one that typically shows the action or state of being of the subject**. Main verbs can stand alone, or they can be used with a helping verb, also called an auxiliary verb.

What is a verb phrase in English language?

In English a verb phrase **combines with a noun or noun phrase acting as subject to form a simple sentence**. a phrase consisting of a main verb and any auxiliaries but not including modifiers, objects, or complements.

What are the types of verb phrases?

Verb phrases generally are divided among two types: **finite, of which the head of the phrase is a finite verb; and nonfinite, where the head is a nonfinite verb, such as an infinitive, participle or gerund**.

How do you find a verb phrase?

A verb phrase consists of a verb plus another word that further illustrates the verb tense, action, and tone. The other word or words tied to a verb in a verb phrase are its dependents, which can be adverbs, prepositional phrases, helping verbs, or other modifiers

Verb Phrase Examples

- She was walking quickly to the mall.
- He should wait before going swimming.
- Those girls are trying very hard.
- Ted might eat the cake.
- You must go right now.
- You can't eat that!
- My mother is fixing us some dinner.
- Words were spoken.

7.3 PREPOSITIONAL PHRASES

Prepositional phrases are groups of words containing **prepositions**. Remember that prepositions are words that indicate the relationships between various elements within a sentence, and you'll never have difficulty identifying prepositional phrases.

A prepositional phrase is a group of words that lacks either a **verb** or a subject, and that functions as a unified part of speech. It normally consists of a preposition and a **noun** or a preposition and a **pronoun**.

Remember the following rules for prepositional phrases and you will find that using them becomes much easier.

- Prepositional phrases always consist of two basic parts at minimum: the object and the preposition.
- In formal English, prepositions are almost always followed by objects.
- Adjectives can be placed between the prepositions and objects in prepositional phrases.

- Prepositional phrases can act as **adverbs** or **adjectives**. When they are used as adjectives, they modify nouns and pronouns in the same way single-word adjectives do.
- When prepositional phrases are used as adverbs, they at the same way single-word adverbs and adverb clauses do, modifying adjectives, verbs, and other adverbs.

Examples of Prepositional Phrases

The following sentences contain examples of prepositional phrases; the prepositional phrase in each sentence is italicized for easy identification.

The cupcake *with sprinkles* is yours.
The cupcake *with colorful sprinkles* is yours.
We climbed *up the hill*.
We climbed *up the very steep hill*.
The rabbits hopped *through the garden*.
The rabbits hopped *through the perfectly manicured garden.*

7.4. RELATIONS AMONG LEXEMES & THEIR SENSES

What is the relation among lexemes and their senses?

A lexeme is an individual entry in the lexicon. A lexicon is meaning structure holding meaning relations of lexemes. A lexeme may have different meanings. **A lexeme's meaning component is known as one of its senses.**

What are the sense relation?

sense relations as "**the relations of meaning between words, as expressed in synonymy, hyponymy, and antonymy**." Thus, sense relations can be seen from the similarity of meaning as in synonymy, the inclusion of meaning as in hyponymy, and the oppositeness of meaning as in antonymy.

What are the types of sense relations?

Two major types of sense relations can be distinguished:

- Sense relations of inclusion, esp. hyponymy and synonymy.
- Sense relations of exclusion, esp. complementarity and antonymy (both of which are instances of the relationship of incompatibility).

7.4.1 -Homonymy, Polysemy, Synonymy

The word Homonymy (from the Greek—homos: same, onoma: name) is **the relation between words with identical forms but different meanings**—that is, the condition of being homonyms. A stock example is the word bank as it appears in "river bank" and "savings bank."

What is the meaning of Homonymy?

Homonymy is **the relationship between words that are homonyms—words that have different meanings but are pronounced the same or spelled the same or both**. It can also refer to the state of being homonyms.

10 Homonyms with Meanings and Sentences

- Cache – Cash:
- Scents – Sense:
- Chile – Chili:
- Choir – Quire:
- Site – Sight:
- Facts- Fax:
- Finnish – Finish:

What is polysemy and examples?

polysemy Add to list Share. **When a symbol, word, or phrase means many different things**, that's called polysemy. The verb "get" is a good example of polysemy — it can mean "procure," "become," or "understand."

What are the polysemy words?

A polysemous word is **a word that has different meanings that derive from a common origin**; a homograph is a word that has different meanings with unrelated origins. Polysemous words and homographs constitute a known problem for language learners.

How many types of polysemy are there?

Types of polysemy

Linear polysemy accounts for a specialization-generalization relation between senses and, in turn, is divided into **four types**:

autohyponymy, automeronymy, autosuperordination and autoholonymy.

Synonymy is **a relation between individual senses of words, so that a single word typically has different sets of synonyms for each of its senses**. For example, coat has different synonyms for its senses 'outer garment' (e.g., jacket) and 'covering layer'(e.g., layer)

What is the concept of synonymy? Meaning of synonymy in English?

the state of being synonymous (= havings the same or almost the same meaning as another word or phrase in the same language) or the fact that words or phrases are synonymous: ... Children learn to accept two labels for the same object (synonymy).

What are the 5 examples of synonyms?

II. Examples of Synonyms

- Bad: awful, terrible, horrible.
- Good: fine, excellent, great.
- Hot: burning, fiery, boiling.
- Cold: chilly, freezing, frosty.
- Easy: Simple, effortless, straightforward.
- Hard: difficult, challenging, tough.
- Big: large, huge, giant.
- Small: tiny, little, mini.

7.5 ROBUST, WORD SENSES

What is the meaning of being robust?

Definition of robust

1a : **having or exhibiting strength or vigorous health**. b : having or showing vigor, strength, or firmness a robust debate a robust faith.

What is robust example?

The definition of robust is a strong and healthy person or animal, something rich and full of flavor or an activity that requires someone strong and healthy to complete. An example of robust is **a winning race horse**. An example of robust is strong coffee.

What is word sense in NLP?

In natural language processing, word sense disambiguation (WSD) is **the problem of determining which "sense" (meaning) of a word is activated by the use of the** word in a particular context, a process which appears to be largely unconscious in people.

What type of word is sense?

noun. **any of the faculties**, as sight, hearing, smell, taste, or touch, by which humans and animals perceive stimuli originating from outside or inside the body: My sense of smell tells me that dinner is ready.

Summary and Concluding Remarks

This chapter has discussed the task of semantic analysis. We reviewed the sorts of language rules that can be classified as syntax, static semantics, and dynamic semantics, and discussed the issue of whether to generate code to perform dynamic semantic checks. We also considered the role that the semantic analyzer plays in a typical compiler. We noted that both the enforcement of static semantic rules and the generation of intermediate code can be cast in terms of annotation, or decoration, of a parse tree or syntax tree.

We then presented attribute grammars as a formal framework for this decoration process. An attribute grammar associates attributes with each symbol in a context-free grammar or tree grammar, and attribute rules with each production.

Synthesized attributes are calculated only in productions in which their symbol appears on the left-hand side. The synthesized attributes of tokens are initialized by the scanner. Inherited attributes are calculated in productions in which their symbol appears within the right-hand side; they allow calculations internal to a symbol to depend on the context in which the symbol appears. Inherited attributes of the start symbol (goal) can represent the external environment of the compiler. Strictly speaking, attribute grammars allow only copy rules (assignments of one attribute to another) and simple calls to semantic functions, but we usually relax this restriction to allow more-or-less arbitrary code fragments in some existing programming language. Just as context-free grammars can be categorized according to the parsing algorithm(s) that can use them, attribute grammars can be categorized according to the complexity of their pattern of attribute flow. S-attributed grammars, in which all attributes are synthesized, can naturally be evaluated in a single bottom up pass over a parse tree, in precisely the order the tree is discovered by an LR family parser. L-attributed grammars, in which all attribute flow is depth-first left-to-right, can be evaluated in precisely the order that the parse tree is predicted and matched by an LL-family parser.

Attribute grammars with more complex patterns of attribute flow are not commonly used in production compilers, but for syntax-based editors, incremental compilers, and various other tools. While it is possible to construct automatic tools to analyze attribute flow and decorate parse trees, most compilers rely on action routines, which the compiler writer embeds in the right-hand sides of productions to evaluate attribute rules at specific points in a parse. In an LL-family parser, action routines can be embedded at arbitrary points in a production's right-hand side. In an LR-family parser, action routines must follow the production's left corner. Space for attributes in a bottom-up compiler is naturally allocated in parallel with the parse stack, but this complicates the management of inherited attributes. Space for attributes in a top-down compiler can be allocated automatically, or managed explicitly by the writer of action routines. The automatic approach has the advantage of regularity, and is easier to maintain; the ad hoc approach is slightly faster and more flexible. In a one-pass compiler, which interleaves scanning, parsing, semantic analysis, and code generation in a single traversal of its input, semantic functions or action routines are responsible for all of semantic analysis and code generation. More commonly, action routines simply build a syntax tree, which is then decorated during separate traversal(s) in subsequent pass(es). we will consider a wide variety of programming language constructs. Rather than present the actual attribute grammars required to implement these constructs, we will describe their semantics informally, and give examples of the target code. We will return to attribute grammars in Chapter 14, when we consider the generation of intermediate code in more detail.

7.5.1 Disambiguation (WSD)

What do you mean by word sense disambiguation WSD?

In natural language processing, word sense disambiguation (WSD) is **the problem of determining which "sense"** (meaning) of a word is activated by the use of the word in a particular context, a process which appears to be largely unconscious in people

What is word sense disambiguation write down the application of WSD?

Word Sense Disambiguation (WSD), has been a trending area of research in Natural Language Processing and Machine Learning. WSD is **basically solution to the ambiguity which arises due to different meaning of words in different** context. ... The ambiguity that arises due to this, is tough for a machine to detect and resolve

What is disambiguation example?

The definition of a disambiguation is **a removal of uncertainty or confusion**. An example of disambiguation is when a study explains the discrepancy between two different scientific studies that point to different results that create uncertainty. noun. 2

What are the different approaches to WSD?

WSD approaches are categorized mainly into three types,
Knowledge-based,
Supervised and
Unsupervised methods,

Word Sense Disambiguation is an important method of NLP by which the meaning of a word is determined, which is used in a particular context. NLP systems often face the challenge of properly identifying words, and determining the specific usage of a word in a particular sentence has many applications.

Word Sense Disambiguation basically solves the ambiguity that arises in determining the meaning of the same word used in different situations.

Word Sense Disambiguation Applications

WSD has many applications in various text processing and NLP fields.

- WSD can be used alongside Lexicography. Much of the modern Lexicography is corpus-based. WSD, used in Lexicography can provide significant textual indicators.
- WSD can also be used in Text Mining and Information Extraction tasks. As the major purpose of WSD is to accurately understand the meaning of a word in particular usage or sentence, it can be used for the correct labeling of words.

- For example, from a security point of view, a text system should be able to understand the difference between a **coal "mine"** and a **land "mine"**.
- While the former serves industrial purposes, the latter is a security threat. So a text mining application must be able to determine the difference between the two.
- Similarly, WSD can be used for Information Retrieval purposes. Information Retrieval systems work through text data primarily based on textual information. Knowing the relevance of using a word in any sentence will surely help.
- Challenges in Word Sense Disambiguation
- WSD faces a lot of challenges and problems.
- The most common problem is the difference between various dictionaries or text corpus. Different dictionaries have different meanings for words, which makes the sense of the words to be perceived as different. A lot of text information is out there and often it is not possible to process everything properly.
- Different applications need different algorithms and that is often a challenge for WSD.
- A problem also arises is that words cannot be divided into discrete meanings. Words often have related meanings and this causes a lot of problems.

How to implement WSD?

There are four main ways to implement WSD.

These are:

Dictionary- and knowledge-based methods:

These methods rely on text data like dictionaries, thesaurus, etc. It is based on the fact that words that are related to each other can be found in the definitions. The popularly used Leask method, which we shall discuss more later is a seminal dictionary-based method.

Supervised methods:

In this type, sense-annotated corpora are used to train machine learning models. But, a problem that may arise is that such corpora are very tough and time-consuming to create.

Semi-supervised Methods:

Due to the lack of such corpus, most word sense disambiguation algorithms use semi-supervised methods. The process starts with a small amount of data, which is often manually created.

This is used to train an initial classifier. This classifier is used on an untagged part of the corpus, to create a larger training set. Basically, this

method involves bootstrapping from the initial data, which is referred to as the seed data.

Semi-supervised methods thus, use both labeled and unlabeled data.

Unsupervised Methods:

Unsupervised Methods pose the greatest challenge to researchers and NLP professionals. A key assumption of these models is that similar meanings and senses occur in a similar context. They are not dependent on manual efforts, hence can overcome the knowledge acquisition deadlock.

Lesk Algorithm

Lesk Algorithm is a classical Word Sense Disambiguation algorithm introduced by Michael E. Lesk in 1986.

The Lesk algorithm is based on the idea that words in a given region of the text will have a similar meaning. In the Simplified Lesk Algorithm, the correct meaning of each word context is found by getting the sense which overlaps the most among the given context and its dictionary meaning.

Read More about the Lesk Algorithm here.
Let us start by importing the libraries.
So, the NLTK library can implement the Lesk method properly.
Do check out the code here.

Why is WSD any relevant?

Word Sense Disambiguation is closely related to Parts of speech tagging and is an important part of the whole Natural Language Processing process.

WSD if implemented properly, can lead to breakthroughs in NLP. A problem that often arises is the whole meaning of word sense. Word sense is not a numeric quantity that can be measured or a true or false value that can be denoted as 1 or 0.

The whole idea of word sense is controversial. The meaning of a word is highly contextual and depends on its usage. It is not something that can be easily measured as a discrete quantity.

Lexicography deals with generalizing the corpus and explaining the full and extended meaning of a word. But, sometimes these meanings might not apply to the algorithms or data.

