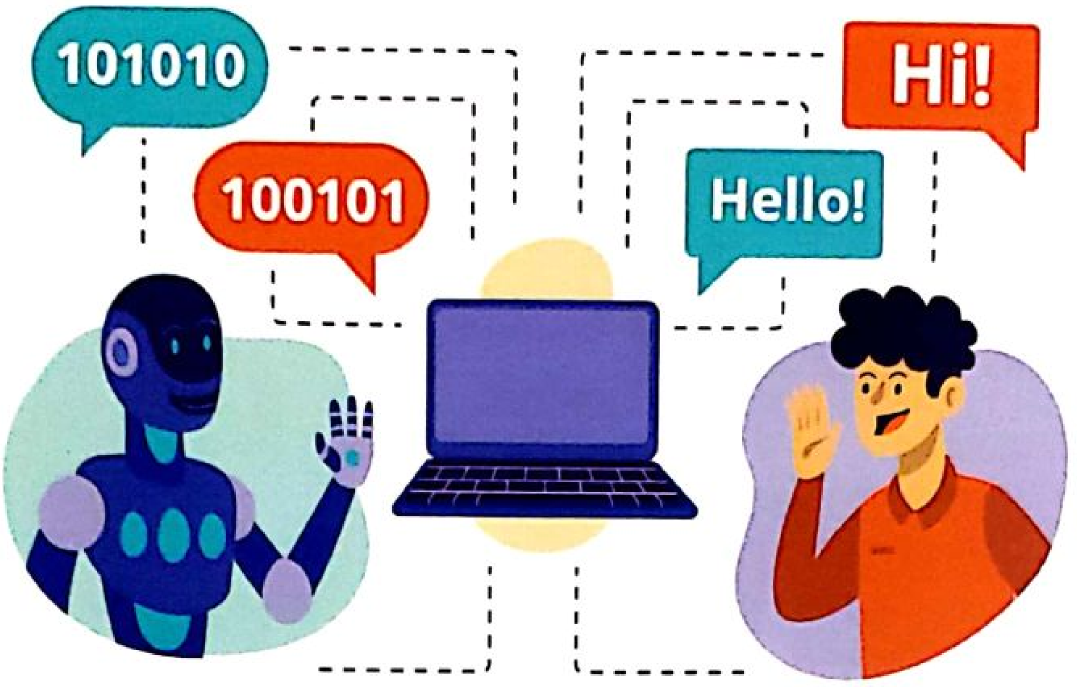


Sub. Code : 3170723

Natural Language Processing

AS PER NEW SYLLABUS - GTU - SEM VII (CE/CSE) Professional Elective - VI

- Simplified & Conceptual Approach
- Solved GTU Question Paper Winter 2021

first edition : july 2022

Price : ₹ 185/-
ISBN 978-93-90770-28-1

TECHNICAL® PUBLICATIONS®
SINCE 1993
An Up-Thrust for Knowledge

Pranjali Deshpande
Soudamini Patil

SYLLABUS

Natural Language Processing - (3170723)

Credits	Examination Marks				Total Marks
C	Theory Marks		Practical Marks		
	ESE (E)	PA (M)	ESE (V)	PA (I)	
4	70	30	30	20	150

1. **Introduction to NLP :** What is NLP ? Why NLP is Difficult ? History of NLP, Advantages of NLP, Disadvantages of NLP, Components of NLP, Applications of NLP, How to build an NLP pipeline ? Phases of NLP, NLP APIs, NLP Libraries. **(Chapter - 1)**

2. **Language Modeling and Part of Speech Tagging :** Unigram Language Model, Bigram, Trigram, N-gram, Advanced smoothing for language modeling, Empirical Comparison of Smoothing Techniques, Applications of Language Modeling, Natural Language Generation, Parts of Speech Tagging, Morphology, Named Entity Recognition. **(Chapter - 2)**

3. **Words and Word Forms :** Bag of words, skip-gram, Continuous Bag-Of-Words, Embedding representations for words Lexical Semantics, Word Sense Disambiguation, Knowledge Based and Supervised Word Sense Disambiguation. **(Chapter - 3)**

4. **Text Analysis, Summarization and Extraction :** Sentiment Mining, Text Classification, Text Summarization, Information Extraction, Named Entity Recognition, Relation Extraction, Question Answering in Multilingual Setting; NLP in Information Retrieval, Cross-Lingual IR. **(Chapter - 4)**

5. **Machine Translation :** Need of MT, Problems of Machine Translation, MT Approaches, Direct Machine Translations, Rule-Based Machine Translation, Knowledge Based MT System, Statistical Machine Translation (SMT), Parameter learning in SMT (IBM models) using EM, Encoder-decoder architecture, Neural Machine Translation. **(Chapter - 5)**

TABLE OF CONTENTS

Chapter 1

Introduction to NLP

Syllabus

What is NLP? Why NLP is Difficult? History of NLP, Advantages of NLP, Disadvantages of NLP, Components of NLP, Applications of NLP, How to build an NLP pipeline? Phases of NLP, NLP APIs, NLP Libraries.

Contents

1.1 What is NLP ?

What is NLP

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

1.2 Why NLP is Difficult ?

NLP is different

- What distinguishes NLP applications from other data processing systems is their use of knowledge of language.
- Below are the main differences between Natural Language and Computer Language :

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

- NLP tasks require knowledge about written human language i.e. word sense, grammar, disambiguation. It should understand variations of individual words (for example recognizing that doors is plural) requires knowledge about morphology, which captures information about the shape and behaviour of words in context.
- The knowledge needed to order and group words together comes under the heading of syntax.
- NLP requires knowledge of the meanings of the component words i.e. the domain of lexical semantics.
- The knowledge of how these components combine to form larger meanings, is referred as compositional semantics.
- The appropriate use of implied, indicative and indirect language comes under the heading of pragmatics.
- In addition to written language processing, spoken language processing tasks require additional computational knowledge about phonetics and phonology.
- Correctly structuring the words in conversations requires knowledge of discourse conventions.

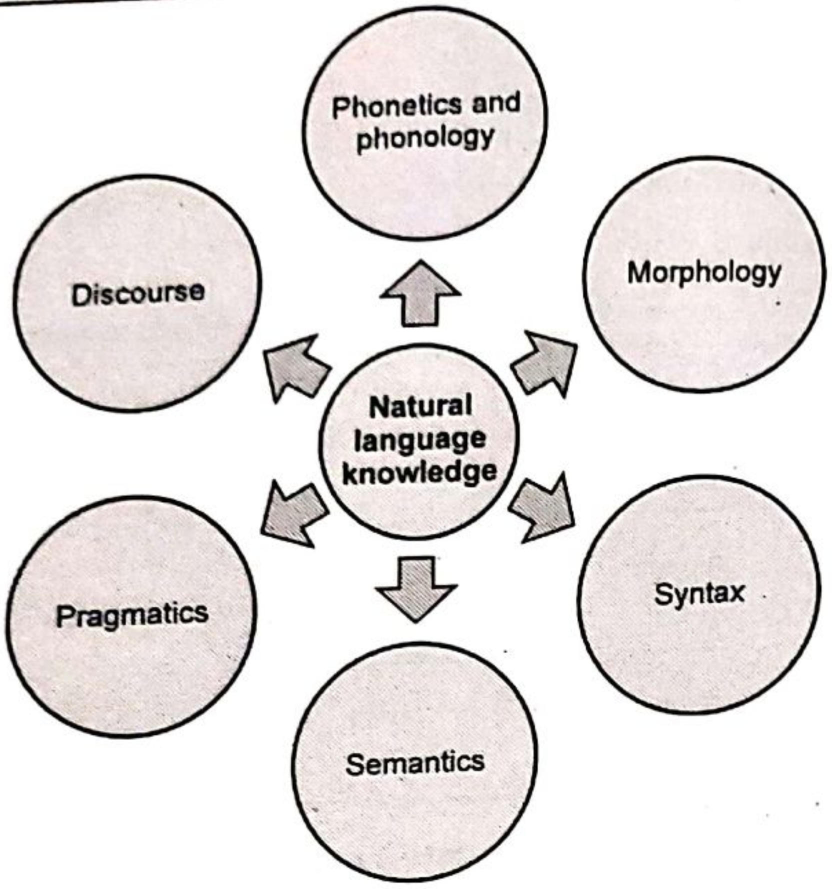


Fig. 1.2.1 : Categories of language knowledge

Challenges in natural language processing

- These challenges in NLP frequently involve speech recognition, natural language understanding, and natural - language generation.
- Natural Language Processing is considered a difficult problem in computer science. It is the nature of the human language that makes NLP difficult. Human brains can easily master a language, the ambiguity and imprecise characteristics of the natural languages. These aspects of natural language make NLP difficult for computers to implement
- The central challenge of Natural Language Processing is ambiguity. It exists at every level or stage of NLP.
- For NLP, we say some input is ambiguous when there are multiple alternative linguistic structures than can be built for it. Here is an example.