My personal opinion and experience, say that working with text data can be tricky. So, implementation of WSD can be often very difficult and problem-creating.

But, WSD has immense applications and uses.

If a computer algorithm can just read a text and identify different uses of a text, it would mean vast improvements in the field of text analytics.

Comparing and evaluating various WSD methods can be difficult. But, with time more research is going on regarding WSD and it can be improved.

7.5.2 Dictionary based approach

Dictionary Based Algorithm. A simple approach to segment text is to scan each character one at a time from left to right and look up those characters in a dictionary. If the series of characters found in the dictionary, then we have a matched word and segment that sequence as a word.

Functional Modelling and Mathematical Models: A Semantic Analysis

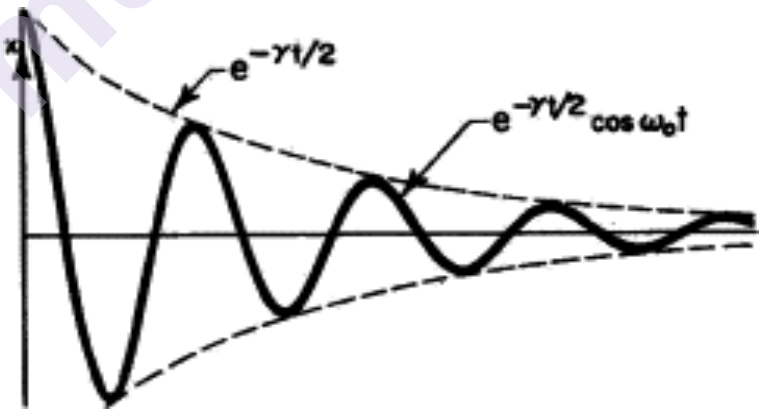
How the model M is connected to the system S.

Feynman's weight on a spring is a convenient place to start. In all three examples below, *S* is a weight on a spring, either a real one or one that we propose to construct. But the connection takes a different form in each example.

(A) The model is a computer mockup in virtual reality. It looks like the real thing, and maybe sounds like the real thing if the programmer was having fun. In this case the connection with the system is *resemblance* or *similarity*; we say that the model is *pictorial*.

In functional modelling the modeler will sometimes turn an early stage of the specification into a toy working system, called a *prototype*. The prototype is a pictorial model of the final system. It shows how the final system will operate, by working more or less like the final system but maybe with some features missing.

(B) The model is a graph, for example



This graph describes the system less directly. Left to right in the graph represents time, up and down represents the vertical distance of the center of mass of the weight from its resting position. In both dimensions a distance in the graph is proportional to a distance in space or time. A model that can be read in this way, by taking some dimensions in the model as corresponding to some dimensions in the system, is called an *analogue* model.

In hydraulic and aeronautical engineering one often meets *scale models*. These are analogue models where the dimensions of the final system are accurately scaled up or down (usually down) so that the model is a more convenient size than the final system. But if all the dimensions are scaled down in a ratio r, then the areas are scaled down in ratio r^2 and the volumes (and hence the weights) in ratio r^3. So given the laws of physics, how should we scale the time if we want the behavior of the model to predict the behavior of the system?

Dimensional analysis answers this question.

A model can be both pictorial and analogue. For example, the architect's model is both. But the model is clearly not a pictorial model; it doesn't look anything like a weight on a spring.

(C) Feynman himself models his system with an equation, (6). His equation is a piece of text which makes a statement about the system. We call it a *textual* model.

Some fields have developed specialist notations for their subject matter. Generally, these notations are textual, in the sense that they build up expressions from a finite alphabet, though there may be pictorial reasons why one symbol was chosen rather than another. Often, they are meant to be written and read rather than spoken. Musical scores are an obvious example. There is no inherent problem about translating the score of a Beethoven piano sonata symbol by symbol into English ('The time signature is common time, the first bar begins with a minim at *g* in the right hand … ') in such a way that the original score could be recovered from the English-though I can't think why anybody would want to do it. The analogue model doesn't translate into English in any similar way.

Another example of a textual notation is Universal Modelling Language (UML), which is often used in early stages of software modelling; it's less specialist than musical scores but still very limited in what it can express.

The diagram notation of Barwise and Etchemin's logic software *Hyper proof* [Barwise and Etchemin, 1994] looks pictorial but can be read as textual. The upper part of the screen carries pictures of a chessboard with various objects arranged on it, while the lower part carries conventional logical formulas. Built-in rules (Observe and Apply) carry information from pictures to formulas and from formulas to pictures. Although we can say in formulas anything we can say in pictures, and to some extent vice versa, the two kinds of representation encourage different habits of reasoning.

Keith Stenning [Stenning, 2002] reports some valuable experiments on how students with different cognitive styles use *Hyperproof*. His surveys different forms of representation from a cognitive point of view.

Textual models are particularly important for us because in principle Tarski's semantic analysis applies to them. Tarski himself said

Whoever wishes … to pursue the semantics of colloquial language with the help of exact methods will be driven first to undertake the thankless task of a reform of this language…. It may, however, be doubted whether the language of everyday life, after being 'rationalized', in this way, would still preserve its naturalness and whether it would not rather take on the characteristic features of the formalized languages.

Tarski may have intended these remarks to discourage people from extending his semantic theory beyond the case of formalized languages. But today his theory is applied very generally, and the 'rationalization', that he refers to is taken as part of the job of a semanticist. For example the diagrams of Barwise and Etchemendy (above) are studied in this spirit.

Very many models are text from the outset, or can be read as text. One important case that we need to consider is computer models. For example models for wind turbines are usually presented as computer programs together with some accompanying theory to justify the programs. For semantic analysis we need to be more precise about exactly what feature of a computer model is the actual model. Let me give my own answer; other analysts may see things differently.

The information about the proposed wind turbine is got by running the program. So we should count the model as being the *output* of the program. The output may include text printed on the screen or saved in a file; in this respect the model is textual. The output may also consist of pictures on the screen, or graphs; in this respect the model is pictorial, and possibly also analogue. Dynamic real-time simulations are certainly analogue; they may include sound as well as graphics.

Often the same program can be run with different parameters. (For example in wind turbine modelling one uses programs that simulate wind conditions and are seeded with a random number.

For practical purposes these programs can be thought of as allowing infinitely many parameter values.) Only a few of these runs will ever take place; the rest are virtual outputs. So we have to allow that a textual model can consist of virtual text-or perhaps better, it can consist of a family of different virtual texts.

On this reading, the computer program itself is not the model; it is a device for generating the model. Also the background theory supporting the program is not the model-though often it contains explanatory models of other things.

Feynman's example throws up a further object that we need to notice. This is the function *f* which, for any real numbers $A, \gamma, t, \omega_\gamma$, takes the value

(13)$f(A,\gamma,t.\omega\gamma)=Ae-\gamma t/2\cos\omega\gamma t.$

This function is an abstract object. For definiteness some people give it a set-theoretic form by identifying it with a set of ordered 5-tuples of real numbers. Although the function clearly bears some close relationship to the equation , it's a wholly different kind of object. We can't put it on a page or a screen, or make it out of wood or plaster of paris. In short it is not 'accessible', to us in any direct way.

We can only have any cognitive relationship to it through some description of it-for example the equation (6). For this reason I think we should hesitate to call the function a 'model', of the spring-weight system. (Later we will see that it's closer to a semantic model, though it isn't quite that either.) Nor should we confuse functions in this sense with the 'function', of an artefact as in functional modelling (on which see the chapter by Vermaas and Garbacz in this Volume).

Reference book pdf link

https://www.researchgate.net/publication/289220287_Semantic_Analysis

A Coursebook SECOND EDITION JAMES R. HURFORD Professor of General Linguistics, University of Edinburgh BRENDAN HEASLEY Consultant (Postgraduate Training), Sharjah Women's College, United Arab Emirates MICHAEL B. SMITH Associate Professor of Linguistics, Oakland University

❄❄❄❄❄❄❄

8

TEXT SUMMARIZATION, TEXT CLASSIFICATION

Unit Structure

8.0 OBJECTIVES

An essential NLP task is text summarization. Extraction and abstraction are the two basic kinds of approaches to text summarization. To create a summary, extractive methods choose a subset of existing words, phrases, or sentences from the original text. Abstractive methods, on the other hand, construct an internal semantic representation first, then generate a summary using natural language generation techniques. Such a summary may include words that aren't explicitly mentioned in the original paper. The majority of text summarizing systems uses extractive summarization.

The main issues in text summarization are topic identification, interpretation, summary generation, and evaluation of the created summary. Identifying significant phrases in the document and applying them to pick sentences for inclusion in the summary are critical jobs in extraction-based summarization. Abstraction-based approaches, on the other hand, paraphrase chunks of the source document.

8.1 INTRODUCTION

The text summary is a technique for condensing vast passages of text into manageable chunks. The goal is to develop a logical and fluent summary that only includes the document's major ideas. In machine learning and natural language processing, automatic text summarization is a prevalent challenge (NLP).

The technique has proven to be crucial in quickly and accurately summarizing large texts, which would be costly and time-consuming if done manually.

Before producing the requisite summary texts, machine learning models are frequently trained to comprehend documents and condense the useful information.

The purpose of text summarization Data is to this century what oil was to the previous one, propelled by modern technical breakthroughs. The collection and transmission of massive volumes of data have parachuted into our world today.

With so much data moving in the digital world, machine learning algorithms that can automatically condense lengthy texts and offer accurate summaries that carry the intended contents fluently are needed.

Furthermore, using text summarization reduces reading time, speeds up the research process, and expands the quantity of information that may fit in a given space.

8.2 OVERVIEW

Text summarization is an important activity in Natural Language Processing (NLP) that will undoubtedly have a significant impact on our lives. With the rise of the digital world and its impact on the publishing industry, devoting time to thoughtfully read an article, document, or book to determine its relevance is no longer an option, especially given time constraints.

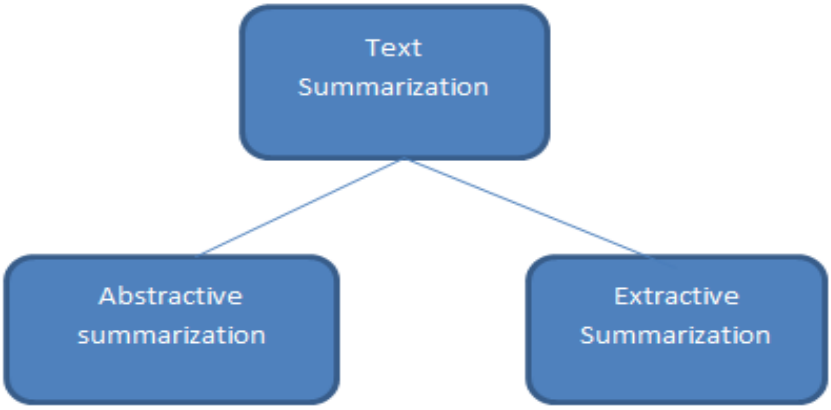
Furthermore, with the growing number of articles being published and the digitization of traditional print magazines, keeping track of the growing number of publications available on the web has become practically impossible. This is where text summary can aid in the reduction of lengthy texts.

THE DEFINITION OF TEXT SUMMARIZATION

Automatic Text Summarization is a technique in which computer software shortens lengthy texts and provides summaries to convey the desired content. It is a prevalent challenge in machine learning and natural language processing (NLP).

The process of constructing a concise, cohesive, and fluent summary of a lengthier text document, which includes highlighting the text's important points, is known as text summarization.

- Extractive Summarization
- Abstractive Summarization



Extractive approaches aim to summarize articles by finding key sentences or phrases from the original text and piecing together sections of it to create a shortened version. The summary is then created using the retrieved sentences.

Summarization of the Abstract unlike extraction, this method depends on advanced natural language algorithms to paraphrase and reduce portions of a document. Abstractive machine learning algorithms can construct new phrases and sentences to capture the meaning of the source content. When done effectively in deep learning issues, such abstraction can help overcome grammatical mistakes.

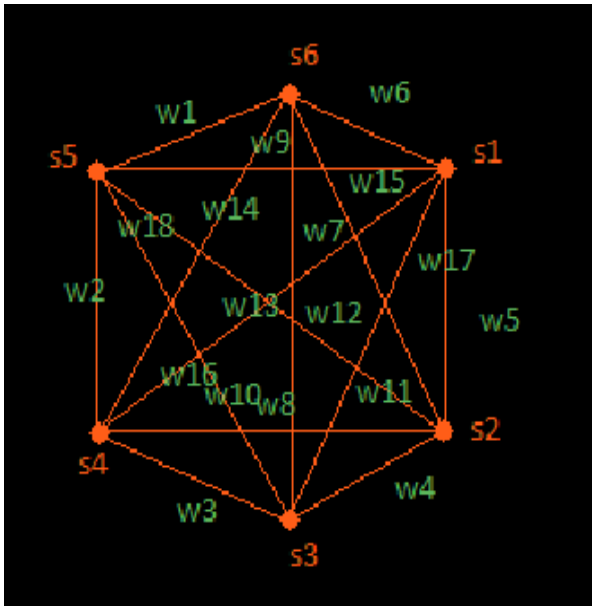
8.3 TEXT SUMMARIZATION- LEXRANK

LexRank is a graph-based unsupervised technique for text summarization. Sentence grading is done using the graph approach. LexRank is a tool for calculating sentence relevance in a graph representation of sentences based on the concept of eigenvector centrality.

In this paradigm, the adjacency matrix of the graph representation of sentences is a connectivity matrix based on intra-sentence cosine similarity. This sentence extraction is primarily concerned with a group of sentences with the same goal in mind, i.e. a centroid sentence is chosen as the mean for all other sentences in the document. The sentences are then graded based on their similarity.

Any multi-set documents can be used as the target document.