- Here is an example that gives different meanings highlighting ambiguity at some level :

Call me a jack please

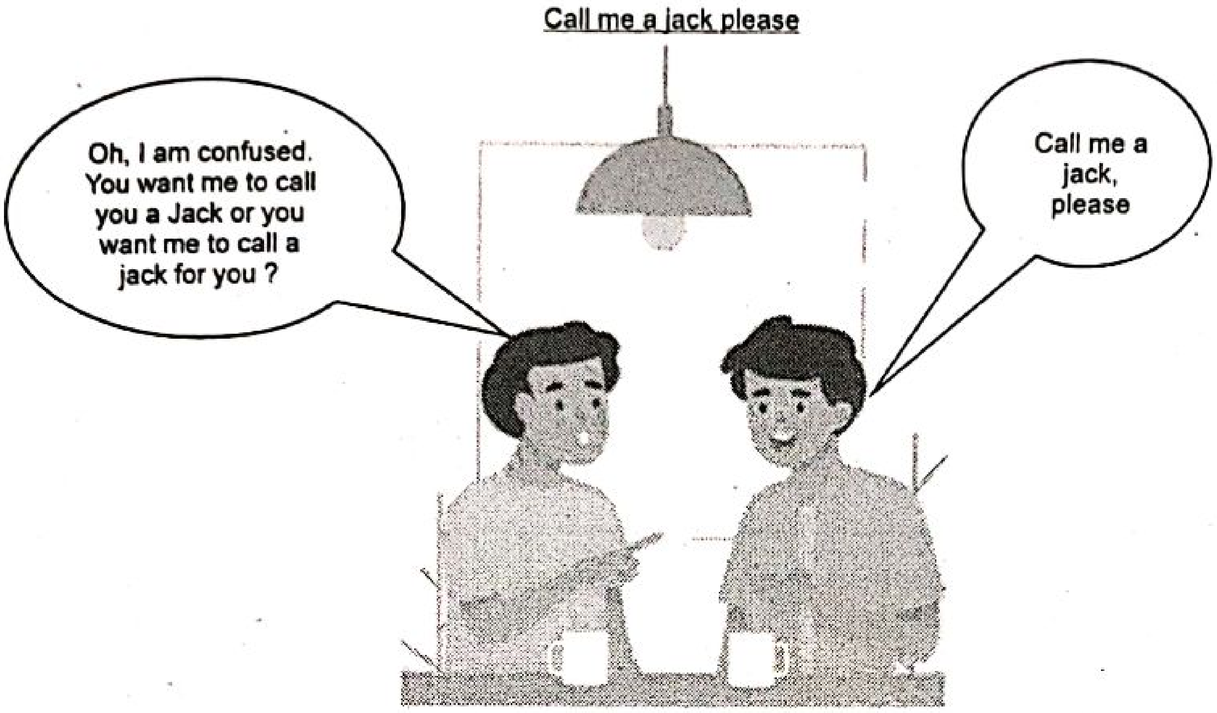

Call me a jack please
Oh, I am confused. You want me to call you a Jack or you want me to call a jack for you ?
Call me a jack, please

Fig 1.2.2 : Example of ambiguity in natural language sentence(s)

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

- Ambiguities that may arise in this example (like whether a given sentence is a request or an order or a plain sentence) will can be resolved, for example by speech act interpretation.

Difficulties in NLP

- Natural languages are very ambiguous and many a times, the meaning of words change with reference to the context and domain in which it is used. Hence natural language words have different levels of ambiguity like :
  - **Lexical ambiguity :** It is at very primitive level such as word-level. For example, treating the word "board" as noun or verb ?
  - **Syntax level ambiguity :** A sentence can be parsed in different ways. For example, "He lifted the beetle with red cap." – Did he use cap to lift the beetle or he lifted a beetle that had red cap ?
  - **Referential ambiguity :** Referring to something using pronouns. For example, Tom went to John. He said, "I am tired." – Exactly who is tired ? Here, one input can mean different meanings and sometimes many inputs can mean the same thing.

1.3 History of NLP

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

- In 1957, Noam Chomsky's syntactic structures revolutionized linguistics with 'universal grammar', a rule based system of syntactic structures.
- The Georgetown experiment in 1954 involved fully automatic translation of more than sixty Russian sentences into English. The authors claimed that within three or five years, machine translation would be a solved problem.
- In 1969 Roger Schank introduced the conceptual dependency theory for natural language understanding. This model, partially influenced by the work of Sydney Lamb, was extensively used by Schank's students at Yale University, such as Robert Wilensky, Wendy Lehnert, and Janet Kolodner.
- In 1970, William A. Woods introduced the augmented transition network (ATN) to represent natural language input. Instead of phrase structure rules ATNs used an equivalent set of finite state automata that were called recursively.
- Up to the 1980s, most natural language processing systems were based on complex sets of hand - written rules. Starting in the late 1980s, however, there was a revolution in natural

language processing with the introduction of machine learning algorithms for language processing.

- 1990s : Many of the notable early successes on statistical methods in NLP occurred in the field of machine translation, due especially to work at IBM Research. These systems were able to take advantage of existing multilingual textual corpora
- 2000s : With the growth of the web, increasing amounts of raw (unannotated) language data has become available since the mid - 1990s. Research has thus increasingly focused on unsupervised and semi - supervised learning algorithms.
- In the 2010s, representation learning and deep neural network - style machine learning methods became widespread in natural language processing, due in part to a flurry of results showing that such techniques can achieve state - of - the - art results in many natural language tasks, for example in language modelling, parsing and many others.

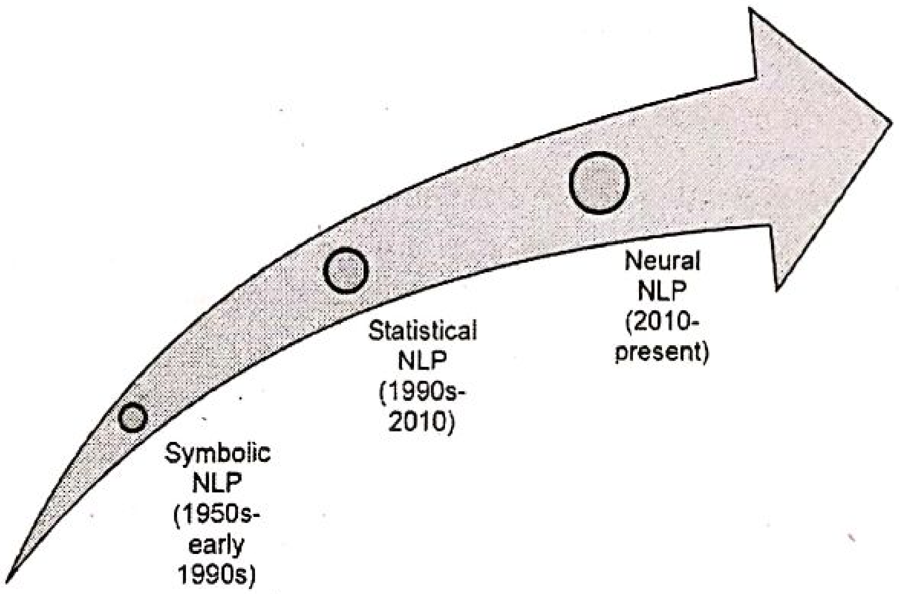

Symbolic NLP (1950s- early 1990s)
Statistical NLP (1990s- 2010)
Neural NLP (2010- present)

Fig. 1.3.1 : Symbolic representation of NLP history

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

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

1.4 Advantages of NLP

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

- NLP process help computer communicate with a human in their language and scales other language - related tasks.
- It is easy to implement in limited scope.
- It is very time efficient.
- NLP helps users to ask questions about any subject and get a direct response within seconds.
- NLP system provides answers to the questions in natural language.
- NLP improves the efficiency of documentation processes, accuracy of documentation, and identify the information from large databases.
- Using NLP has advantages (less costly than employing human staff, provides quicker customer service response times and is easy to implement)
- NLP system offers exact answers to the questions, no unnecessary or unwanted some information.
- The accuracy of the answer increases with the amount of relevant information provided in the questions.

- Structuring a high unstructured data source.
- Users can ask questions about any subject and get a direct response in seconds.
- Using an NLP system is less costly than hiring a person. A person can take two or three times longer than a machine to execute the tasks mentioned.
- NLP allows to perform more language - based data compares to a human being without fatigue and in an unbiased and consistent way.
- It is a faster customer service response time.

1.5 Disadvantages of NLP

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

1.6 Components of NLP

For natural language communication to take place, following two things are necessary.

- Language understanding
- Language generation (response based on what is understood)

When humans talk to each other, the first thing that other humans do is understand the context. Later formulate the response accordingly that makes sense. This is what the two terms try to say with Natural language understanding, it means to understand the context, and Natural language generation relates to sensible response to the context. Natural language processing also addresses both these components.

Natural Language Understanding (NLU)

- Natural Language Understanding (NLU) helps the machine to understand and analyse human language.
- NLU used in business applications to understand the customer's problem in both spoken and written language.
- NLU involves the two tasks -
  - Used to map the given input into useful representation.
  - Used to analyse different aspects of the language.

Natural Language Generation (NLG)

- Natural Language Generation (NLG) acts as a translator that converts the computerized data into natural language representation.
- It involves in
  - Text planning,
  - Sentence planning
  - Text realization

Difference between NLU and NLG

Natural Language Understanding (NLU)	Natural Language Generation (NLG)
NLU is the process of reading and interpreting language.	NLG is the process of writing or generating language.
It produces non-linguistic outputs from natural language inputs.	It produces constructing natural language outputs from non-linguistic inputs.

- NLP Applications like machine translation, summarization, question-answering uses both the components of understanding and generation.

1.7 Applications of NLP

- NLP applications include word counting and automatic hyphenation, to cutting edge applications such as automated question answering on the Web, and real-time spoken language translation and many more.

- NLP is a widely used technology today. Here, are common Natural Language Processing applications.
  1. **Information retrieval and web search :** Google, Yahoo, Bing, and other search engines base their machine translation technology on NLP deep learning models. It allows algorithms to read text on a webpage, interpret its meaning and translate it to another language.
  2. **Grammar correction :** NLP technique is widely used by word processor software like MS-word for spelling correction and grammar check.
  3. **Question answering :** Type in keywords to ask questions in Natural Language.
  4. **Text summarization :** The process of summarising important information from a source to produce a shortened version.
  5. **Machine translation :** Use of computer applications to translate text or speech from one natural language to another.
  6. **Sentiment analysis :** NLP helps companies to analyse a large number of reviews on a product. It also allows their customers to give a review of the particular product.
- NLP uses language knowledge that can be separated into six distinct categories (Refer Fig 1.2.1)
  1. **Phonetics and phonology :** The study of linguistic sounds.
  2. **Morphology :** The study of the meaningful components of words.
  3. **Syntax :** The study of the structural relationships between words.
  4. **Semantics :** The study of meaning.
  5. **Pragmatics :** The study of how language is used to accomplish goals.
  6. **Discourse :** The study of linguistic units larger than a single utterance. statement
- NLP covers a wide range of research areas, including parsing, semantics interpretation, and pragmatics. The tasks related to each of these areas are shown in Fig. 1.2.2.

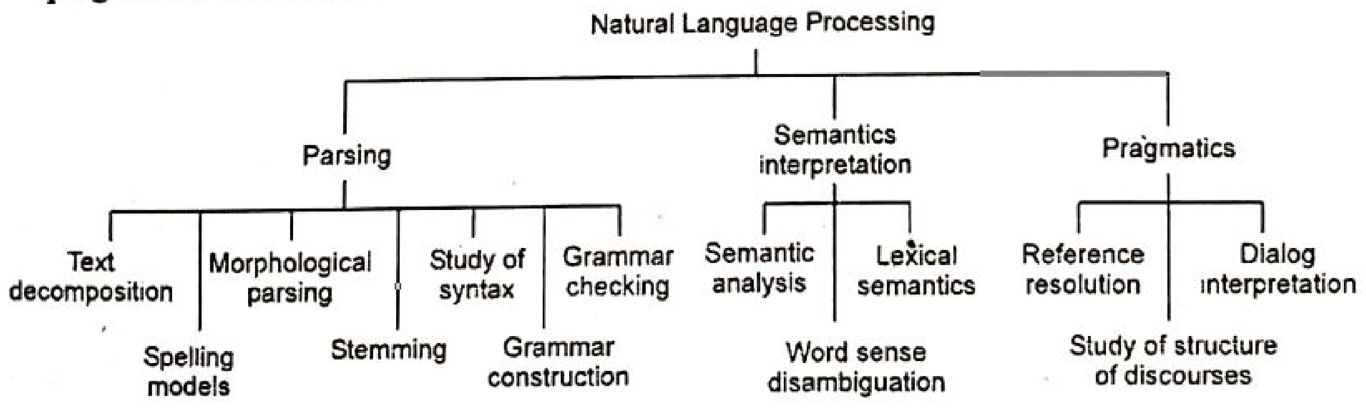


Fig. 1.7.1 : Research areas in NLP

Some of the NLP based systems that one can see around are :

1. Human voice response systems like - Alexa, Siri
2. Spelling correctors like - MS Word, Automated Dictionaries
3. Language translators like - Google Translate, Translate option in browsers
4. Text response systems like - Chatbots
5. Word (Entity) recognition like - spam detection in emails
6. Opinion detection / sentiment analysis like - Twitter sentiment analyser

1.8 How to Build an NLP Pipeline ?

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

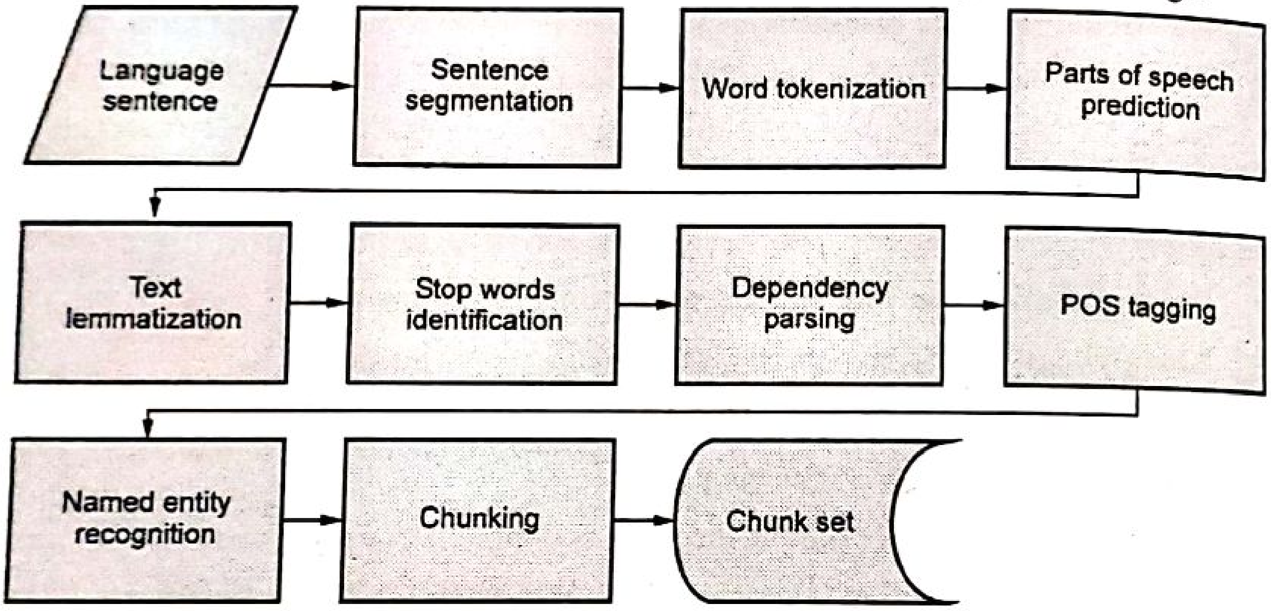


Fig. 1.8.1 : NLP pipeline and stages

Sentence segmentation

- When a language paragraph is to be processed, the best way to proceed is to go with one sentence at a time. It reduces the complexity and simplifies the process, even gets you the most accurate results. Computers never understand language the way humans do, but they can always do a lot if you approach them in the right way.
- For example, consider the above paragraph. Then, your next step would be breaking the paragraph into single sentences.

Word Tokenization

- Tokenization is the process of breaking a phrase, sentence, paragraph, or entire documents into the smallest unit, such as individual words or terms. And each of these small units is known as tokens.

- These tokens could be words, numbers, or punctuation marks. Based on the word's boundary - ending point of the word. Or the beginning of the next word. It is also the first step for stemming and lemmatization.
- This process is crucial because the meaning of the word gets easily interpreted through analysing the words present in the text.
- Let's take an example :

  That dog is a husky breed.

  When you tokenize the whole sentence, the answer you get is ['That', 'dog', 'is', a, 'husky', 'breed'].
- There are numerous ways you can do this, but we can use this tokenized form to :
  - Count the number of words in the sentence.
  - Measure the frequency of the repeated words.

Parts of Speech (PoS) Prediction

- In a part of the speech, we have to consider each token. And then, try to figure out different parts of the speech - whether the tokens belong to nouns, pronouns, verbs, adjectives, and so on. All these help to know which sentence we all are talking about.
- Here are some keywords used in PoS
  1. **Corpus :** Body of text, singular. Corpora are the plural of this.
  2. **Lexicon :** Words and their meanings.
  3. **Token :** Each "entity" that is a part of whatever was split up based on rules.

Text lemmatization

- English is also one of the languages where we can use various forms of base words. When working on the computer, it can understand that these words are used for the same concepts when there are multiple words in the sentences having the same base words. The process is what we call lemmatization in NLP.
- It goes to the root level to find out the base form of all the available words. They have ordinary rules to handle the words and most of us are unaware of them.

Identifying stop words

- When you finish the lemmatization, the next step is to identify each word in the sentence. English has a lot of filler words that do not add any meaning but weakens the sentence. It is always better to omit them because they appear more frequently in the sentence.
- Most data scientists remove these words before running into further analysis. The basic algorithms to identify the stop words by checking a list of known stop words as there is no standard rule for stop words.

Dependency Parsing

- Parsing is divided into three prime categories further. And each class is different from the others. They are part of speech tagging, dependency parsing, and constituency phrasing.
- The dependency phrasing case : Analyses the grammatical structure of the sentence. Based on the dependencies in the words of the sentences. Whereas in constituency parsing : the sentence breakdown into sub - phrases. And these belong to a specific category like Noun Phrase (NP) and Verb Phrase (VP).

POS tags

- POS stands for Parts Of Speech, which includes noun, verb, adverb and adjective. It indicates that how a word functions with its meaning as well as grammatically within the sentences.
- A word has one or more parts of speech based on the context in which it is used.

Named Entity Recognition (NER)

- It is a process of detecting the named entity such as person name, movie name, organization name, or location.

Chunking

- Chunking is used to collect the individual piece of information and grouping them into bigger pieces of sentences.

1.9 Phases of NLP

connect closely

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

Phases of NLP

1. **Lexical analysis and morphological analysis :** The first phase of NLP is the lexical Analysis. This phase scans the source code as a stream of characters and converts it into meaningful lexemes. It divides the whole text into paragraphs, sentences and words.
2. **Syntactic analysis (Parsing) :** Syntactic Analysis is used to check grammar, word arrangements and shows the relationship among the words.

   Example : Dog eats bread

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

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

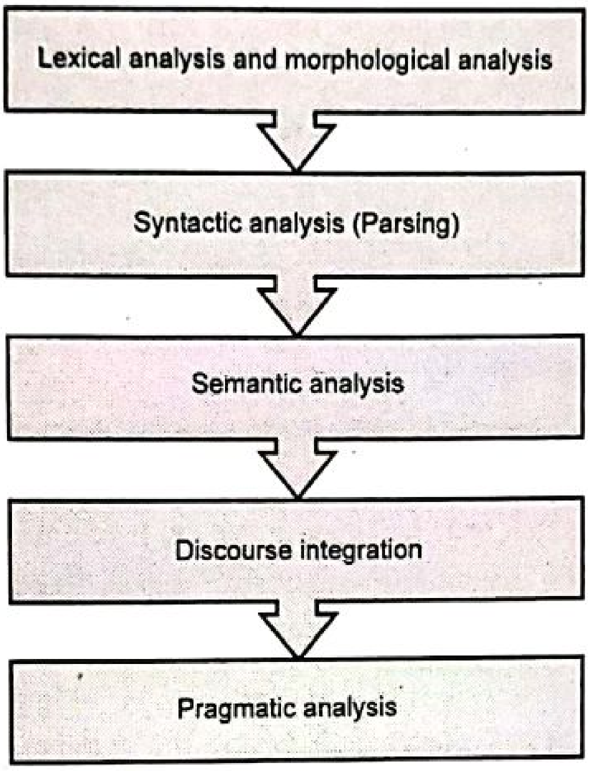

Lexical analysis and morphological analysis
Syntactic analysis (Parsing)
Semantic analysis
Discourse integration
Pragmatic analysis

Fig. 1.9.1 : Phases of NLP

4. **Discourse integration :** Discourse integration depends upon the sentences that proceeds it and also invokes the meaning of the sentences that follow it.
5. **Pragmatic analysis :** Pragmatic is the fifth and last phase of NLP. It helps you to discover the intended effect by applying a set of rules that characterize cooperative dialogues.