Based on Eigen Vector Centrality, a graphical approach was developed. The vertices of the graphs are where sentences are inserted. The cosine similarity measure is used to calculate the weight of the Edges.



Where Si denotes the sentences at the vertices, and Wij denotes the weights on the edges.

Computation of cosine similarity

The bag of words model is used to describe N-dimensional vectors to define similarity, where N is the number of all possible worlds in a given language. The value of the appropriate dimension in the vector representation of the sentence for each word that appears in the sentence is the number of occurrences of the word in the sentence times the idf of the word.

$$\text{idf-modified-cosine}(x,y) = \frac{\sum_{w \in x,y} \text{tf}_{w,x}\text{tf}_{w,y}(\text{idf}_w)^2}{\sqrt{\sum_{x_i \in x}(\text{tf}_{x_i,x}\text{idf}_{x_i})^2} \times \sqrt{\sum_{y_i \in y}(\text{tf}_{y_i,y}\text{idf}_{y_i})^2}}$$

Where $tf_{w,s}$ is the number of occurrences of the word w in the sentence s.

Eigen Vector Centrality and LexRank

To establish the total centrality, each node's contribution is calculated. However, all Nodes in a graphical network do not have to be deemed equally essential. If there are unrelated texts with major important sentences, the centrality of these sentences will be automatically increased. The origin nodes are taken into account to avoid this issue. As a result, here's a quick way to figure out centrality:

$$\mathbf{p} = \mathbf{B}^{\mathrm{T}}\mathbf{p}$$

$$\mathbf{p}^{\mathrm{T}}\mathbf{B} = \mathbf{p}^{\mathrm{T}}$$

Matrix B is the adjacency matrix formed by dividing each element by the associated row sum in the similarity graph. The left eigenvector of matrix B has an eigenvalue of 1 and is denoted by pT.

ALGORITHM

Algortihm to calculate LexRank Scores is as follows:

```
1.MInputAn array S of n sentences, cosine threshold t output: An array L of LexRank scores
2.Array CosineMatrix[n][n];
3.Array Degree[n];
4.Array L[n];
5.for i ?  1 to n do
6.    for j ?  1 to n do
7.       CosineMatrix[i][j] = idf-modified-cosine(S[i],S[j]);
8.       if CosineMatrix[i][j] > t then
9.          CosineMatrix[i][j] = 1;
10.           Degree[i] + +;
11.       end
12.       else
13.          CosineMatrix[i][j] = 0;
14.       end
15.   end
16.end
17.for i ?  1 to n do
18.    for j ?  1 to n do
19.       CosineMatrix[i][j] = CosineMatrix[i][j]/Degree[i];
20.    end
21.end
22. L = PowerMethod(CosineMatrix,n,o);
23. return L;
```


The required arrays for sentences, threshold, LexRank Scores, Cosine matrix, Degree, and Eigen Values are all initialized in the first four phases.

the-idf modified values for the given sentences are used to initialize our matrix in steps 5 to 9. The element is replaced with value 1 if the values are larger than a threshold idf value. If the value is less than the specified threshold, it is replaced with 0. In these stages, we establish a conventional tf-idf table or matrix.

Each value of the cosine matrix is divided by the degree of each node in steps 17 to 21. The matching degree of each node is called degree centrality in this case.

Finally, we calculate the final score using the Power iteration approach.

8.4 OPTIMIZATION BASED APPROACHES FOR SUMMARIZATION

With the growth of the World Wide Web and electronic government services over the last decade, automatic text summarization has gotten a lot of attention (e.g., electronic document management systems). Users can access a vast amount of information thanks to the continued growth of the WWW and online text collections from e-government services. Material overload either wastes a large amount of time viewing all of the information or causes useful information to be overlooked. As a result, new technologies that can successfully process documents are in high demand. In technological surroundings, automatic text summarization is a critical technology for overcoming this barrier. Automatic text summarization technology is improving, and it may give a solution to the information overload problem

The practice of automatically constructing a compressed version of a given text that delivers helpful information to readers is known as document summarising. Summarization has been defined as a reductive transformation of a given text, usually stated as a three-step process: selection of salient bits of text, aggregation and abstraction of information for various selected chunks, and lastly display of the final summary text. This method can be applied to a variety of tasks, including information retrieval, intelligence gathering, data extraction, text mining, and indexing.

Extractive and abstractive approaches to document summarization can be broadly distinguished. In general, an abstract is a summary of concepts/ideas taken from a source that is then "reinterpreted" and given differently, whereas an extract is a summary of units of text taken from a source and presented verbatim. In reality, the vast majority of studies have focused on summary extraction, which takes bits from the source such as keywords, sentences, or even paragraphs to create a summary. Abstractive approaches, in contrast to extractive approaches, are more difficult to adopt since they need more work.

To read source texts and generate new texts, you'll need a lot of domain knowledge. "Reading and understanding the text to recognize its content, which is subsequently compiled in a succinct text" is how abstraction is defined.

The summary might be a single document or a multi-document, depending on the number of papers to be summarised. Only one document can be condensed into a shorter form using single-document summarising. Multi-document summarising reduces a large number of documents into a single summary. It gives users a domain overview of a topic, indicating what is similar and different across multiple papers, as well as relationships between pieces of information in different documents, and lets them zoom in for more information on specific elements of interest. Multi-document summary seeks to extract relevant information about a subject or topic from various texts written about that subject or topic. It accomplishes

knowledge synthesis and knowledge discovery by combining and integrating information from several publications, and it can also be used for knowledge acquisition.

There are two types of extraction-based summarization methods: supervised and unsupervised. Supervised approaches are based on algorithms that use a large number of human-made summaries, and are thus best suited to texts that are relevant to the summarizer model. As a result, they don't always give a sufficient summary for papers that aren't exactly like the model. In addition, when users modify the aim of summarising or the properties of documents, the training data must be rebuilt or the model must be retrained. To train the summarizer, unsupervised methods do not require training data such as human-made summaries. To score and rank sentences using unsupervised approaches, early researchers frequently choose various statistic and linguistic criteria.

On the internet, you can now obtain a wealth of information on people's perspectives on practically any topic. In complicated, high-stakes decision-making processes, the capacity to interpret such data is crucial. A person interested in purchasing a laptop can browse consumer reviews from others who have purchased and utilized the device. Customer feedback on a freshly introduced product at the corporate level might help discover flaws and features that need to be improved.

To express people's opinions to users, effective summarising techniques are required. The creation of a content selection strategy that defines what vital information should be given is a difficult problem when adopting this approach in a specific domain. In general, content selection is a crucial task at the heart of both summarization and NLG, and it's an area where cross-fertilization could be fruitful.

Existing NLG systems typically handle content selection by creating a heuristic based on a number of relevant parameters and optimising it. ILEX (Intelligent Labelling Explorer) is a database-based system for creating labels for groups of objects, such as museum artefacts (O'Donnell et al., 2001). Its content selection approach comprises assigning knowledge elements a heuristic relevance score and returning the things with the highest scores.

Evaluative arguments are generated in GEA (Generator of Evaluative Arguments) to describe an entity as good or negative. An entity is divided into a hierarchy of features, with a relevance score generated individually for each feature depending on the user's preferences and the product's value of that feature. The most relevant characteristics for the current user are chosen during content selection.

8.5 SUMMARIZATION EVALUATION

The purpose of automatic summarization1 is to take an information source, extract content from it, and deliver the most significant content to the user in a condensed form that is sensitive to the demands of the user or

application. Summaries can be user-focused (or topic-focused, or query-focused), that is, tailored to the needs of a specific user or set of users, or they can be 'generic,' that is, directed at a certain—usually broad—readership community. They can be in the form of an extract, which is a summary made entirely of content copied from the input, or an abstract, which is a summary made up of at least some material not found in the input.

Automatic summarization has long been interested in a review, with thorough evaluations dating back to the 1960s, for example. While there hasn't been a lot of agreement on evaluation issues, a reasonably clear picture emerges from looking at the evaluation approaches that have been studied thus far. These strategies will be briefly described and evaluated in this short work.

- Communicated in plain language. There may be a proper answer in circumstances when the output is an answer to a question, but it is difficult to determine what the correct result is in other cases. There's always the chance that a system will produce an excellent summary that differs significantly from any human summary used as a rough approximation to the correct result.
- Review could skyrocket. Because it is easy to repeat, an evaluation that uses a score program rather than human judgments is ideal.

Because summarization entails compression, it's critical to be able to assess summaries at various compression rates. This expands the scale and the evaluation's complexity.

These considerations must be considered since summarising entails presenting information in a way that is sensitive to the demands of the user or application. As a result, the design of an evaluation becomes more difficult.

There are two basic groups of methods for evaluating text summarization (and, indeed, natural language processing systems). The first, a self-evaluation, puts the summarization method to the test. The second, called an extrinsic evaluation, looks at how the summary affects the completion of another job. The coherence and informativeness of summaries have been the focus of intrinsic evaluations. Extrinsic evaluations, on the other hand, have looked at how summarization affects tasks such as relevance assessment, reading comprehension, and so on.

8.6 TEXT CLASSIFICATION

Over the last few decades, text categorization problems have been extensively researched and handled in a variety of real-world applications. Many researchers are currently interested in building applications that use text categorization algorithms, especially in light of recent achievements in Natural Language Processing (NLP) and text mining. Feature extraction, dimension reductions, classifier selection, and assessments are the four processes that most text classification and document

categorization systems go through. The organization and technical implementations of text classification systems are discussed in this study in terms of the pipeline shown in Figure 1.

The raw text data collection serves as the pipeline's initial input. Text data sets, in general, contain text sequences in documents like D = X1, X2,..., XN, where Xi refers to a data point (i.e., document, text segment) with s number of sentences, each sentence containing ws words with lw letters. A class value is assigned to each point from a collection of k discrete value indices.

Then, for our training purposes, we should develop a structured collection called Feature Extraction. The dimensionality reduction step is an optional part of the pipeline that could be part of the classification system (for example, if we use Term Frequency-Inverse Document Frequency (TF-IDF) as our feature extraction and the train set has 200k unique words, computational time will be very expensive, so we could reduce this option by bringing feature space into another dimensional space). Choosing the correct classification algorithm is the most important stage in document categorization.

The evaluation step, which is separated into two parts, is the other half of the pipeline (prediction the test set and evaluating the model). In general, there are four main levels of scope that can be applied to the text classification system:

1. Document-level: The method collects the necessary categories of a full document at the document level.
2. Paragraph level: The algorithm obtains the appropriate categories of a single paragraph at the paragraph level (a portion of a document).
3. Sentence level: Gets the relevant categories of a single sentence (a section of a paragraph) at the sentence level.
4. Sub-sentence level: The algorithm obtains the relevant categories of sub-expressions within a sentence (a piece of a sentence) at the sub-sentence level.

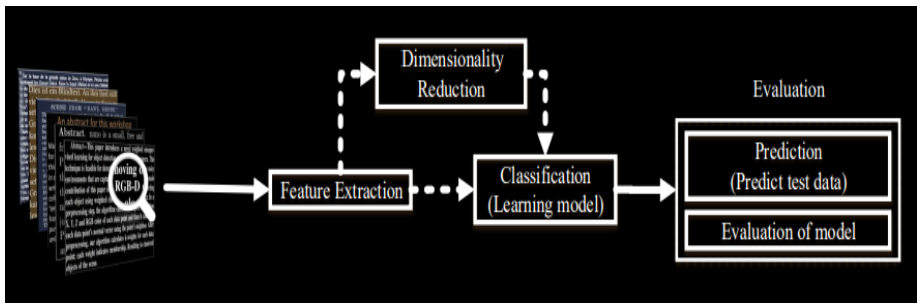


Figure 1 shows a flowchart of the text categorization pipeline.

(I) Text and Document Feature Extraction: Texts and documents are unstructured data sets in general. When employing mathematical modeling as part of a classifier, however, these unstructured text sequences must be translated into a structured feature space. To

begin, the data must be cleaned to remove any extraneous characters or words. Formal feature extraction approaches can be used once the data has been cleansed. Term Frequency-Inverse Document Frequency (TF-IDF), Term Frequency (TF), Word2Vec, and Global Vectors for Word Representation (GloVe) are some of the most prevalent features extraction techniques.

(II) Dimensionality Reduction: Because text or document data sets frequently contain a large number of unique words, data pre-processing operations might take a long time and require a lot of memory. Using low-cost algorithms is a typical approach to this problem. This kind of low-cost algorithm, however, does not perform as well as expected in particular data sets. Many researchers prefer to employ dimensionality reduction to lower the time and memory complexity of their applications to avoid performance degradation. Pre-processing with dimensionality reduction could be more efficient than constructing low-cost classifiers.

Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and non-negative matrix factorization are some of the most prominent dimensionality reduction approaches (NMF). We also go through new strategies for dimensionality reduction in unsupervised feature extraction, such as random projection, autoencoders, and t-distributed stochastic neighbor embedding (t-SNE).

(III) Text Classification Techniques: Choosing the optimal classifier is the most critical phase in the text classification pipeline. We can't properly choose the most efficient model for a text categorization application without a comprehensive conceptual knowledge of each approach. Following that, we discuss ensemble-based learning approaches like boosting and bagging, which have primarily been applied to query learning strategies and text analysis. Logistic regression (LR) is a simple classification approach that has been addressed in most data mining applications. The history of information retrieval can be traced back to the dawn of time.

The Naive Bayes Classifier (NBC) was a very popular application. We'll take a quick look at the Naive Bayes Classifier, which is computationally light and requires very little memory. Non-parametric approaches, such as a k-nearest neighbor, have been investigated and employed as classification tasks. Another prominent technique is the Support Vector Machine (SVM).

For document categorization, it uses a discriminative classifier. This method can be applied to any situation. Bioinformatics, image, video, human activity classification, safety, and other data mining disciplines security, and so on Many researchers utilize this model as a benchmark against which to assess their findings.