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

1.10 NLP APIs

Natural Language Processing APIs allow developers to integrate human - to - machine communications and complete several useful tasks such as speech recognition, chatbots, spelling correction, sentiment analysis, etc.

A list of NLP APIs is given below :

IBM Watson API

IBM Watson API combines different sophisticated machine learning techniques to enable developers to classify text into various custom categories. It supports multiple languages, such as English, French, Spanish, German, Chinese, etc. With the help of IBM Watson API, you can extract insights from texts, add automation in workflows, enhance search and understand the sentiment. The main advantage of this API is that it is very easy to use.

Chatbot API

Chatbot API allows you to create intelligent chatbots for any service. It supports unicode characters, classifies text, multiple languages, etc. It is very easy to use. It helps you to create a chatbot for your web applications.

Cloud NLP API

The Cloud NLP API is used to improve the capabilities of the application using natural language processing technology. It allows you to carry various natural language processing functions like sentiment analysis and language detection. It is easy to use.

Google Cloud Natural Language API

Google Cloud Natural Language API allows you to extract beneficial insights from unstructured text. This API allows you to perform entity recognition, sentiment analysis, content classification, and syntax analysis in more the 700 predefined categories. It also allows you to perform text analysis in multiple languages such as English, French, Chinese, and German.

Speech to text API

Speech to text API is used to convert speech to text.

Sentiment Analysis API

Sentiment Analysis API is also called as 'opinion mining' which is used to identify the tone of a user (positive, negative, or neutral).

Translation API by SYSTRAN

The translation API by SYSTRAN is used to translate the text from the source language to the target language. You can use its NLP APIs for language detection, text segmentation, named entity recognition, tokenization and many other tasks.

Text analysis API by AYLIEN

Text analysis API by AYLIEN is used to derive meaning and insights from the textual content. It is available for both free as well as paid from $119 per month. It is easy to use.

1.11 NLP Libraries

Here is a list of NLP libraries that can be readily used while coding an NLP application. There are numerous NLP libraries designed for specific NLP applications. However Python is the lead programming language that supports most of the NLP libraries. Here is a list of few popular NLP libraries.

Core NLP :

- Stanford CoreNLP comprises of an assortment of human language technology tools. It aims to make the application of linguistic analysis tools to a piece of text easy and efficient. With CoreNLP, you can extract all kinds of text properties (like named-entity recognition, part - of - speech tagging, etc.) in only a few lines of code.
- The tool incorporates numerous Stanford's NLP tools like the parser, sentiment analysis, bootstrapped pattern learning, Part - Of - Speech (POS) tagger, Named Entity Recognizer (NER), and coreference resolution system, to name a few. Furthermore, CoreNLP supports four languages apart from English - Arabic, Chinese, German, French and Spanish.

Scikit - learn :

It provides a wide range of algorithms for building machine learning models in Python.

Natural Language Tool Kit (NLTK) :

NLTK is a complete toolkit for all NLP techniques. NLTK comes with a host of text processing libraries for sentence detection, tokenization, lemmatization, stemming, parsing, chunking, and POS tagging.

Pattern :

Pattern is a text processing, web mining, natural language processing, machine learning, and network analysis tool for Python. It comes with a host of tools for data mining (Google, Twitter, Wikipedia API, a web crawler and an HTML DOM parser), NLP (part-of-speech taggers, n-gram search, sentiment analysis, WordNet), ML (vector space model, clustering, SVM) and network analysis by graph centrality and visualization.

TextBlob :

- It provides an easy interface to learn basic NLP tasks like sentiment analysis, noun phrase extraction, or pos - tagging. TextBlob is a Python (2 and 3) library designed for processing textual data. It focuses on providing access to common text - processing operations through familiar interfaces. TextBlob objects can be treated as Python strings that are trained in Natural Language Processing.
- TextBlob offers a neat API for performing common NLP tasks like part - of - speech tagging, noun phrase extraction, sentiment analysis, classification, language translation, word inflection, parsing, n - grams, and WordNet integration.

PyNLPl :

- Pronounced as 'pineapple,' PyNLPl is a Python library for Natural Language Processing. It contains a collection of custom-made Python modules for Natural Language Processing tasks. One of the most notable features of PyNLPl is that it features an extensive library for working with FoLiA XML (Format for Linguistic Annotation).
- PyNLPl is segregated into different modules and packages, each useful for both standard and advanced NLP tasks. While you can use PyNLPl for basic NLP tasks like extraction of n-grams and frequency lists and to build a simple language model, it also has more complex data types and algorithms for advanced NLP tasks.

Quepy :

Quepy is used to transform natural language questions into queries in a database query language.

SpaCy :

- SpaCy is an open - source NLP library which is used for data extraction, data analysis, sentiment analysis, and text summarization. SpaCy is an open-source NLP library in Python. It is designed explicitly for production usage - it lets you develop applications that process and understand huge volumes of text.
- SpaCy can preprocess text for Deep Learning. It can be used to build natural language understanding systems or information extraction systems. SpaCy is equipped with pre-trained statistical models and word vectors. It can support tokenization for over 49 languages. SpaCy boasts of state - of - the - art speed, parsing, named entity recognition, convolutional neural network models for tagging and deep learning integration.

Gensim :

- Gensim works with large datasets and processes data streams. Gensim is a Python library designed specifically for "topic modeling, document indexing, and similarity retrieval with large corpora." All algorithms in Gensim are memory-independent, w.r.t., the corpus size, and hence, it can process input larger than RAM.
- With intuitive interfaces, Gensim allows for efficient multicore implementations of popular algorithms, including online Latent Semantic Analysis (LSA / LSI / SVD), Latent Dirichlet Allocation (LDA), Random Projections (RP), Hierarchical Dirichlet Process (HDP) or word2vec deep learning.

Review Questions

1. Discuss NLP in brief.
2. How NLP is different than traditional computational processing ?
3. Why NLP is considered difficult than computer language processing ?
4. What are the challenges in NLP ?
5. Discuss various difficulties in NLP.
6. Explain four paradigms in NLP.
7. What are the advantages of using NLP in business applications? Discuss any four advantages of NLP.
8. What are the limitations / disadvantages of NLP based systems ?
9. Write a short note on

   a. NLU b. NLG
10. Differentiate between NLU and NLG.
11. Discuss any Four common Natural Language Processing applications.
12. Match the pair.

Sr. No.	NLP System	Sr. No.	Example
1.	Human voice response systems	a	Spam detection in emails
2.	Spelling correctors	b	Alexa, Siri
3.	Language translators	c	Twitter Sentiment Analyser
4.	Text response systems	d	MS Word, Automated Dictionaries
5.	Word (Entity) recognition	e	Google translate, Translate option in browsers
6.	Opinion detection / sentiment analysis	f	Chatbots

13. Discuss any five stages of NLP pipeline.
14. Draw stages of NLP pipeline and discuss anyone stage.
15. Write a note on phases in NLP applications.
16. List out phases of NLP applications and discuss one phase in detail.
17. What are the different NLP APIs available for developing NLP applications.
18. Discuss any two Python based NLP libraries.

❑❑❑

Chapter 2
Language Modeling & Part of Speech Tagging

Syllabus

Unigram Language Model, Bigram, Trigram, N-gram, Advanced smoothing for language modeling, Empirical Comparison of Smoothing Techniques, Applications of Language Modeling, Natural Language Generation, Parts of Speech Tagging, Morphology, Named Entity Recognition

Contents

2.1 Introduction

- Natural language processing is different than that of formal languages or programming languages. This is primarily because
  - Formal languages / programming languages, can be fully specified.
  - All the reserved words in formal language can be defined and identified vice a versa.
- However, it is not possible with natural language. Natural languages are not designed; they evolve continually and therefore there is no formal specification.
- In case of formal languages, there are formal rules for parts of the language and heuristics, but natural language that does not confirm is often used.
- Natural languages involve vast numbers of terms that do not create ambiguities for human brains but create ambiguities in understanding by means of computers.
- An alternative approach is to specify the model of the natural language.
- The key knowledge of the almost 70 years of research in natural language processing can be captured through the use of a small number of formal models or theories.
- Language models are a crucial component in the Natural Language Processing (NLP).
- These language models power all the popular NLP applications we are familiar with - Google Assistant, Siri, Amazon's Alexa, etc.

Language modeling for NLP

- Statistical Language Modeling or Language Modeling and LM for short, is the development of probabilistic models that are able to predict the next word in the sequence given the words that precede it.
- A language model learns the probability of word occurrence based on examples of text. Simpler models may look at a context of a short sequence of words, whereas larger models may work at the level of sentences or paragraphs. Most commonly, language models operate at the level of words.
- A language model can be developed and used standalone, such as to generate new sequences of text that appear to have come from the corpus.
- Language modeling is a root problem for a large range of natural language processing tasks. More practically, language models are used on the front-end or back-end of a more sophisticated model for a task that requires language understanding.
- Language modeling is central to many important natural language processing tasks.
- Recently, neural-network-based language models have demonstrated better performance than classical methods both standalone and as part of more challenging natural language processing tasks.

- There may be formal rules for parts of the language, and heuristics, but natural language that does not confirm is often used. Natural languages involve vast numbers of terms that can be used in ways that introduce all kinds of ambiguities, yet can still be understood by other humans.
- Further, languages change, word usages change: it is a moving target.Nevertheless, linguists try to specify the language with formal grammars and structures. It can be done, but it is very difficult and the results can be fragile.
- An alternative approach to specifying the model of the language is to learn it from examples.
- There are primarily two types of language models :
  1. **Statistical language models :** These models use traditional statistical techniques like N-grams, Hidden Markov Models (HMM) and certain linguistic rules to learn the probability distribution of words.
  2. **Neural language models :** These are new in the NLP town and have surpassed the statistical language models in their effectiveness. They use different kinds of neural networks to model language.
- These models and theories are all drawn from the standard toolkits of Computer Science, Mathematics and Linguistics and should be generally familiar to the experts in the respective fields. One can re-visit these terms (highlighted below) from the course Theory of Computation in the earlier years of Computer Science/ Computer Engineering.