Attempts to bring attention to new ideas and contributions

Non-parametric approaches, such as a k-nearest neighbor, have been investigated and employed as classification tasks. Another prominent technique is the Support Vector Machine (SVM). For document categorization, it uses a discriminative classifier. This method can be applied to any situation.

Bioinformatics, image, video, human activity classification, safety, and other data mining disciplines security, and so on Many researchers utilize this model as a benchmark against which to assess their findings. Attempts to bring attention to new ideas and contributions Deep learning approaches have recently outperformed prior machine learning algorithms on problems including image classification, natural language processing, and face recognition, among others. The ability of these deep learning algorithms to represent complicated and non-linear relationships inside data is critical to their success.

(IV) Evaluation: The text categorization pipeline's final step is evaluation. The use and development of text categorization methods require an understanding of how a model works. There are numerous methods for assessing supervised techniques. The simplest way of evaluation is accuracy calculation, but it does not work for unbalanced data sets. F Score, Matthews Correlation Coefficient (MCC), receiver operating characteristics (ROC), and area under the ROC curve are all ways for evaluating text categorization systems (AUC).

feature extractions, dimensionality reduction, and other pipeline steps

Techniques of classification and methods of evaluation In this part, state-of-the-art techniques are compared based on a variety of criteria, including model architecture, the novelty of the work, feature extraction technique, corpus (the data set/s used), validation measure, and limitation of each work. Choosing a feature extraction approach is necessary for determining the optimum system for a certain application. Because some feature extraction strategies are inefficient for a specific application, this decision is entirely dependent on the application's purpose and data source. For a given use, it is effective. for example, is trained on Wikipedia and does not perform as well as TF-IDF when used for short text messages such as short message service (SMS). Furthermore, due to the tiny amount of data, this model cannot be trained as well as other strategies. The categorization technique is the next step in this pipeline, where we briefly discuss the limitations and drawbacks of each technique.

Text classification is a serious difficulty for scholars in many domains and fields. Text categorization methods are extensively used in information retrieval systems and search engine applications. Text classification could be utilized for information filtering (e.g., email and text message spam filtering) as an extension of existing applications. Following that, we discuss document categorization's acceptance in public health and human behavior. Document organization and knowledge management are two

further areas where text classification has aided. Finally, we'll talk about recommender systems, which are common in marketing and advertising.

TEXT PREPROCESSING

For text classification applications, feature extraction and pre-processing are critical tasks. This section introduces strategies for cleaning text data sets, reducing implicit noise and allowing for more accurate analysis. For enlightenment. We also go over two popular strategies for extracting text features: weighted words and word embedding.

CLEANING AND PRE-PROCESSING OF TEXT

Many redundant words, such as stopwords, misspellings, and slang, can be found in most text and document data sets. Noise and superfluous characteristics can degrade system performance in many algorithms, particularly statistical and probabilistic learning algorithms. In this section, we'll go through several text cleaning and pre-processing approaches and methodologies.

TOKENIZATION

Tokenization is a pre-processing technique that divides a stream of text into tokens, which are words, phrases, symbols, or other meaningful pieces. The investigation of the words in a sentence is the primary purpose of this step. Both text classification and text mining necessitate the use of a parser to handle the tokenization of documents, such as the following sentence:

He opted to sleep for another four hours after four hours of sleep.

The tokens in this scenario are as follows:

"After" "four" "hours of" "sleeping," "he" "decided" "to" "sleep" "for" "another" "four."

STOP WORDS

Many terms in text and document classification are not significant enough to be employed in classification algorithms, such as "a", "about", "above", "across", "after", "afterwards", "again", and so on. To deal with these words, the most typical method is to eliminate them from texts and documents.

CAPITALIZATION

To compose a sentence, text and document data points have a variety of caps. Since

When a document has a lot of sentences, different capitalization might be a real pain.

the classification of huge documents Dealing with uneven capitalization is the most usual solution.

is to change all of the letters to lower case. This approach converts all text and document words into

the same feature area, yet it complicates the understanding of some words significantly

(For instance, "US" (United States of America) to "us" (pronoun)) Converters for slang and abbreviations are available.

assist in accounting for these occurrences

ABBREVIATIONS AND SLANG

Other types of text abnormalities that are handled in the pre-processing step include slang and abbreviation. SVM stands for Support Vector Machine, and an abbreviation is a shorter form of a word or phrase that contains mostly first letters from the terms.

Slang is a subset of spoken or written language that has several meanings, such as "lost the plot," which essentially means "gone insane". Converting these terms into formal language is a typical approach to dealing with them

NOISE ABATEMENT

Many unneeded characters, such as punctuation and special characters, can be found in most text and document data sets. Although critical punctuation and special characters are necessary for human comprehension of documents, they can harm categorization systems.

CORRECTIONS TO SPELLING

Spelling correction is a pre-processing step that can be skipped. Typos (short for typographical errors) are widespread in texts and documents, particularly in text data sets from social media (e.g., Twitter). This challenge has been addressed by many algorithms, strategies, and methods in NLP . For researchers, there are a variety of strategies and methods accessible, including hashing-based and context-sensitive spelling correction algorithms, as well as spelling correction utilizing the Trie and Damerau–Levenshtein distance bigrams.

STEMMING

In NLP, a single word might exist in multiple forms (for example, singular and plural noun forms), all of which have the same semantic meaning. Stemming is one way for combining different variants of a word into the same feature space. Text stemming uses linguistic processes like affixation (adding affixes) to modify words to produce variant word forms. The stem of the word "studying," for example, is "study."

LEMMATIZATION

Lemmatization is a natural language processing technique that replaces a word's suffix with a different one or eliminates the suffix to reveal the fundamental word form (lemma)

8.7 LET US SUM UP

The text summary is a technique for condensing vast passages of text into manageable chunks. In machine learning and natural language processing, text summarization is a prevalent challenge. The main issues are topic identification, interpretation, summary generation, and evaluation of the created summary. Text summarization is the process of constructing a concise, cohesive, and fluent summary of a lengthier text document. This involves finding key sentences or phrases from the original text and piecing together sections to create a shortened version.

It is a prevalent challenge in machine learning and natural language processing (NLP). The bag of words model is used to describe N-dimensional vectors to define similarity. Matrix B is the adjacency matrix formed by dividing each element by the associated row sum in the similarity graph. The degree of each node is called degree centrality in this case

8.8 REFFERENCES

- https://www.sciencedirect.com/topics/computer-science/text-summarization
- https://towardsdatascience.com/a-quick-introduction-to-text-summarization-inmachine-learning-3d27ccf18a9f
- https://iq.opengenus.org/lexrank-text-summarization/
- Text Classification Algorithms: A Survey Kamran Kowsari 1,2,* ID , Kiana Jafari Meimandi 1 , Mojtaba Heidarysafa 1 , Sanjana Mendu 1 ID , Laura Barnes 1,2,3 ID and Donald Brown 1,3 I
- Summarization Evaluation: An Overview Inderjeet MANI The MITRE Corporation, W640 11493 Sunset Hills Road Reston, VA 20190-5214, USA
- AN OPTIMIZATION APPROACH TO AUTOMATIC GENERIC DOCUMENT SUMMARIZATION RASIM M. ALGULIEV, RAMIZ M. ALIGULIYEV, AND CHINGIZ A. MEHDIYEV Institute of Information Technology of Azerbaijan National Academy of Sciences, Baku, Azerbaijan
- “Optimization-based Content Selection for Opinion Summarization Jackie Chi Kit Cheung Department of Computer Science University of Toronto Toronto”, ON, M5S 3G4, Canada jcheung@cs.toronto.edu Giuseppe Carenini and Raymond T. N

8.9 BIBLIOGRAPHY

- https://www.sciencedirect.com/topics/computer-science/text-summarization
- https://towardsdatascience.com/a-quick-introduction-to-text-summarization-inmachine-learning-3d27ccf18a9f
- https://iq.opengenus.org/lexrank-text-summarization/

- "Text Classification Algorithms: A Survey Kamran Kowsari" 1,2,* ID , Kiana Jafari Meimandi 1 , Mojtaba Heidarysafa 1 , Sanjana Mendu 1 ID , Laura Barnes 1,2,3 ID and Donald Brown 1,3 I
- "Summarization Evaluation: An Overview Inderjeet MANI The MITRE Corporation", W640 11493 Sunset Hills Road Reston, VA 20190-5214, USA
- "AN OPTIMIZATION APPROACH TO AUTOMATIC GENERIC DOCUMENT SUMMARIZATION RASIM M. ALGULIEV, RAMIZ M. ALIGULIYEV, AND CHINGIZ A. MEHDIYEV" Institute of Information Technology of Azerbaijan National Academy of Sciences, Baku, Azerbaijan
- "Optimization-based Content Selection for Opinion Summarization Jackie Chi Kit Cheung Department of Computer Science University of Toronto Toronto", ON, M5S 3G4, Canada jcheung@cs.toronto.edu Giuseppe Carenini and Raymond T. N

8.10 UNIT EXCERISE

1. What should the input dimension for text summarization be?
2. Is it feasible to categorize a conversation's topic?
3. What are the most effective methods for text simplification?
4. How would you address a problem of text summarization?
5. How do you go about putting text summarising into practice?
6. What does a categorization text serve?
7. What are the various approaches to text classification?
8. What exactly is a text classification issue?

❄❄❄❄❄❄❄

9

SENTIMENT ANALYSIS AND OPINION MINING - I

Unit Structure

9. OBJECTIVE

- Opinion mining and sentiment analysis done from big data
- Opinion mining in data mining
- Sentiment analysis in text mining
- Explain sentiment analysis

9.1 INTRODUCTION

Sentiment analysis systems help organizations gather insights from unorganized and unstructured text that comes from online sources such as emails, blog posts, support tickets, web chats, social media channels, forums and comments. Algorithms replace manual data processing by implementing rule-based, automatic or hybrid methods. Rule-based systems perform sentiment analysis based on predefined, lexicon-based rules while automatic systems learn from data with machine learning techniques. A hybrid sentiment analysis combines both approaches.

In addition to identifying sentiment, opinion mining can extract the polarity (or the amount of positivity and negativity), subject and opinion holder within the text. Furthermore, sentiment analysis can be applied to varying scopes such as document, paragraph, sentence and sub-sentence levels.

Vendors that offer sentiment analysis platforms or SaaS products include Brandwatch, Hootsuite, Lexalytics, NetBase, Sprout Social, Sysomos and Zoho. Businesses that use these tools can review customer feedback more regularly and proactively respond to changes of opinion within the market.

Sentiment analysis, also called opinion mining, is the field of study that analyzes people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes. It represents a large problem space. There are also many names and slightly different tasks, e.g., sentiment analysis, opinion mining, opinion extraction, sentiment mining, subjectivity analysis, affect analysis, emotion analysis, review mining, etc. However, they are now all under the umbrella of sentiment analysis or opinion mining. While in industry, the term sentiment analysis is more commonly used, but in academia both sentiment analysis and opinion mining are frequently employed. They basically represent the same field of study., we use the terms sentiment analysis and opinion mining interchangeably.

To simplify the presentation, throughout this book we will use the term opinion to denote opinion, sentiment, evaluation, appraisal, attitude, and emotion. However, these concepts are not equivalent. We will distinguish them when needed. The meaning of opinion itself is still very broad. Sentiment analysis and opinion mining mainly focuses on opinions which express or imply positive or negative sentiments. Although linguistics and natural language processing (NLP) have a long history, little research had been done about people's opinions and sentiments before the year 2000. Since then, the field has become a very active research area. There are several reasons for this. First, it has a wide arrange of applications, almost in every domain.

The industry surrounding sentiment analysis has also flourished due to the proliferation of commercial applications. This provides a strong motivation for research. Second, it offers many challenging research problems, which had never been studied before. Sentiment Analysis and Opinion Mining 8 the first time in human history, we now have a huge volume of opinionated data in the social media on the Web. Without this data, a lot of research would not have been possible. Not surprisingly, the inception and the rapid growth of sentiment analysis coincide with those of the social media. In fact, sentiment analysis is now right at the center of the social media research. Hence, research in sentiment analysis not only has an important impact on NLP, but may also have a profound impact on management sciences, political science, economics, and social sciences as they are all affected by people's opinions. Although the sentiment analysis research mainly started from early 2000, there were some earlier work on interpretation of metaphors, sentiment adjectives, subjectivity, view points, and affects (Hatzivassiloglou and McKeown, 1997; Hearst, 1992; Wiebe, 1990; Wiebe, 1994; Wiebe, Bruce and O'Hara, 1999).

9.1.1 Definition of Sentiment Analysis

Sentiment analysis is **contextual mining of text which identifies and extracts subjective information in source material, and helping a business to understand the social sentiment of their brand, product or service while monitoring online conversations**.

Sentiment Analysis is the most common text classification tool that analyses an incoming message and tells whether the underlying sentiment is positive, negative our neutral.

In linguistics, **semantic analysis** is the process of relating syntactic structures, from the levels of phrases, clauses, sentences and paragraphs to the level of the writing as a whole, to their language-independent meanings. It also involves removing features specific to particular linguistic and cultural contexts, to the extent that such a project is possible. The elements of idiom and figurative speech, being cultural, are often also converted into relatively invariant meanings in semantic analysis. Semantics, although related to pragmatics, is distinct in that the former deals with word or sentence choice in any given context, while pragmatics considers the unique or particular meaning derived from context or tone. To reiterate in different terms, semantics is about universally coded meaning, and pragmatics, the meaning encoded in words that is then interpreted by an audience.[1]

Semantic analysis can begin with the relationship between individual words. This requires an understanding of lexical hierarchy, including hyponymy and hypernymy, meronomy, polysemy, synonyms, antonyms, and homonyms.[2] It also relates to concepts like connotation (semiotics) and collocation, which is the particular combination of words that can be or frequently are surrounding a single word. This can include idioms, metaphor, and simile, like, "white as a ghost."

Some important remarks about this definition are in order:

1. In this definition, we purposely use subscripts to emphasize that the five pieces of information in the quintuple must correspond to one another. That is, the opinion must be given by opinion holder about aspect of entity at time. Any mismatch is an error.

2. The five components are essential. Missing any of them is problematic in general. For example, if we do not have the time component, we will not be able to analyze opinions on an entity according to time, which is often very important in practice because an opinion two years ago and an opinion yesterday is not the same. Without opinion holder is also problematic. For example, in the sentence “the mayor is loved by the people in the city, but he has been criticized by the state government,” the two opinion holders, “people in the city” and “state government,” are clearly important for applications.

3. The definition covers most but not all possible facets of the semantic meaning of an opinion, which can be arbitrarily complex. For example, it does not cover the situation in “The view finder and the lens are too close,” which expresses an opinion on the distance of two parts. It also does not cover the context of the opinion, e g., “This car is too small for a tall person,” which does not say the car is too small for everyone. “Tall person” is the context here. Note also that in the original definition of entity, it is a hierarchy of parts, sub-

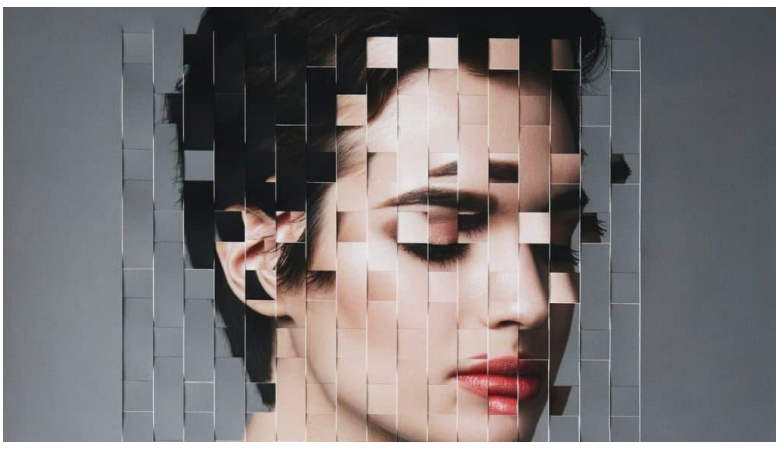
parts, and so on. Every part can have its set of attributes. Due to the simplification, the quintuple representation can result in information loss. For example, “ink” is a part/component of a printer. In a printer review, one wrote “The ink of this printer is expensive.” This does not say that the printer is expensive (which indicates the aspect price). If one does not care about any attribute of the ink, this sentence just gives a negative opinion to the ink, which is an aspect of the printer entity. However, if one also wants to study opinions about different aspects of the ink, e.g., price and quality, the ink needs to be treated as a separate entity. Then, the quintuple representation still applies, but the part-of relationship needs to be saved. Of course, conceptually we can also expand the representation of opinion target using a nested relation. Despite the limitations, the definition does cover the essential information of an opinion which is sufficient for most applications.

4. This definition provides a framework to transform unstructured text to structured data. The quintuple above is basically a database schema, based on which the extracted opinions can be put into a database table. Then a rich set of qualitative, quantitative, and trend analyses of opinions can be performed using the whole suite of database management systems (DBMS) and OLAP tools.

5. The opinion defined here is just one type of opinion, called regular opinion. Another type is comparative opinion (Jindal and Liu, 2006b; Liu, 2006 and 2011), which needs a different definition. Section 2.3 will discuss different types of opinions. Chapter 8 defines and analyzes comparative opinions. For the rest of this section, we only focus on regular opinions. For simplicity, we just called them opinions.

9.2 SENTIMENT ANALYSIS TYPES

Modern-day sentiment analysis approaches are classified into three categories: **knowledge-based, statistical, and hybrid**. Here's how to perform sentiment analysis. Knowledge-Based: This approach included the classification of text based on words that emanate emotion.



One of the things that define us as human Is our ability to showcase a wide range of emotions. We can be sad, excited, worried, and angry within seconds. We can learn and analyze situations like no other animal can. These are just some of the things that makes us unique and special.

This is evident during shopping, too. Almost all purchases are emotion-driven. It could be out of fear, jealousy, or happiness. With emotions playing a critical role in customer behavior, it has become essential for brands to analyze the sentiments of their consumers.

Here's where the concept of sentiment analysis comes into play. In this post, we'll discuss the idea and different types of sentiment analysis.

Sentiment Analysis Research

As discussed above, pervasive real-life applications are only part of the reason why sentiment analysis is a popular research problem. It is also highly challenging as a NLP research topic, and covers many novel subproblems as we will see later. Additionally, there was little research before the year 2000 in either NLP or in linguistics. Part of the reason is that before then there was little opinion text available in digital forms. Since the year 2000, the field has grown rapidly to become one of the most active research areas in NLP. It is also widely researched in data mining, Web mining, and information retrieval. In fact, it has spread from computer science to management sciences (Archak, Ghose and Ipeirotis, 2007; Chen and Xie, 2008; Das and Chen, 2007; Dellarocas, Zhang and Awad, 2007; Ghose, Ipeirotis and Sundararajan, 2007; Hu, Pavlou and Zhang, 2006; Park, Lee and Han, 2007).

Different Levels of Analysis

Introduction to the main research problems based on the level of granularities of the existing research. In general, sentiment analysis has been investigated mainly at three levels:

Document level: The task at this level is to classify whether a whole opinion document expresses a positive or negative sentiment. For example, given a product review, the system determines whether the review expresses an overall positive or negative opinion about the product. This task is commonly Sentiment Analysis and Opinion Mining 11 known as document-level sentiment classification. This level of analysis assumes that each document expresses opinions on a single entity (e.g., a single product). Thus, it is not applicable to documents which evaluate or compare multiple entities.

Sentence level: The task at this level goes to the sentences and determines whether each sentence expressed a positive, negative, or neutral opinion. Neutral usually means no opinion. This level of analysis is closely related to subjectivity classification (Wiebe, Bruce and O'Hara, 1999), which distinguishes sentences (called objective sentences) that express factual information from sentences (called subjective sentences) that express subjective views and opinions. However, we should note that subjectivity

is not equivalent to sentiment as many objective sentences can imply opinions, e.g., "We bought the car last month and the windshield wiper has fallen off." Researchers have also analyzed clauses (Wilson, Wiebe and Hwa, 2004), but the clause level is still not enough, e.g., "Apple is doing very well in this lousy economy." Entity and Aspect level: Both the document level and the sentence level analyses do not discover what exactly people liked and did not like.

Aspect level performs finer-grained analysis. Aspect level was earlier called feature level (feature-based opinion mining and summarization) . Instead of looking at language constructs (documents, paragraphs, sentences, clauses or phrases), aspect level directly looks at the opinion itself. It is based on the idea that an opinion consists of a sentiment (positive or negative) and a target (of opinion). An opinion without its target being identified is of limited use. Realizing the importance of opinion targets also helps us understand the sentiment analysis problem better. For example, although the sentence "although the service is not that great, I still love this restaurant" clearly has a positive tone, we cannot say that this sentence is entirely positive. In fact, the sentence is positive about the restaurant (emphasized), but negative about its service (not emphasized). In many applications, opinion targets are described by entities and/or their different aspects.

Thus, the goal of this level of analysis is to discover sentiments on entities and/or their aspects. For example, the sentence "The iPhone's call quality is good, but its battery life is short" evaluates two aspects, call quality and battery life, of iPhone (entity). The sentiment on iPhone's call quality is positive, but the sentiment on its battery life is negative. The call quality and battery life of iPhone are the opinion targets. Based on this level of analysis, a structured summary of opinions about entities and their aspects can be produced, which turns unstructured text to structured data and can be used for all kinds of qualitative and quantitative analyses. Both the document level and sentence level classifications are already Sentiment Analysis and Opinion Mining 12 highly challenging. The aspect-level is even more difficult. It consists of several sub-problems, which we will discuss.

To make things even more interesting and challenging, there are two types of opinions, i.e., **regular opinions and comparative opinions** .

A regular opinion expresses a sentiment only on a particular entity or an aspect of the entity, e.g., "Coke tastes very good," which expresses a positive sentiment on the aspect taste of Coke.

A comparative opinion compares multiple entities based on some of their shared aspects, e.g., "Coke tastes better than Pepsi," which compares Coke and Pepsi based on their tastes (an aspect) and expresses a preference for Coke ,

What Is Sentiment Analysis?

First things first, what is sentiment analysis? Sentiment analysis is a type of market analysis that includes the use of text analysis, biometrics, natural language processing (NLP), and computational linguistics to recognize the state of the said information.

In simple terms, it's the process of determining whether a piece of content – email, social media post, or article – is negative, positive, or neutral. Sentiment analysis enables you to ascertain public opinion and understand consumer experiences.

But why should you even bother about sentiment analysis? For starters, it's extremely helpful in social media monitoring. It helps you gauge public opinion on certain topics on an enormous scale.

Besides, it can play a pivotal role in market research and customer service. With sentiment analysis, you can see what people think about your products or your competitors' products. This helps you understand customer attitudes and preferences, enabling you to make informed decisions.

Types of Sentiment Analysis

People have a wide range of emotions – sad or happy, interested or uninterested, and positive or negative. Different sentiment analysis models are available to capture this variety of emotions.

Let's look at the most important types of sentiment analysis.

1. Fine-Grained

This sentiment analysis model helps you derive polarity precision. You can conduct a sentiment analysis across the following polarity categories: very positive, positive, neutral, negative, or very negative. Fine-grained sentiment analysis is helpful for the study of reviews and ratings.

For a rating scale from 1 to 5, you can consider 1 as very negative and five as very positive. For a scale from 1 to 10, you can consider 1-2 as very negative and 9-10 as very positive.

2. Aspect-Based

While fine-grained analysis helps you determine the overall polarity of your customer reviews, aspect-based analysis delves deeper. It helps you determine the particular aspects people are talking about.

Let's say; you're a mobile phone manufacturer, and you get a customer review stating, "the camera struggles in artificial lighting conditions."

With aspect-based analysis, you can determine that the reviewer has commented on something "negative" about the "camera."

3. Emotion Detection

As the name suggests, emotion detection helps you detect emotions. This can include anger, sadness, happiness, frustration, fear, worry, panic, etc. Emotion detection systems typically use lexicons – a collection of words that convey certain emotions. Some advanced classifiers also utilize robust machine learning (ML) algorithms.

It's recommended to use ML over lexicons because people express emotions in a myriad of ways. Take this line, for example: "This product is about to kill me." This line may express feelings of fear and panic.

A similar line – this product is killing it for me – has an entirely different and positive meaning. But the word "kill" might be associated with fear or panic in the lexicon. This may lead to inaccurate emotion detection.

4. Intent Analysis

Accurately determining consumer intent can save companies time, money, and effort. So many times, businesses end up chasing consumers that don't plan to buy anytime soon. Accurate intent analysis can resolve this hurdle.

The intent analysis helps you identify the intent of the consumer – whether the customer intends to purchase or is just browsing around.

If the customer is willing to purchase, you can track them and target them with advertisements. If a consumer isn't ready to buy, you can save your time and resources by not advertising to them.

9.2.1 Sentiment Analysis - Affective lexicons

Definition:

- It is a *Natural Language Processing* task; *Sentiment Analysis* refers to finding patterns in data and inferring the emotion of the given piece of information which could be classified into one of these categories:
    - Negative
    - Neutral
    - Positive

Utility:

- We are truly living in the information age where the data is generated by both humans and machines at an unprecedented rate, therefore it's nearly impossible to gain insights into such data for making intelligent decisions, manually. One such insight is assessing/calculating the sentiment of a big (rather huge) dataset
- Software to the rescue yet again. One way of assessing, rather calculating the emotion on top of such a huge dataset is by way of employing sentiment analysis techniques (more on those later)

Applications and examples:

- Market sentiment – Huge amount of data is being generated in the global financial world, every second. It is imperative for investors to remain on top of such information so that they can manage their portfolios more effectively. One way of gauging the market sentiment is by way of implementing sentiment analyzing tools
- Gauging the popularity of the products, books, movies, songs, services by analyzing the data generated from different sources such as tweets, reviews, feedback forums, gestures (swipes, likes) etc.

Now that we know what sentiment analysis is on a broader level, let's dig a little deeper. Analyzing data can be quite tricky because the data, in almost all the cases, is unstructured, opinionated and in some cases incomplete- meaning it is addressing only a fraction of the subject matter and therefore unsuitable for calculating the emotion. This data must be cleaned for it to be turned into something actionable. It is intended, in most cases, to focus on deducing the polarity (good or bad, spam or ham, black or white etc.) from the given piece of information as the opinions can be nuanced and can lead to ambiguity. Putting it another way, the "problem" should be framed in a certain way so that a binary classification (polarity) can be deduced from it. Now I know most of you will say, what about that "neutral" classification between "negative" and "positive" classes? Fair question. Well the answer is, if the deduction/calculation comes out to be somewhere in the middle of 0 (negative) and 1 (positive), let's say 0.5 then it is obvious that the given piece of information also known as ***problem instance*** can neither be classified as *negative* nor *positive* and therefore the classification of *neutral* seems more appropriate.

Analysis Mechanisms:

On a higher level, there are two techniques that can be used for performing sentiment analysis in an automated manner, these are: *Rule-based* and *Machine Learning based*. *I will explore the former in this blog and take up the latter in part 2 of the series*.

- **Rule based**

Rule based sentiment analysis refers to the study conducted by the language experts. The outcome of this study is a set of rules (also known as *lexicon* or *sentiment lexicon*) according to which the words classified are either *positive* or *negative* along with their corresponding intensity measure.