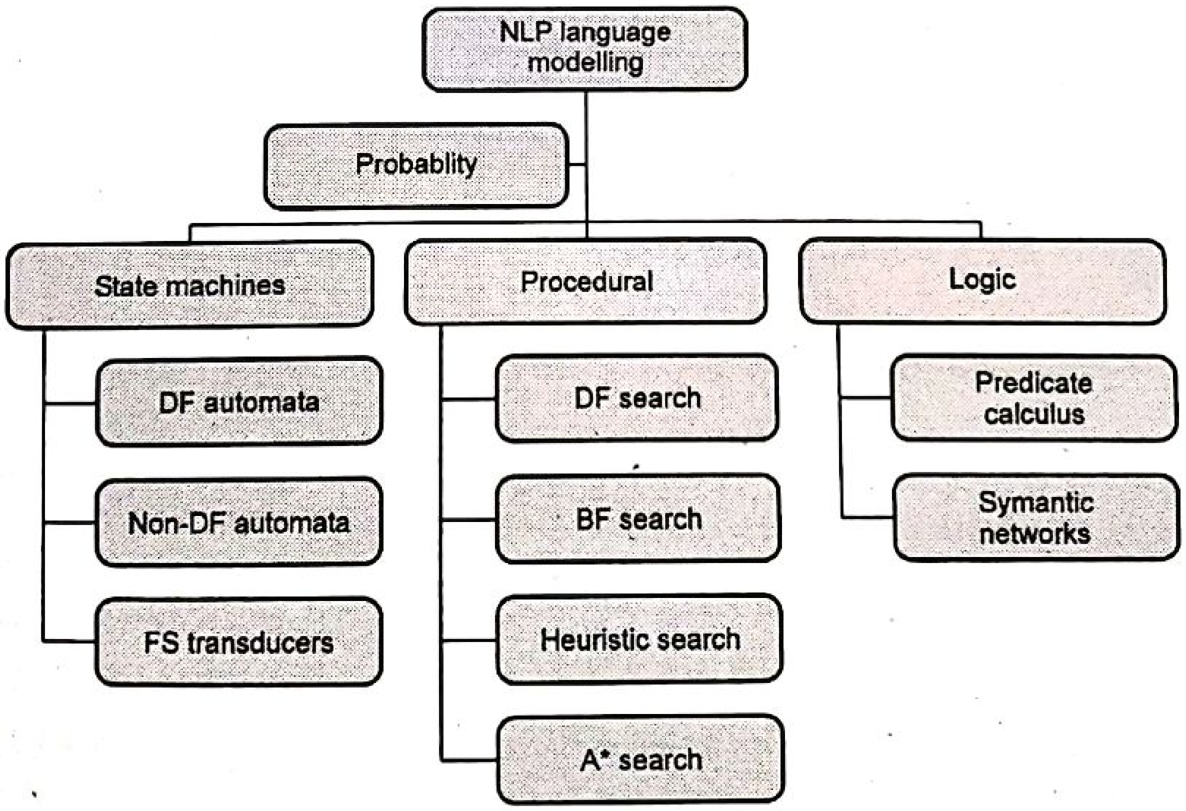


Fig. 2.1.1 : Spectrum of NLP language modeling

Statistical Language Model

- A statistical language model is a probability distribution over sequences of words.
- Given such a sequence, say of length m, it assigns $P(w_1, \ldots\ldots w_m)$ a to the whole sequence.
- The language model provides context to distinguish between words and phrases that sound phonetically similar.

  For example, in English, the phrases "do raid" and "do red" sound similar, but mean different things.
- Data sparsity is a major problem in building language models. This is primarily due to limitations in data modelling. Most possible word sequences are not observed in training.
- One solution is to make the assumption that the probability of a word only depends on the previous n words. This is known as an n-gram model or unigram model when n = 1.
- The unigram model is also known as the bag of words model.
- Estimating the relative likelihood of different phrases is useful in many natural language processing applications, especially those that generate text as an output.
- Discussed as in above example, of "do raid" and "do red", sounds are matched with word sequences.
- Ambiguities are easier to resolve when evidence from the language model is integrated with a pronunciation model and an acoustic model.
- Language models are used in information retrieval in the query likelihood model. There, a separate language model is associated with each document in a collection.
- Documents are ranked based on the probability of the query Q in the document's language model M_d : $P(Q \mid M_d)$. Commonly, the unigram language model is used for document ranking.

2.2 Unigram Language Model

- A unigram model can be treated as the combination of several one-state finite automata. It splits the probabilities of different terms in a context, e.g. from

$$P(t_1 t_2 t_3) = P(t_1) P(t_2 \mid t_1) P(t_3 \mid t_1 t_2)$$

to

$$P_{uni}(t_1 t_2 t_3) = P(t_1) P(t_2) P(t_3)$$

- In unigram language model, the probability of each word only depends on that word's own probability in the document, so we only have one-state finite automata as units.

- The following is an example of a unigram model of a document, that contains some terms / words as a, world, likes, we, share, .. and many more.

$$\sum_{\text{term in doc}} P(\text{term}) = 1$$

Terms	Probability in doc
a	0.1
world	0.2
likes	0.05
we	0.05
share	0.3
...	...

- The probability generated for specific terms is calculated as

$$P(\text{query}) = \prod_{\text{term in query}} P(\text{term})$$

- Different documents have unigram models, with different hit probabilities of words in it. The probability distributions from different documents are used to generate hit probabilities for each query. Documents can be ranked for a query according to the probabilities. Example of unigram models of two documents :

Terms	Probability in Doc1	Probability in Doc2
a	0.1	0.3
world	0.2	0.1
likes	0.05	0.03
we	0.05	0.02
share	0.3	0.2
...	...	...

- In information retrieval contexts, unigram language models are often smoothed to avoid instances where P(term) = 0.
- A common approach is to generate a maximum-likelihood model for the entire collection and linearly interpolate the collection model with a maximum-likelihood model for each document to smooth the model.

2.3 N-gram

- In the fields of computational linguistics and probability, an n-gram is a contiguous sequence of n items from a given sample of text or speech.
- The items can be phonemes, syllables, letters, words or base pairs according to the application. The n-grams typically are collected from a text or speech corpus. When the items are words, n-grams may also be called shingles. small
- Using Latin numerical prefixes, an n-gram of size 1 is referred to as a "unigram"; size 2 is a "bigram" (or, less commonly, a "diagram"); size 3 is a "trigram". English cardinal numbers are sometimes used, e.g., "four-gram", "five-gram" and so on. Here is an example of some n-grams frequently found in titles of publications about Corona virus disease 2019.

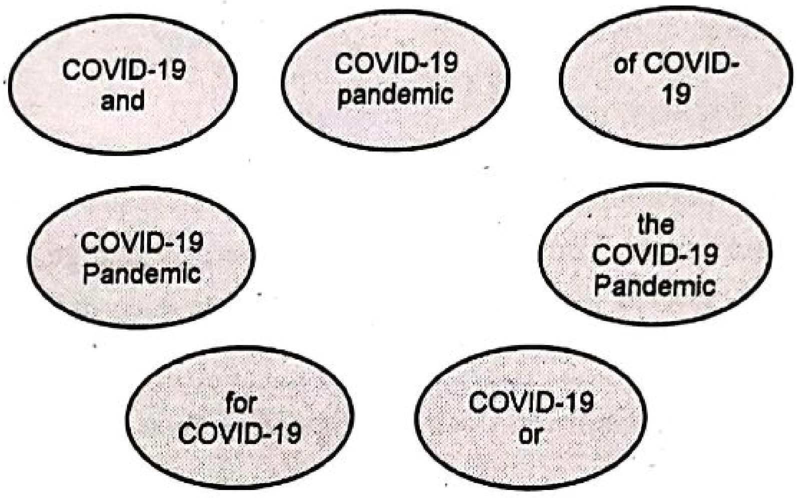


Fig. 2.3.1 : n-grams found in titles of publications about Corona virus disease 2019

Applications of n-gram models :

- An n-gram model is a type of probabilistic language model for predicting the next item in such a sequence in the form of a $(n - 1)$ order Markov model.
- n-gram models are now widely used in probability, communication theory, computational linguistics (for instance, statistical natural language processing), computational biology (for instance, biological sequence analysis) and data compression.
- Two benefits of n-gram models (and algorithms that use them) are simplicity and scalability.
- With larger n, a model can store more context with a well-understood space–time tradeoff, enabling small experiments to scale up efficiently.

- n-grams find use in several areas of computer science, computational linguistics and applied mathematics. They have been used to :
  1. Design kernels that allow machine learning algorithms such as support vector machines to learn from string data.
  2. Find likely candidates for the correct spelling of a misspelled word.
  3. Improve compression in compression algorithms where a small area of data requires n-grams of greater length.
  4. Assess the probability of a given word sequence appearing in text of a language of interest in pattern recognition systems, speech recognition, OCR (optical character recognition), intelligent character recognition (ICR), machine translation and similar applications.
  5. Improve retrieval in information retrieval systems when it is hoped to find similar "documents" (a term for which the conventional meaning is sometimes stretched, depending on the data set) given a single query document and a database of reference documents.
  6. Improve retrieval performance in genetic sequence analysis as in the blast family of programs.
  7. Predict letters or words at random in order to create text, as in the dissociated press algorithm.
  8. Cryptanalysis
- The terms bigram and trigram language models denote n-gram models with n = 2 and n = 3 respectively.

2.4 Bigram

- The bigram model approximates the probability of a word given all the previous words by using only the conditional probability of one preceding word.
- In the other words, when we use a bigram model to predict the conditional probability of the next word, resulting in the following approximation :

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-1})$$

- A bigram or diagram is a sequence of two adjacent elements from a string of tokens, which are typically letters, syllables or words. A bigram is an n-gram for n = 2.
- The frequency distribution of every bigram in a string is commonly used for simple statistical analysis of text in many applications, including in computational linguistics, cryptography, speech recognition and so on.

- Bigrams help provide the conditional probability of a token given the preceding token, when the relation of the conditional probability is applied :

$$P(W_n \mid W_{n-1}) = \frac{P(w_{n-1}, w_n)}{(W_{n-1})}$$

- That is, the probability P() of the given token W_n given the preceding token W_{n-1} is equal to the probability of their bigram or the co-occurrence of the two tokens $P(W_{n-1}, W_n)$ divided by the probability of the preceding token.