Generally, the following steps are needed to be performed while applying the rule-based approach:

- Extract the data
- Tokenize text. The task of splitting the text into individual words

- Stop words removal. Those words which do not carry any significant meaning and should not be used for the analysis activity. Examples of stop words are: a, an, the, they, while etc.
- Punctuation removal (in some cases)
- Running the *preprocessed* text against the sentiment lexicon which should provide the number/measurement corresponding to the inferred emotion
- Example:
- Consider this problem instance: "Sam is a great guy."
- Remove stop words and punctuation
- Running the lexicon on the preprocessed data, returns a **positive sentiment** score/measurement because of the presence of a positive word "great" in the input data.
- This is a very simple example. Real world data is much more complex and nuanced e.g. your problem sentiment may contain sarcasm (where seemingly positive words carry negative meaning or vice versa), shorthand, abbreviations, different spellings (e.g. flavor vs flavour), misspelled words, punctuation (especially question marks), slang and of course, emojis.
- In order to tackle the complex data for analysis is to make use of sophisticated lexicons that can take into consideration the intensity of the words (e.g. if a word is positive then how positive it is, basically there is a difference between good, great and amazing and that is represented by the intensity assigned to a given word), the subjectivity or objectivity of the word and the context also. There are several such lexicons available. Following are a couple of popular ones:
  - **VADER** (Valence Aware Dictionary and Sentiment Reasoner): Widely used in analyzing sentiment on social media text because it has been specifically *attuned* to analyze sentiments expressed in social media (as per the linked docs). It now comes out of the box in Natural language toolkit, NLTK. VADER is sensitive to both polarity and the intensity. Here is how to read the measurements:
  - -4: extremely negative
  - 4: extremely positive
  - 0: Neutral or N/A

Examples (taken from the docs):

TextBlob: Very useful NLP library that comes prepackaged with its own sentiment analysis functionality. It is also based on NLTK. The sentiment property of the api/library returns polarity and subjectivity.

- Polarity range: -1.0 to 1.0
- Subjectivity range: 0.0 – 1.0 (0.0 is very objective and 1.0 is very subjective)
- Example (from the docs):

- Problem instance: "Textblob is amazingly simple to use. What great fun!"
- Polarity: 39166666666666666
- Subjectivity: 0.39166666666666666
- **Polarity:** as mentioned earlier, it is a measurement of how positive or negative the given problem instance is. In other words, it is related to the emotion of the given text
- **Subjectivity:** It refers to opinions or views (can be allegations, expressions or speculations) that need to be analyzed in the given context of the problem statement. The more subjective the instance is, the less objective it becomes and vice versa. A subjective instance (e.g. a sentence) may or may not carry any emotion. Examples:
- "Sam likes watching football"
- "Sam is driving to work"
- Sentiwordnet: This is also built into NLTK. It is used for opinion mining. This helps in deducing the polarity information from the given problem instance. SWN extends wordnet which is a lexical database of words (the relationship between words, hence the term *net*), developed at Princeton and is a part of NLTK corpus. Here I'd focus primarily on *synsets*, which are the logical groupings of nouns, verbs, adjectives and adverbs, into sets or collections of cognitive synonyms, hence the term *synset* (*you can read more on wordnet online, I found the links this, this and this quite helpful*).

Coming back to Sentiwordnet and its relationship with wordnet; Sentiwordnet assigns polarity to each synset.

Example:

- Let's take a word *hunger*
- The wordnet part:
- Get a list of all the synsets against *hunger*
- The Sentiwordnet (swn) part:
- Get the polarity for all those synsets
- Use one or maybe a combination of all of them to determine the polarity score for that word (*hunger in ours)*. The following diagrams

9.2.2 Learning affective lexicons

More generally, we use the phrase affective lexicon to refer to that **subset of words in a language that are about affect or affective conditions**. Many, but by no means all, of the words in the affective lexicon refer to emotions. Emotion lexicons describe the affective meaning of words and thus constitute a centerpiece for advanced sentiment and emotion analysis. Yet, manually curated lexicons are only available for a handful of languages, leaving most languages of the world without such a precious resource for downstream applications. Even worse, their coverage is often limited both in terms of the lexical units they contain and the emotional

variables they feature. In order to break this bottleneck, we here introduce a methodology for creating almost arbitrarily large emotion lexicons for any target language. Our approach requires nothing but a source language emotion lexicon, a bilingual word translation model, and a target language embedding model. Fulfilling these requirements for 91 languages, we are able to generate representationally rich high-coverage lexicons comprising eight emotional variables with more than 100k lexical entries each. We evaluated the automatically generated lexicons against human judgment from 26 datasets, spanning 12 typologically diverse languages, and found that our approach produces results in line with state-of-the-art monolingual approaches to lexicon creation and even surpasses human reliability for some languages and variables

Sentiment Lexicon and Its Issues

Not surprisingly, the most important indicators of sentiments are sentiment words, also called opinion words. These are words that are commonly used to express positive or negative sentiments. For example, good, wonderful, and amazing are positive sentiment words, and bad, poor, and terrible are negative sentiment words. Apart from individual words, there are also phrases and idioms, e.g., cost someone an arm and a leg. Sentiment words and phrases are instrumental to sentiment analysis for obvious reasons. A list of such words and phrases is called a sentiment lexicon (or opinion lexicon). Over the years, researchers have designed numerous algorithms to compile such lexicons. Although sentiment words and phrases are important for sentiment analysis, only using them is far from sufficient. The problem is much more complex. In other words, we can say that sentiment lexicon is necessary but not sufficient for sentiment analysis.

Below, we highlight several issues:

1. A positive or negative sentiment word may have opposite orientations in different application domains. For example, “suck” usually indicates negative sentiment, e.g., “This camera sucks,” but it can also imply positive sentiment, e.g., “This vacuum cleaner really sucks.”

2. A sentence containing sentiment words may not express any sentiment. This phenomenon happens frequently in several types of sentences. Question (interrogative) sentences and conditional sentences are two important types, e.g., “Can you tell me which Sony camera is good?” and “If I can find a good camera in the shop, I will buy it.” Both these sentences contain the sentiment word “good”, but neither expresses a positive or negative opinion on any specific camera. However, not all conditional sentences or interrogative sentences express no sentiments, e.g., “Does anyone know how to repair this terrible printer” and “If you Sentiment Analysis and Opinion Mining 13 are looking for a good car, get Toyota Camry.”

Sarcastic sentences with or without sentiment words are hard to deal with, e.g., "What a great car! It stopped working in two days." Sarcasms are not so common in consumer reviews about products and services, but are very common in political discussions, which make political opinions hard to deal with. Many sentences without sentiment words can also imply opinions. Many of these sentences are actually objective sentences that are used to express some factual information. Again, there are many types of such sentences. Here we just give two examples. The sentence "This washer uses a lot of water" implies a negative sentiment about the washer since it uses a lot of resource (water). The sentence "After sleeping on the mattress for two days, a valley has formed in the middle" expresses a negative opinion about the mattress. This sentence is objective as it states a fact. All these sentences have no sentiment words. These issues all present major challenges. In fact, these are just some of the difficult problems.

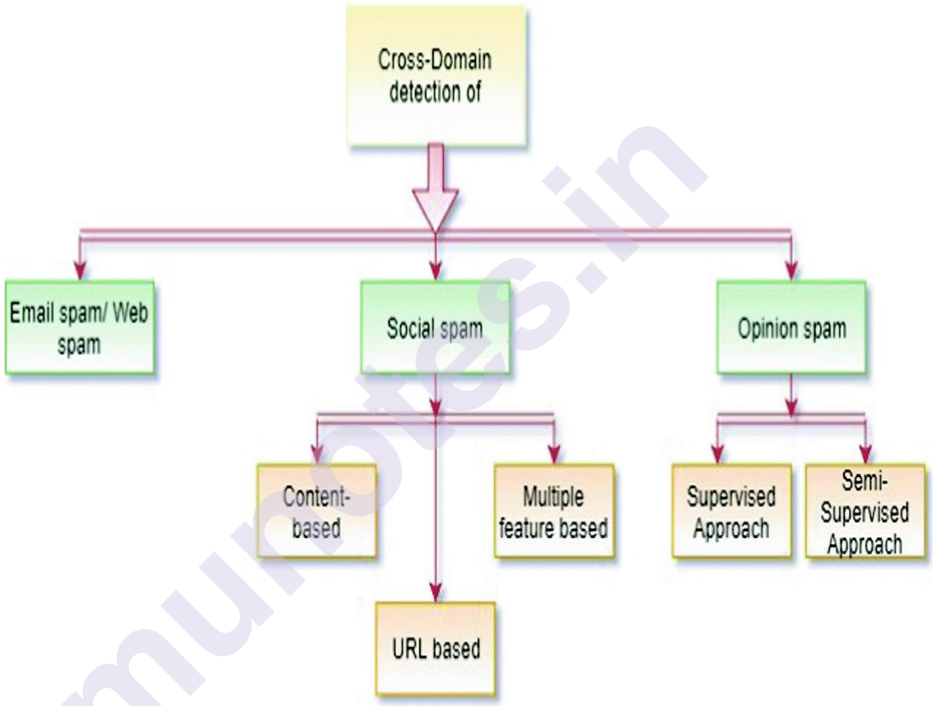
Natural Language Processing Issues Finally, we must not forget sentiment analysis is a NLP problem. It touches every aspect of NLP, e.g., coreference resolution, negation handling, and word sense disambiguation, which add more difficulties since these are not solved problems in NLP. However, it is also useful to realize that sentiment analysis is a highly restricted NLP problem because the system does not need to fully understand the semantics of each sentence or document but only needs to understand some aspects of it, i.e., positive or negative sentiments and their target entities or topics. In this sense, sentiment analysis offers a great platform for NLP researchers to make tangible progresses on all fronts of NLP with the potential of making a huge practical impact content.

Researchers now also have a much better understanding of the whole spectrum of the problem, its structure, and core issues. Numerous new (formal) models and methods have been proposed. The research has not only deepened but also broadened significantly. Earlier research in the field mainly focused on classifying the sentiment or subjectivity expressed in documents or sentences, which is insufficient for most real-life applications. Practical applications often demand more in-depth and fine-grained analysis. Due to the maturity of the field, the book is also written in a structured form in the sense that the problem is now better defined and different research directions are unified around the definition.

Opinion Spam Detection

A key feature of social media is that it enables anyone from anywhere in the world to freely express his/her views and opinions without disclosing his/her true identify and without the fear of undesirable consequences. These opinions are thus highly valuable. However, this anonymity also comes with a price. It allows people with hidden agendas or malicious intentions to easily game the system to give people the impression that they are independent members of the public and post fake opinions to promote or to discredit target products, services, organizations, or individuals without disclosing their true intentions, or the person or

organization that they are secretly working for. Such individuals are called opinion spammers and their activities are called opinion spamming. Opinion spamming has become a major issue. Apart from individuals who give fake opinions in reviews and forum discussions, there are also commercial companies that are in the business of writing fake reviews and bogus blogs for their clients. Several high-profile cases of fake reviews have been reported in the news. It is important to detect such spamming activities to ensure that the opinions on the Web are a trusted source of valuable information. Unlike extraction of positive and negative opinions, opinion spam detection is not just a NLP problem as it involves the analysis of people's posting behaviors. It is thus also a data mining problem. Chapter 10 will discuss the current state-of-the-art detection techniques.



❄❄❄❄❄❄❄

10

SENTIMENT ANALYSIS AND OPINION MINING - II

Unit Structure

10.0 COMPUTING WITH AFFECTIVE LEXICONS

Computing with Affective Lexicons

Affective, Sentimental, and Connotative Meaning in the Lexicon

Why compute affective meaning?

Affective computing is the study and development of systems and devices that can **recognize, interpret, process, and simulate human affects.**

In general, affective computing refers to **a computer's capabilities to recognize a user's emotional states**, to express its own emotions, and to respond to the user's emotions (Picard, 1997).

Affective computing has potential applications in human–computer interaction, such as **affective mirrors allowing the user to see how he or she performs; emotion monitoring agents sending a warning before one sends an angry email; or even music players selecting tracks based on mood.**

Human-centered computation through affective computing can **help in recognizing emotions, and generate proper actions to have richer communication**. Applications of affective computing range from education , autonomous driving , entertainment , to health-care.

Affective Computing. The artificial intelligence (AI) field picked up the idea that human rational thinking depends on emotional processing. **Rosalind Picard's** "Affective Computing" had a major effect on both the AI and HCI fields (Picard, 1997)

What is an affective experience? As we use the term, an affective experience is **an experience that presents its objects in a positively or negatively valenced way**. To explain what we mean, we'll begin with some examples. Our first example is the feeling of desire.

What is affective AI?

Emotion AI, also known as affective computing, is essentially all about **detecting emotions using artificial intelligence**. Machines with this kind of emotional intelligence are able to understand not only the cognitive but also the emotive channels of human communication

What is meant by cognitive computing?

Cognitive computing is **the use of computerized models to simulate the human thought process in complex situations where the answers may be ambiguous and uncertain**. The phrase is closely associated with IBM's cognitive computer system, Watson.

What principles guide our social behavior?

Affect refers to the feelings that we experience as part of life and includes both moods and emotions. Social behavior is influenced by principles of **reciprocal altruism and social exchange**.

Computer vision algorithms identify key landmarks on the face – for example the corners of your eyebrows, the tip of your nose, the corners of your mouth. Machine learning algorithms then analyse pixels in those regions to classify facial expressions.

Narrow AI is goal-oriented, designed to perform singular tasks - i.e. **facial recognition, speech recognition/voice assistants, driving a car, or searching the internet** - and is very intelligent at completing the specific task it is programmed to do.

Major sub-fields of AI now include: **Machine Learning, Neural Networks, Evolutionary Computation, Vision, Robotics, Expert Systems, Speech Processing, Natural Language Processing, and Planning**.