Applications

- Bigrams are used in most successful language models for speech recognition. These are a special case of N-gram.
- Bigram frequency attacks can be used in cryptography to solve cryptograms. See frequency analysis.
- Bigram frequency is one approach to statistical language identification.
- Some activities in logology or recreational linguistics involve bigrams. These include attempts to find english words beginning with every possible bigram, or words containing a string of repeated bigrams, such as logogogue.

2.5 Trigram

- Trigrams are a special case of the n-gram, where n is 3. Trigram models, conditions on the previous two words rather than only the previous word.
- The basic problem in trigram language modelling is to estimate the probability of a word given the two words preceding it. This is typically done by smoothing the maximum likelihood estimate.
- The probability of trigram is calculated as

$$\hat{p}ML(w_3 \mid w_1, w_2) = \frac{c(w_1\ w_2\ w_3)}{c(w_1\ w_2)}$$

- They are often used in natural language processing for performing statistical analysis of texts and in cryptography for control and use of ciphers and codes.
- The trigram assumption is arguably quite strong and linguistically naive. However, it leads to models that are very useful in practice.

2.6 Advanced Smoothing for Language Modeling

What is smoothing techniques in NLP ?

main

- Language models are a staple in many domains including speech recognition, optical character recognition, handwriting recognition, machine translation and spelling correction.

- The dominant technology in language modeling is n-gram models, which are straightforward to construct except for the issue of smoothing, a technique used to better estimate probabilities when there is insufficient data to estimate probabilities accurately.

Why smoothing techniques are required ?

- A key problem in N-gram modeling is the inherent data sparseness.

  For example, in several million words of english text, more than 50 % of the trigrams occur only once; 80 % of the trigrams occur less than five times (see SWB data also).
- Higher order N-gram models tend to be domain or application specific. Smoothing provides a way of generating generalized language models.
- If an N-gram is never observed in the training data, can it occur in the evaluation data set?
- **Solution :** Smoothing is the process of flattening a probability distribution implied by a language model so that all reasonable word sequences can occur with some probability. This often involves broadening the distribution by redistributing weight from high probability regions to zero probability regions.
- Smoothing techniques in NLP are used to address scenarios related to determining probability / likelihood estimate of a sequence of words (say, a sentence) occurring together when one or more words individually (unigram) or N-grams such as bigram(w_i/w_i-1) or trigram ($w_i/w_i - 1/w_i - 2$) in the given set have never occured in the past.

When smoothing techniques are useful

Chen and Goodman (1998) had stated that applicability of smoothing techniques as follows :

- Whenever data sparsity is an issue, smoothing can help performance and data sparsity is almost always an issue in statistical modeling.
- In the extreme case where there is so much training data that all parameters can be accurately trained without smoothing, one can almost always expand the model, such as by moving to a higher n-gram model, to achieve improved performance.
- With more parameters data sparsity becomes an issue again, but with proper smoothing the models are usually more accurate than the original models.
- Thus, no matter how much data one has, smoothing can almost always help performance, and for a relatively small effort.

Smoothing techniques for language modeling

- Additive smoothing / Laplace smoothing
- Good-turing estimate
- Kneser-ney smoothing
- Katz smoothing (back-off)

- Absolute discounting

Note : *There are many more smoothing techniques used in NLP. However we have discussed few due to space and time limitations of syllabus.*

Test data for smoothing techniques

Let's take the following sequence of words as corpus and test data set.

• **Corpus (Training Data) :** The following represents the corpus of words :	• Test Data
▯ cats chase rats ▯ cats meow ▯ rats chatter ▯ cats chase birds ▯ rats sleep	▯ rats chase birds ▯ cats sleep

- Based on the training data set, what is the probability of "cats sleep" assuming bigram technique is used? Based on bigram technique, the probability of the sequence of words "cats sleep" can be calculated as the product of following :

$$P(\text{catssleep}) = P\left(\frac{\text{cats}}{\text{<s>}}\right) \times P\left(\frac{\text{sleep}}{\text{cats}}\right) \times P\left(\frac{\text{</s>}}{\text{sleep}}\right)$$

- One will notice that $P\left(\frac{\text{sleep}}{\text{cats}}\right)$.Thus, the overall probability of occurrence of "cats sleep" would result in zero (0) value. However, the probability of occurrence of a sequence of words should not be zero at all. This is where various different smoothing techniques come into the picture.

2.6.1 Additive Smoothing / Laplace Smoothing

- The simplest way to do smoothing is to add one to all the bigram counts, before we normalize them into probabilities. All the counts that used to be zero will now have a count of 1, the counts of 1 will be 2 and so on. This algorithm is also called Laplace smoothing.
- Laplace smoothing does not perform well enough to be used smoothing in modern n-gram models, but it usefully introduces many of the concepts that we see in other smoothing algorithms, gives a useful baseline and is also a practical smoothing algorithm for other tasks like text classification.
- Let's start with the application of Laplace smoothing to unigram probabilities. Recall that the unsmoothed maximum likelihood estimate of the unigram probability of the word w_i is

its count c_i normalized by the total number of word tokens N :

$$P(w_i) = \frac{c_i}{N}$$

- Laplace smoothing merely adds one to each count (hence its alternate name addone smoothing). Since there are V words in the vocabulary and each one was incremented, we also need to adjust the denominator to take into account the extra V observations.

$$P_{Laplace}(w_i) = \frac{c_i + 1}{N + V}$$

- A related way to view smoothing is as discounting (lowering) some non-zero counts in order to get the probability mass that will be assigned to the zero counts.

2.6.2 Good-Turing Estimate

- Good turing smoothing technique uses the frequencies of the count of occurrence of N-Grams for calculating the maximum likelihood estimate.
- For example, consider calculating the probability of a bigram (chatter/cats) from the corpus given above. Note that this bigram has never occurred in the corpus and thus, probability without smoothing would turn out to be zero. As per the good-turing smoothing, the probability will depend upon the following:
  - o In case, the bigram (chatter/cats) has never occurred in the corpus (which is the reality), the probability will depend upon the number of bigrams which occurred exactly one time and the total number of bigrams.
  - o In case, the bigram has occurred in the corpus (for example, chatter/rats), the probability will depend upon number of bigrams which occurred more than one time of the current bigram (chatter/rats) (the value is 1 for chase/cats), total number of bigram which occurred same time as the current bigram (to/bigram) and total number of bigram.
- For the unknown N-grams, the following formula is used to calculate the probability

$$P_{unknown}\left(\frac{w_i}{w_{i-1}}\right) = \frac{N_1}{N}$$

- In above formula, N_1 is count of N-grams which appeared one time and N is count of total number of N-grams.

For the known N-grams, the following formula is used to calculate the probability :

$$P\left(\frac{w_i}{w_{i-1}}\right) = \frac{N_1}{N}$$

$$\text{Where} \quad c^* = (c+1) \times \frac{N_{t+1}}{N_c}$$

- In the above formula, c represents the count of occurrence of n-gram, N_{c+1} represents count of n-grams which occurred for c + 1 times, N_c represents count of n-grams which occurred for c times and N represents total count of all n-grams.

2.6.3 Katz Smoothing

In Katz smoothing good-turing technique is combined with interpolation. It outperforms good-turing by redistributing different probabilities to different unseen units.

2.6.4 Kneser-Ney Smoothing

- In good turing smoothing, it is observed that the count of n-grams is discounted by a constant/absolute value such as 0.75. The same intuition is applied for Kneser-Ney smoothing where absolute discounting is applied to the count of n-grams in addition to adding the product of interpolation weight and probability of word to appear as novel continuation.

$$P_{Kneser-Ney}\left(\frac{w_1}{w_{t-1}}\right) = \frac{\max(c\, w_{t-1}, w_1 - d, 0)}{c(w_{t-1})} + \lambda(w_{i-1}) \times P_{continuation}(w_i)$$

where λ is a normalizing constant which represents probability mass that have been discounted for higher order. The following represents how λ is calculated :

$$\lambda(w_{i-1}) = \frac{d \times |c(w_{t-1}, w_i)|}{c(w_{i-1})}$$

2.6.5 Absolute Discounting

- Absolute discounting involves subtracting a fixed discount, D, from each nonzero count, an redistributing this probability mass to N-grams with zero counts.

$$P_{abs}(w_i | w_{i-n+1} \cdots w_{i-1}) = \frac{\max\{c(w_{i-n+1} \cdots w_i) - D, 0\}}{\sum_{w_i} (w_{i-n+1} \cdots w_i} +$$

$$(1 - \lambda w_{i-n+1} \cdots w_{i-1})\, P_{abs}(w_i | w_{i-n+1} \cdots w_{i-1})$$

2.7 Empirical Comparison of Smoothing Techniques

- In order to completely characterize the relative performance of two techniques, it is necessary to consider multiple training set sizes and to try both bigram and trigram models.
- Multiple runs should be performed whenever possible to discover whether any calculated differences are statistically significant.

- Furthermore, it is also required to show that sub-optimal parameter selection can also significantly affect relative performance.
- Katz smoothing and Jelinek-Mercer smoothing, perform consistently well across training set sizes for both bigram and trigram models, with Katz smoothing performing better on trigram models produced from large training sets and on bigram models in general.
- These results question the generality of Katz smoothing : Katz (1987) reported that his method slightly outperforms an unspecified version of Jelinek-Mercer smoothing on a single training set of 750,000 words.
- Furthermore, it is observed that Church-Gale smoothing, which previously had not been compared with common smoothing techniques, outperforms all existing methods on bigram models produced from large training sets.
- Additive smoothing performs poorly and that methods katz and interpolation p-held-out consistently perform well.
- Finally, it is found that that our novel smoothing methods average-count and one-count are superior to existing methods for trigram models and perform well on bigram models; method one-count yields marginally worse performance but is extremely easy to implement.
- In this study, we measure performance solely through the cross-entropy of test data; it would be interesting to see how these cross-entropy differences correlate with performance in end applications such as speech recognition.
- In addition, it would be interesting to see whether these results extend to fields other than language modeling where smoothing is used, such as prepositional phrase attachment (Collins and Brooks, 1995), part-of-speech tagging (Church, 1988) and stochastic parsing (Magerman, 1994)

2.8 Applications of Language Modelling

- A statistical language model is a probability distribution over sequences of words.
- Estimating the relative likelihood of different phrases is useful in many natural language processing applications, especially those that generate text as an output.