Affective computing is becoming more and more important as **it enables to extend the possibilities of computing technologies by incorporating emotions**. In fact, the detection of users' emotions has become one of the most important aspects regarding Affective Computing.

Affective computing is becoming more and more important as **it enables to extend the possibilities of computing technologies by incorporating emotions**. In fact, the detection of users' emotions has become one of the most important aspects regarding Affective Computing.

Social Media are the platforms created using these Web 2.0 technologies. - **Social Computing is simply the branch of computer science that focuses on analyzing the relationship between human behavior and computer systems**.

How do you identify primary emotions?

Thomas says that primary emotions are simply **our initial reactions to external events or stimuli**. Secondary emotions are the reactions we then have to our reaction

Knowing the Difference Between Primary and Secondary Emotions Could Be the Key to Fighting Fairly with Your Partner

"It's just emotion that's taking me over," they weren't exaggerating—emotions affect us in powerful ways, impacting our behavior and our relationships. So, it's important we understand them as best we can. Which, if we're being honest, is easier said than done. In fact, civilizations have been trying to categorize human emotions for centuries. In today's

modern era, one of those categorizations focuses on primary and secondary emotions. Read on to learn more about these types of emotions—and how to best identify them.

What are emotions in the first place?

Happy, sad, mad, scared, friendly, thoughtful, lonely, cranky, grateful, delighted—the list goes on and on. How do you actually describe these invisible forces? Merriam-Webster defines emotions as "a conscious mental reaction (such as anger or fear) subjectively experienced as strong feeling usually directed toward a specific object and typically accompanied by physiological and behavioral changes in the body," meaning that these feelings aren't just fleeting, but they have real-life impacts on our actions a psychologist and emotional scientist who works with emotionally sensitive people, likes to think of emotions as energy in motion (e-*motion*—get it?).

So, what's the difference between primary and secondary emotions?

Think of it most this way: Primary comes first and secondary comes, well, second. From there, the psych field tends to differ on what *exactly* primary vs. secondary emotions are.. ***Psychology Today*****,** "In the 20th century, Paul Ekman identified six basic emotions (anger, disgust, fear, happiness, sadness, and surprise) and Robert Plutchik [identified] eight, which he grouped into four pairs of polar opposites (joy-sadness, anger-fear, trust-distrust, surprise-anticipation)." Some experts, explains Burton, believe these basic emotions are hard-wired, "innate and universal, automatic, and fast, and trigger behavior with a high survival value." Secondary emotions—like apprehension, interest, trust, for example—are hypothesized to come from a combination of the primary emotions

The key word there is *hypothesize*. Human emotions are so complicated that experts are continually studying, testing, disagreeing on and updating these theories.

Dr. Thomas, on another hand, sees emotions on a more individual level—instead of looking at a color-coded wheel of definite primary and secondary emotions, Dr. Thomas says that primary emotions are simply our initial reactions to external events or stimuli. Secondary emotions are the reactions we then have to our reactions. For example, If you're a sensitive person, this might ring especially true for you—most emotionally sensitive people tend to not only react to stimuli in their environments but also to that aforementioned reaction. This leads to a chain of reactions that can get more intense and last a longer time.

Are happiness, anger and sadness primary or secondary emotions?

That answer depends on which psychologist you talk to. If a professional follows the Plutchik Model, the basic, primary emotions are joy, trust, fear, surprise, sadness, anticipation, anger and disgust. Emotional scientist Dr. Thomas, however, doesn't believe emotions are experienced exclusively as primary or secondary. Says Thomas: "One can actually

experience a full range of emotions as a primary response to our environment." Secondary emotions, Dr. Thomas argues, are responses to our primary emotions. Think: Your submission to a literary journal is rejected. Your primary response is discouragement. Your secondary response might be anxiety. Why? You've become anxious in response to your own emotions.

Emotions are also arguably highly impacted by external factors, like culture. In clinical psychologist Harriet Lerner's *The Dance of Anger: A Woman's Guide To Changing The Patterns Of Intimate Relationships*, the author argues that the oh-so common anger women feel is not a primary emotion but an alarm bell signaling a deeper issue. Kathy Caprino interviewed Lerner in 2014 for *Forbes* asking, "Do you consider anger a negative emotion, especially in our culture of positivity?" Lerner responded: "Anger is neither positive nor negative. Anger simply is. It's an important emotion that deserves our attention and respect. But most of us have little experience using our anger as a vehicle for positive change. Instead we silence our anger or vent it in a way that leaves us feeling helpless and powerless."

How do primary vs. secondary emotions impact my relationships?

While we're not going to tell you the frustration you're feeling towards your partner who left the bathroom floor sopping wet is a primary or secondary emotion, it is clear that our emotions are complicated, and we should honor them by taking a step back to really consider what it is we're feeling. If we can get to the crux of what we're experiencing—and why we're experiencing it—we can make better decisions. It's important to be able to understand the reason why any feeling, from joy to deep sadness, arises.

Here's an example: Your wife broke your favorite dish and you're angry.

Are you angry that your wife would be so careless with something you love or are you embarrassed at how you yelled at her when she dropped the platter? This would feed into Dr. Thomas's ideas of emotions. Remember, Dr. Thomas says that when it comes to our relationships (including the one with yourself!), the most important thing is to be able to recognize whether you're reacting to your internal stimuli or the environment around you.

In the Plutchik Model, your anger surrounding the broken platter might be considered a secondary reaction to the initial, primal terror you felt when you heard it crash to the ground and instinctively thought, "danger!" But *Dance of Anger* author, Harriet Lerner, has another postulation: Your anger surrounding the broken dish might be trying to alert you to a more important story about your relationship.

If you're yelling at your wife about how you'll never be able to find a replacement, but really, you're hurt that your wife is often careless with things that are important to you, you won't get the resolution you

need. Think of the big waves of emotions you experience as opportunity for positive change—ride the waves safely to the shore instead of battling against the current. Take a breath (try some square breathing) and ask yourself, is this really a time to lambast your loved one or is this an occasion to access what's really going on with your relationship and it address it head on? If it's the latter, perhaps you calmly suggest, "I'm hurt you broke something important to me, but I need some time to process my thoughts. Let's enjoy dinner and make time to talk about this tomorrow."

OK, so how do I interpret my emotions?

There are lots of ways to gain clarity on your feelings. Here are some of our suggestions.

1. **Seek professional help**

   Whether it's a psychiatrist, psychologist, therapist, social worker or other mental health professional, often times gaining insight into your own emotions means bringing in some outside help. Seeing the big picture can be impossible when you're stuck inside the whirling chaos of it all. But as soon as you bring in a third, neutral party whose job is to help you understand yourself the truth can unravel fairly quickly—but only if you're open to it.

2. **Journaling**

   You'd be surprised at how effective putting pen to paper can be. Try spending ten minutes every day free writing. Jot down any thought that comes to you for that set period of time without worrying about making sense, and you'll be amazed at what comes to the surface. Who knew you were so moved by your roommate's cooking? Or so offended by the criticism your friend texted you? Not sure you can commit to journaling just yet? Try a one-word journal (consider it self-care for when you're feeling a little lazy.)

3. **Mindful Meditation**

   You've heard all the fuss about mindful meditation, but the hype is legit. The 2,000-year-old Buddhist practice has been a proven way to improve awareness around the present moment and self. While meditation can seem lofty and unfamiliar, there are so many ways to bring the ancient practice into your life. Start with a quick ten-minute session with an app like Headspace or Calm. Not into sitting still? Try mindful running!

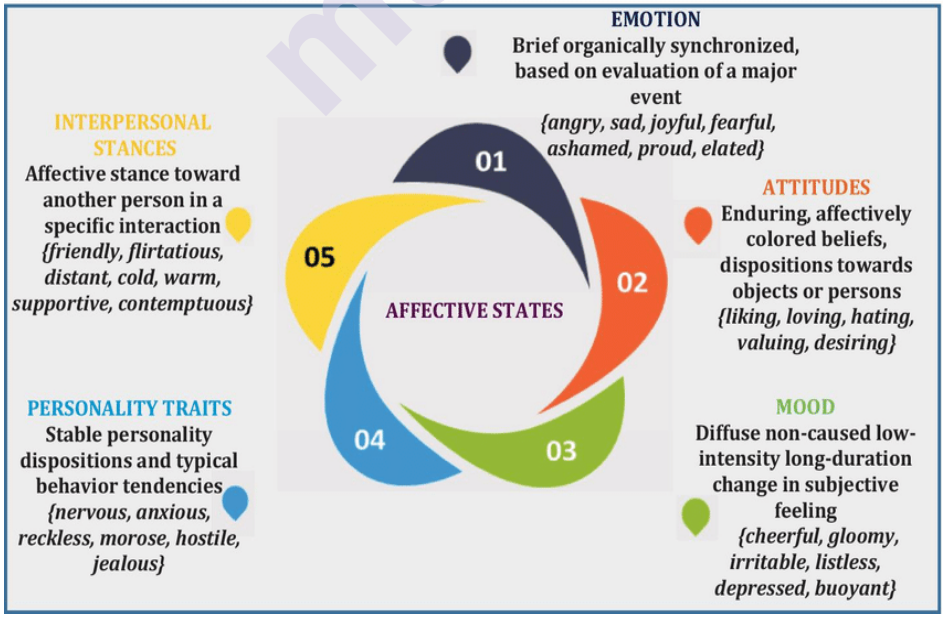
4. **Read**

   If one of the emotions you're feeling often is anger, get your hands on a copy of Harriet Lerner's *Dance of Anger* ASAP. If talking or thinking about your feelings is brand-new to you, try Marc Brackett's *Permission to Feel: Unlocking the Power of Emotions to Help Our Kids, Ourselves, and Our Society Thrive,* which provides a blueprint for acknowledging and skillfully understanding our emotions in a society that shames emotions.

Effective Computing:

- Detecting:
  - sentiment towards politicians, products, countries, ideas
  - frustration of callers to a help line
  - stress in drivers or pilots
  - depression and other medical conditions
  - confusion in students talking to e-tutors
  - emotions in novels (e.g., for studying groups that are feared over time)
- Could we generate:
  - emotions or moods for literacy tutors in the children's storybook domain
  - emotions or moods for computer games
  - personalities for dialogue systems to match the user

Scherer's typology of affective states



Emotion: relatively brief episode of synchronized response of all or most organismic subsystems in response to the evaluation of an event as being of major significance

angry, sad, joyful, fearful, ashamed, proud, desperate

Mood: diffuse affect state ...change in subjective feeling, of low intensity but relatively l duration, often without apparent cause

cheerful, gloomy, irritable, listless, depressed, buoyant

Interpersonal stance: affective stance taken toward another person in a specific interact coloring the interpersonal exchange

distant, cold, warm, supportive, contemptuous

Attitudes: relatively enduring, affectively colored beliefs, preferences predispositions towards objects or persons

liking, loving, hating, valuing, desiring

Personality traits: emotionally laden, stable personality dispositions and behavior tendencies, typical for a person

nervous, anxious, reckless, morose, hostile, envious, jealous

10.1 Aspect based sentiment analysis

Aspect-Based Sentiment Analysis (ABSA) is **a type of text analysis that categorizes opinions by aspect and identifies the sentiment related to each aspect**. By aspects, we consider attributes or components of an entity (a product or a service, in our case).

Traditional document-level sentiment classification tries to identify the general sentiment polarity of a given text as positive, negative, or neutral. Unlike document-level sentiment, aspect-level sentiment classification **identifies the sentiment of one specific aspect in its context sentence**.

What is aspect in NLP?

We determining the orientation of sentiment expressed on each aspect in a sentence. There are two main approaches, i.e., the supervised learning approach and the lexicon-based approach. For the supervised learning approach, the learning-based methods used for sentence-level and clause-level sentiment classification. In a hierarchical classification model was also proposed. However, the key issue is how to determine the scope of each sentiment expression, i.e., whether it covers the aspect of interest in the sentence. The current main approach is to use parsing to determine the dependency and the other relevant information. For example, a dependency parser was used to generate a set of aspect dependent features for classification. A related approach was also used which weights each feature based on the position of the feature relative to the target aspect in the parse tree. For comparative sentences, “than” or other related words can be used to segment a sentence. Supervised learning is dependent on the training data. As we discussed a model or classifier trained from labeled data in one domain often performs poorly in another domain.

Although domain adaptation (or transfer learning) has been studied by researchers, the technology is still far from mature, and the current methods are also mainly used for document level sentiment classification as documents are long and contain more features for classification than individual sentences or clauses. Thus, supervised learning has difficulty to scale up to a large number of application domains. Sentiment Analysis and Opinion Mining 60 The lexicon-based approach can avoid some of the issues , and has been shown to perform quite well in a large number of domains. Such methods are typically unsupervised. They use a sentiment lexicon (which contains a list of sentiment words, phrases, and idioms), composite expressions, rules of opinions , and (possibly) the sentence parse tree to determine the sentiment orientation on each aspect in a sentence. They also consider sentiment shifters, but-clauses and many other constructs which may affect sentiments.

Of course, the lexicon-based approach also has its own shortcomings, which we will discuss later. An extension of this method to handling comparative sentences . Below, we introduce one simple lexicon-based method to give a flavor of this approach. The method and it has four steps. Here, we assume that entities and aspects are known. Mark sentiment words and phrases: For each sentence that contains one or more aspects, this step marks all sentiment words and phrases in the sentence. Each positive word is assigned the sentiment score of +1 and each negative word is assigned the sentiment score of −1. For example, we have the sentence, "The voice quality of this phone is not good, but the battery life is long." After this step, the sentence becomes "The voice quality of this phone is not good [+1], but the battery life is long" because "good" is a positive sentiment word (the aspects in the sentence are italicized). Note that "long" here is not a sentiment word as it does not indicate a positive or negative sentiment by itself in general, but we can infer its sentiment in this context shortly. In fact, "long" can be regarded as a context-dependent sentiment word, which we will discuss in Chapter 10. In the next section, we will see some other expressions that can give or imply positive or negative sentiments.