Unigram language modeling is used in

1. Speech recognition
2. Machine translation
3. Part-of-speech tagging

4. Parsing
5. Optical character recognition
6. Handwriting recognition
7. Information retrieval, etc.

Applications of n-gram models :

- An n-gram model is a type of probabilistic language model for predicting the next item in such a sequence in the form of a (n – 1) order Markov model.
- n-gram models are now widely used in probability, communication theory, computational linguistics (for instance, statistical natural language processing), computational biology (for instance, biological sequence analysis) and data compression.
- Two benefits of n-gram models (and algorithms that use them) are simplicity and scalability
- With larger n, a model can store more context with a well-understood space–time tradeoff, enabling small experiments to scale up efficiently.
- n-grams find use in several areas of computer science, computational linguistics, and applied mathematics. They have been used to:
  - o Design kernels that allow machine learning algorithms such as support vector machines to learn from string data.
  - o Find likely candidates for the correct spelling of a misspelled word.
  - o Improve compression in compression algorithms where a small area of data requires n-grams of greater length.
  - o Assess the probability of a given word sequence appearing in text of a language of interest in pattern recognition systems, speech recognition, ocr (optical character recognition), intelligent character recognition (icr), machine translation and similar applications.
  - o Improve retrieval in information retrieval systems when it is hoped to find similar "documents" (a term for which the conventional meaning is sometimes stretched, depending on the data set) given a single query document and a database of reference documents.
  - o Improve retrieval performance in genetic sequence analysis as in the blast family of programs.
  - o Identify the language a text is in or the species a small sequence of dna was taken from
  - o Predict letters or words at random in order to create text, as in the dissociated press algorithm.
  - o Cryptanalysis

2.9 Morphology

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

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

- The second column contains the stem of each word as well as assorted morphological features. These features specify additional information about the stem.

- Some of the commonly used features are
  1. N - Noun
  2. PL - Plural
  3. SG - Singular
  4. V - Verb
  5. PAST - Past tense
  6. PRES - Present tense
  7. FUTU - Future tense
  8. PART - Participle

Morphological Parser

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

Morphological recognition

- Finite State Automata (FSA)s can be used to solve the problem of morphological recognition; that is, of determining whether an input string of letters makes up a legitimate English word or not.
- It is done by expanding each arc (e.g. the reg-noun-stem arc) with all the morphemes that make up the set of reg-noun-stem. The resulting FSA can then be defined at the level of the individual letter. (Refer below Fig. 2.9.1)
- Morphological recognition is performed by taking the morphotactic FSAs and plugging in each 'sub-lexicon' into the FSA. i.e. each arc (e.g. the reg-noun-stem arc) is expanded with all the morphemes that make up the set of reg-noun-stem. The resulting FSA can then be defined at the level of the individual letter.
- Below is an FSA for a fragment of English derivational morphology.

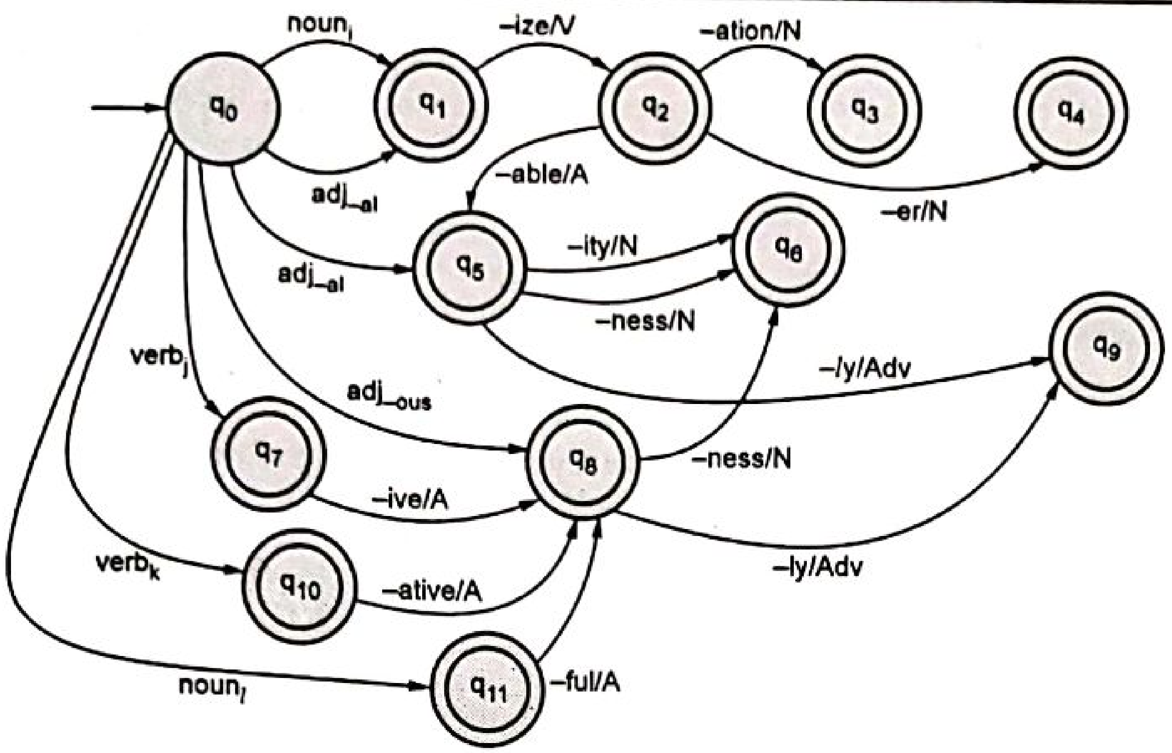


Fig. 2.9.1 : For a fragment of english derivational morphology

Morphological Parsing with Finite-State Transducers

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

  Example - The word like cats on the surface level is mapped into morpheme and features sequences likecat +N +PL on the lexical level.
- The below Fig. 2.9.2 shows two levels for the word cats. The lexical level has the stem for a word, followed by the morphological information +N +PL which tells us that cats is a plural noun.

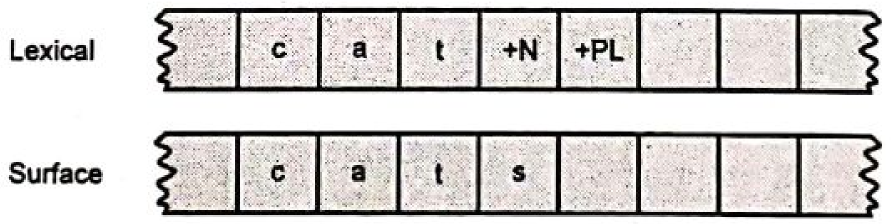


Fig. 2.9.2 : Example of lexical and surface tape

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

Finite-State Transducer or FST

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

Definition of FST

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

Let

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

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

q_0 : The start state

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

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

Example

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

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

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

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

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

Comparing FSA and FST

- Where FSAs are isomorphic to regular languages, FSTs are isomorphic to regular relations.
- FSTs and regular relations are closed under union, although in general they are not closed under difference, complementation and intersection.
- Besides union, FSTs have two additional closure properties that turn out to be extremely useful :

  inversion : The inversion of a transducer T (T^{-1}) simply switches the input and output labels. Thus if T maps from the input alphabet I to the output O, T^{-1} maps from O to I.

  composition : If T_1 is a transducer from I_1 to O_1 and T_2 a transducer from I_2 to O_2, then T_1. to T_2 maps from I_1 to O_2.
- Here is a transducer for English nominal number inflection T_{num}. Since both q_1 and q_2 are accepting states, regular nouns can have the plural suffix or not. The morpheme-boundary symbol ^ and word-boundary marker # are discussed below.

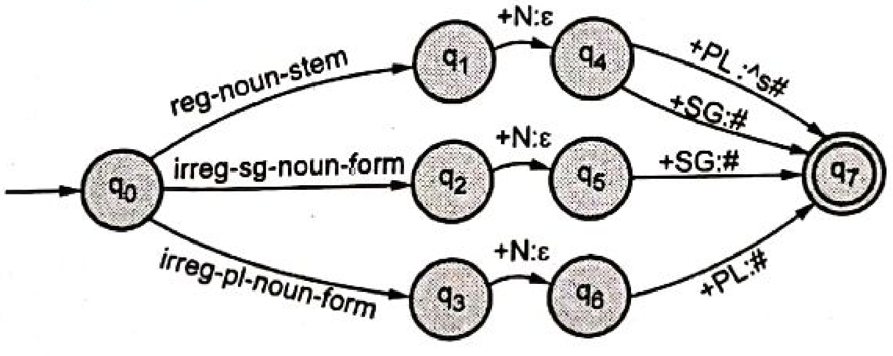


Fig. 2.9.3 A transducer for english nominal number inflection

Orthographic Rules and Finite-State Transducers

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

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

Generating or parsing with FST lexicon and rules

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

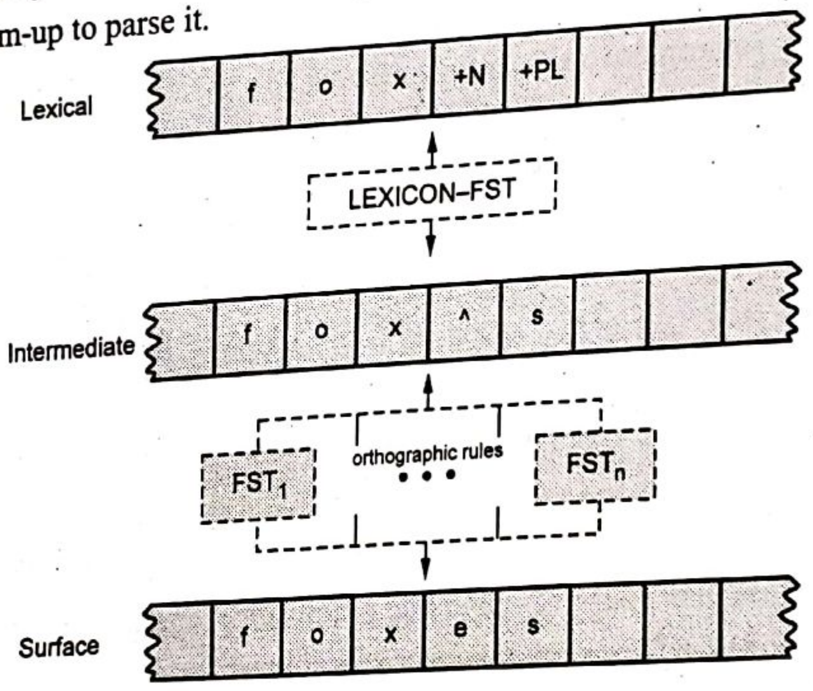


Fig. 2.9.4 : Generating or parsing with FST lexicon and rules

- The architecture in below Fig. 2.9.4 is a two-level cascade of transducers that is used for generating or parsing with FST lexicon and rules.
- Parsing can be slightly more complicated than generation, because of the problem of ambiguity. Disambiguating requires some external evidence such as the surrounding words.
- The power of finite-state transducers is that the exact same cascade with the same state sequences is used when the machine is generating the surface tape from the lexical tape or when it is parsing the lexical tape from the surface tape.