2. **Apply sentiment shifters:** Sentiment shifters (also called valence shifters in (Polanyi and Zaenen, 2004)) are words and phrases that can change sentiment orientations. There are several types of such shifters. Negation words like not, never, none, nobody, nowhere, neither, and cannot are the most common type. This step turns our sentence into "The voice quality of this phone is not good[-1], but the battery life is long" due to the negation word "not." We will discuss several other types of sentiment shifters in the next section. Note that not every appearance of a sentiment shifter changes the sentiment orientation, e.g., "not only … but also." Such cases need to be dealt with care. That is, such special uses and patterns need to be identified beforehand.

3. **Handle but-clauses:** Words or phrases that indicate contrary need special handling because they often change sentiment orientations too. Sentiment Analysis and Opinion Mining 61 The most

commonly used contrary word in English is “but”. A sentence containing a contrary word or phrase is handled by applying the following rule: the sentiment orientations before the contrary word (e.g., but) and after the contrary word are opposite to each other if the opinion on one side cannot be determined. The if-condition in the rule is used because contrary words and phrases do not always indicate an opinion change, e.g., “Car-x is great, but Car-y is better.” After this step, the above sentence is turned into “The voice quality of this phone is not good[-1], but the battery life is long[+1]” due to “but” ([+1] is added at the end of the but-clause). Notice here, we can infer that “long” is positive for “battery life”. Apart from but, phrases such as “with the exception of,” “except that,” and “except for” also have the meaning of contrary and are handled in the same way. As in the case of negation, not every but means contrary, e.g., “not only … but also.” Such non-but phrases containing “but” also need to be identified beforehand.

4. **Aggregate opinions:** This step applies an opinion aggregation function to the resulting sentiment scores to determine the final orientation of the sentiment on each aspect in the sentence. Let the sentence be s, which contains a set of aspects {a1, …, am} and a set of sentiment words or phrases {sw1, …, swn} with their sentiment scores obtained from steps 1- 3. The sentiment orientation for each aspect ai in s is determined by the following aggregation function: , (,) . (,) ∑∈ = ow s j i j i j dist sw a sw so score a s (5) where swj is an sentiment word/phrase in s, dist(swj, ai) is the distance between aspect ai and sentiment word swj in s. swj.so is the sentiment score of swi. The multiplicative inverse is used to give lower weights to sentiment words that are far away from aspect ai. If the final score is positive, then the opinion on aspect ai in s is positive. If the final score is negative, then the sentiment on the aspect is negative. It is neutral otherwise. This simple algorithm performs quite well in many cases. It is able to handle the sentence “Apple is doing very well in this bad economy” with no problem. Note that there are many other opinion aggregation methods. For example, (Hu and Liu, 2004) simply summed up the sentiment scores of all sentiment words in a sentence or sentence segment. Kim, and Hovy (2004) used multiplication of sentiment scores of words. Similar methods were also employed by other researchers (Wan, 2008; Zhu et al., 2009). To make this method even more effective, we can determine the scope of each individual sentiment word instead of using words distance as above. In Sentiment Analysis and Opinion Mining 62 this case, parsing is needed to find the dependency as in the supervised method discussed above. We can also automatically discover the sentiment orientation of context dependent words such as “long” above. In fact, the above simple approach can be enhanced in many directions. For example, Blair-Goldensohn et al. (2008) integrated the lexicon-based method with supervised learning. Kessler and Nicolov (2009) experimented with four different strategies of determining the sentiment on each aspect/target

(including a ranking method). They also showed several interesting statistics on why it is so hard to link sentiment words to their targets based on a large amount of manually annotated data. Along with aspect sentiment classification research, researchers also studied the aspect sentiment rating prediction problem which has mostly been done together with aspect extraction in the context of topic modeling, As indicated above, apart from sentiment words and phrases, there are many other types of expressions that can convey or imply sentiments. Most of them are also harder to handle. Below, we list some of them, which are called the basic rules of opinions .

Aspect-based sentiment analysis goes one step further than sentiment analysis by automatically assigning sentiments to specific features or topics. It involves **breaking down text data into smaller fragments**, allowing you to obtain more granular and accurate insights from your data.

Aspect-Based Opinion Mining (ABOM) **involves extracting aspects or features of an entity and figuring out opinions about those aspects**. It's a method of text classification that has evolved from sentiment analysis and named entity extraction (NER). ABOM is thus a combination of aspect extraction and opinion mining.

Which type of analytics is used in sentiment analysis?

Sentiment analysis is used to determine whether a given text contains negative, positive, or neutral emotions. It's a form of text analytics that uses **natural language processing (NLP) and machine learning**. Sentiment analysis is also known as “opinion mining” or “emotion artificial intelligence.

Issues and Challenges of Aspect-based Sentiment Analysis: A Comprehensive Survey Ambreen Nazir, Yuan Rao, Lianwei Wu, Ling Sun Abstract—The domain of Aspect-based Sentiment Analysis, in which aspects are extracted, their sentiments are analyzed and sentiments are evolved over time, is getting much attention with increasing feedback of public and customers on social media. The immense advancements in the field urged researchers to devise new techniques and approaches, each sermonizing a different research analysis/question, that cope with upcoming issues and complex scenarios of Aspect-based Sentiment Analysis. Therefore, this survey emphasized on the issues and challenges that are related to extraction of different aspects and their relevant sentiments, relational mapping between aspects, interactions, dependencies and contextual-semantic relationships between different data objects for improved sentiment accuracy, and prediction of sentiment evolution dynamicity. A rigorous overview of the recent progress is summarized based on whether they contributed towards highlighting and mitigating the issue of Aspect Extraction, Aspect Sentiment Analysis or Sentiment Evolution.

The reported performance for each scrutinized study of Aspect Extraction and Aspect Sentiment Analysis is also given, showing the quantitative evaluation of the proposed approach. Future research directions are

proposed and discussed, by critically analysing the presented recent solutions, that will be helpful for researchers and beneficial for improving sentiment classification at aspect-level.

Index Terms— Aspect, Computational linguistic, Deep learning, Sentiment analysis, Sentiment evolution, Social media —

INTRODUCTION HE field of Natural Language Processing (NLP) dealing Sentiment Analysis (SA), also named as opinion mining, is an active research area to display emotions and to automatically discover the sentiments expressed within the text.

The object of SA is usually a product or service that is of keen interest among people, that they care to put a sentiment towards it. Traditionally, SA has been considered as opinion polarity that whether someone has expressed positive, negative or neutral sentiment about an event .

Since last decade, researchers are putting efforts to capture, quantify and measure dynamic public sentiments through different methods, tools and techniques, and thus allowing SA as one of the rapidly growing research areas.

SA applications have been widely spread to nearly every domain, like social events, consumer products and services, healthcare, political elections and financial services.

Influential groups and business organizations, like, Google, Microsoft, SAP and SAS, have designed their own in-house capabilities that support them in decision making and assist them in developing better business applications to track and predict evolving market trends. From studies, SA has been generally categorized at three levels.

Document-level
sentence-level
and aspect level

To classify whether a whole document, sentence (subjective or objective) and an aspect expresses a sentiment, i.e., positive, negative or neutral. The Aspect based Sentiment Analysis (AbSA) helps to understand the problem of SA better comparatively, because it directly focuses on sentiments rather than language structure. Where, an aspect is related to an entity, and the basic concept of an aspect is not just limited to judgement but also extends towards thoughts, point of views, ways of thinking, perspectives, an underlying theme or social influence towards an occurrence.

Hence, AbSA provides a great opportunity to analyse sentiments (public) over time across different contents present on media .

AbSA can be categorized by three main processing phases, i.e., Aspect Extraction (AE), Aspect Sentiment Analysis (ASA) and Sentiment Evolution (SE). The first phase deals with the extraction of aspects, which can be

explicit aspects ,
implicit aspects ,
aspect terms ,
entities and Opinion Target Expressions (OTE) .

The second phase classifies sentiment polarity for a predefined aspect, target or entity .

This phase also formulates interactions, dependencies and contextual semantic relationships between different data objects, e.g., aspect, entity, target, multi-word target, sentiment word, for achieving improved sentiment classification accuracy .

The expressed sentiment can be classified into ternary, or fine-gained sentiment values .

The third phase concerns with the dynamicity of peoples' sentiment towards aspects (events) over a period of time. Social characteristics and self-experience are considered as the leading causes for SE .

Focus of this Survey The field of AbSA is not a straight road, it has suffered many diverse changes and touched many new eras to ponder over. Researchers have been working hard to resolve multi-faceted challenges containing many issues. They have come up with thorough solutions of many complicated challenges through different machine learning techniques, mostly deep-learning techniques, that represent their critical idea in the field. They have provided pictorial representations and numerical modeling for handling complex scenarios.

Organization of this Survey :The survey has been started with the brief introduction and paramount significance of AbSA. The rest of the survey is organized as follows: Section II provides the definitions of sentiment with respect to aspect, and lists down the major issues and challenges related to AE, ASA and SE. Section III and IV discusses the major issues of AE and ASA, and concisely describes the recent solutions for these issues. Section V discusses the dynamics of SE. Section VI highlights the future research directions. Section VII concludes this survey.

KEY SOLUTIONS : This section presents AbSA definitions, and outlines the major issues and sub-issues of AE, ASA and SE. This section also illustrates three main processing steps of AbSA, i.e., AE, ASA and SE, through framework for the ease and understandability of the reader. Definitions (AbSA) In AbSA, sentiment is a subjective consciousness of human beings towards an aspect (objective existence). People get affected by the ups and downs of life, at any time and place, which results in the change of their sentiment towards a specific aspect. This change depicts human behavior flexibility, decision autonomy and creative thinking. Pang and Lee defined SA as: "A sentiment is basically an opinion that a person expresses towards an aspect, entity, person, event, feature, object, or a certain target." Now Title Critical Idea Challenges Pang . Opinion Mining and SA · presented applications that determine sentiment of the document, and organized approaches related to opinion-oriented classification

problems How to perform SA at document-level using machine-learning approaches?

A Survey on Sentiment Detection of Reviews · discussed approaches related to subjectivity classification, word classification and opinion discovery in customer review domain How to perform SA at document-level using machine-learning approaches?

A Survey on the Role of Negation in SA · presented computational methods for handling negation in SA How to cope with the scope and challenges of negation modeling?

Survey on Mining Subjective Data on The Web · differentiated between four different approaches that classify word-sentiment value, i.e., machine-learning, semantic, statistical and dictionary-based approaches How to perform subjective SA at documentlevel?

SA and Opinion Mining · covered the field of SA at document, sentence and aspect-level · discussed various issues related to AE, sentiment classification, sentiment lexicons, NLP and opinion-spam detection · surveyed the till date practical solutions along with the future directions How to cope with review ranking, redundancy issues, viewpoints quality, genuine aspects, spammer detection etc… ?

A Survey on Opinion Mining and SA: Tasks, Approaches and Applications · organized subtasks of machine learning, NLP and SA techniques, such as, subjectivity classification, sentiment classification, lexicon relation, opinion-word extraction, and various applications of SA · discussed open issues and future directions in SA How to focus on sentence-level and document-level SA and their subtasks?

Like It or Not: A Survey of Twitter SA Methods · discussed the deep-learning algorithms related to twitter SA · elaborated tasks specific to emotion detection, change of sentiment over time, sarcasm detection, and sentiment classification How to tackle the challenges, tasks and feature selection methods limited to twitter SA?

Survey on Aspect-Level SA · performed approach-based categorization of different solutions those were related to AE, aspect classification and a combination of both · proposed future research direction for semantically-rich-concept-centric AbSA How to cope with the challenges of comparative opinions, conditional sentences, negation modifiers and presentation?

Deep Learning for SA: A Survey · presented applications and deep-learning approaches for the SA related tasks such as sentiment intersubjectivity, lexicon expansion, stance detection How to achieve advances in SA using deep learning approaches?

The State-of-the-Art in Twitter SA: A Review and Benchmark Evaluation · focused on challenges and key trends related to classification errors, twitter monitoring and event detection to perform twitter SA

effectively How to reveal the root causes of commonly occurring classification errors?

A Survey of SA in Social Media · categorized the latest technologies and techniques in SA · introduced latest tools for research approaches related to subjectivity classification in customer-review domain How to focus on different types of data and advance tools, to overcome the limitations of social media SA?

A Survey on Sentiment and Emotion Analysis for Computational Literary Studies · presented approaches of SA and emotion analysis · discussed the computational methods for sentiment and emotion classification How to classify and interpret the emotional text through sentiment and emotional analysis in digital human community? Our Issues and Challenges of Aspect-based Sentiment Analysis: A Comprehensive Survey · discusses the issues and challenges of AE, ASA and SE · presents the progress of AbSA by concisely describing the recent solutions · highlight factors responsible for SE dynamicity · proposes future research directions by critically analyzing the present solutions How to improve the mechanism of AE? What measures should be taken to achieve good classification accuracy at aspect-level? How to predict SE dynamicity?

Reference pdf:

- https://www.cs.uic.edu/~liub/FBS/SentimentAnalysis-and-OpinionMining.pdf
- https://www.pdfdrive.com/sentiment-analysis-and-opinion-mining-e1590362.html
- https://www.researchgate.net/publication/265163299_Sentiment_Analysis_and_Opinion_Mining_A_Survey
- https://link.springer.com/referenceworkentry/10.1007/978-1-4899-7687-1_907
- https://www.morganclaypool.com/doi/abs/10.2200/S00416ED1V01Y201204HLT016

❄❄❄❄❄❄❄