- Transducers in parallel can be combined by automaton intersection. The automaton intersection algorithm just takes the cartesian product of the states.

2.10 Part of Speech Tagging

- Part-Of-Speech (POS) tagging is a very crucial process in NLP. After a string is recognized / accepted by an FST, it needs to be tokenized (also referred as tagged in NLP).
- Part-Of-Speech (POS) tagging (or just tagging for short) is the TAGGING process of assigning a part-of-speech or other lexical class marker to each word in a corpus. (Refer Fig. 2.10.1) In NLP tags are applied to punctuation markers too.

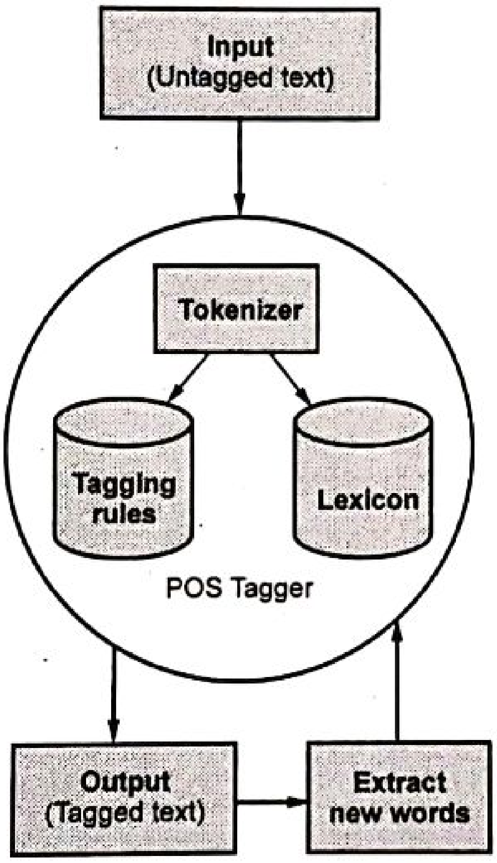


Fig. 2.10.1 : PoS tagging process

- The tagging for natural language is the same process as tokenization for computer languages, although tags for natural languages are much more ambiguous.
- Taggers play an increasingly important role in speech recognition, natural language parsing and information retrieval.
- The input to a tagging algorithm is a string of words and a specified tagset of the kind described in the previous section. The output is a single best tag for each word.
- The problem of POS-tagging is to resolve the ambiguities, choosing the proper tag for the context. Part-of-speech tagging is thus one of the many disambiguation task in NLP.

Tagging is complex

Most words in English are unambiguous; i.e. they have only a single tag. But many of the most common words of English are ambiguous (for example 'can' can be an auxiliary ('to be able'), a noun ('a metal container'), or a verb ('to put something in such a metal container').

- Here are some different POS tags with their corresponding meaning as follows

CC - Coordinating conjunction	PRP - Personal pronoun
CD - Cardinal number	RB - Adverb
DT - Determiner	RBR - Adverb, comparative
EX - Existential there	RBS - Adverb, superlative
FW - Foreign word	RP - particle
IN - Preposition or subordinating conjunction	SYM - Symbol
JJ - Adjective	TO - to
JJR - Adjective, comparative	UH - Interjection
JJS - Adjective, superlative	VB - Verb, base form
NN - Noun, singular or mass	VBD - Verb, past tense
NNS - Noun, plural	VBG - Verb, gerund or present particle
NNP - Proper, noun, singular	VBN - Verb, past participle
NNPS - Proper noun, plural	VBP - Verb, non-3rd person singular present
PDT - Predeterminer	VBZ - Verb, 3rd person singular present
NP - Noun phrase	WDT - Wh-determiner
PP - Prepositional phrase	WP - Wh-pronoun
VP - Verb phrase	WRB - Wh-adverb

Fig. 2.10.2 POS tags

- ***Tagging algorithms*** fall into one of two classes : Rule based taggers and stochastic taggers.

Rule-based taggers

- Rule-based taggers generally involve a large database of hand-written disambiguation rule which specify, for example,that an ambiguous word is a noun rather than a verb if it follows a determiner.

  e.g. ENGTWOL, based on the constraint grammar architecture of Karlsson et al.

- It is based on rules which determine when an ambiguous word should have a given tag.

- It uses 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 analysing 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 -
  1. Context-pattern rules
  2. 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 -
  1. **First stage :** In the first stage, it uses a dictionary to assign each word a list of potential parts-of-speech.
  2. **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 taggers
  1. These taggers are knowledge-driven taggers.
  2. The rules in rule-based POS tagging are built manually.
  3. The information is coded in the form of rules.
  4. There are limited number of rules
  5. Smoothing and language modeling is defined explicitly in rule-based taggers.

Stochastic taggers :

- Stochastic taggers generally resolve tagging ambiguities by using a training corpus to compute the probability of a given word having a given tag in a given context.
- It has a machine-learning component the rules are automatically induced from a previously-tagged training corpus.

  e.g. HMM tagger, also called a Maximum Likelihood Tagger, or a Markov model HMM tagger
- This model includes frequency or probability (statistics) can be called stochastic. The simplest stochastic tagger applies the following approaches for POS tagging -
  1. **Word frequency approach :** In this approach, the stochastic taggers disambiguate the words based on the probability that a word occurs with a particular tag. The main issue with this approach is that it may yield inadmissible sequence of tags.

2. **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.

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

2.11 Named Entity Recognition

- Named-Entity Recognition (NER) (also known as (named) entity identification, entity chunking, and entity extraction) is a subtask of information extraction that seeks to locate and classify named entities mentioned in unstructured text into pre-defined categories
- Named entity recognition (NER) – also called entity identification or entity extraction – is a natural language processing (NLP) technique that automatically identifies named entities in a text and classifies them into predefined categories.
- Entities can be names of people, organizations, locations, medical codes, time expressions, times, quantities, monetary values, percentages and more.

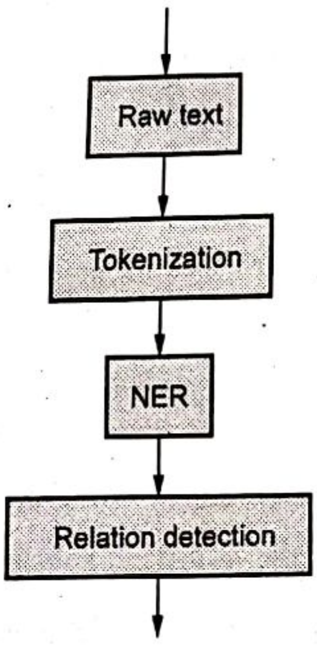


Fig. 2.11.1 Named Entity Recognition (NER) in NLP

NER example : In the sentence "Dheerubhai Ambani Founded Reliance Industries in India " we can identify three types of entities :

<Person>	→	Dheerubhai Ambani
<Company>	→	Reliance Industries
<Location>	→	India

- Recognition of these entities is done through machine learning and Natural Language Processing (NLP).
- With named entity recognition, one can extract key information to understand what a text is about or merely use it to collect important information to store in a database. NER systems have been created that use linguistic grammar-based techniques as well as statistical models such as machine learning. Hand-crafted grammar-based systems typically obtain better precision, but at the cost of lower recall and months of work by experienced computational linguists.
- Statistical NER systems typically require a large amount of manually annotated training data.
- Popular NER platforms include :
  1. GATE supports NER across many languages and domains out of the box, usable via a graphical interface and a Java API.
  2. OpenNLP includes rule-based and statistical named-entity recognition.
  3. SpaCy features fast statistical NER as well as an open-source named-entity visualizer.
- APIs for NER - There are TWO types of NER APIs
- Open-source named entity recognition APIs
  1. **Open-source APIs are for developers :** They are free, flexible and entail a gentle learning curve. Here are a few options :
     - o **Stanford Named Entity Recognizer (SNER) :** This JAVA tool developed by Stanford University is considered the standard library for entity extraction. It's based on Conditional Random Fields (CRF) and provides pre-trained models for extracting person, organization, location and other entities.
     - o **SpaCy :** A Python framework known for being fast and very easy to use. It has an excellent statistical system that you can use to build customized NER extractors.
     - o **Natural Language Toolkit (NLTK) :** This suite of libraries for Python is widely used for NLP tasks. NLKT has its own classifier to recognize named entities called ne_chunk, but also provides a wrapper to use the Stanford NER tagger in python.
- SaaS named entity recognition APIs
  1. SaaS tools are ready-to-use, low-code, and cost-effective solutions. Plus, they are easy to integrate with other popular platforms.

2. Monkey Learn, for example, is a text analysis SaaS platform that you can use for different NLP tasks, one of which is named entity recognition. You can use MonkeyLearn's ready-built API to integrate pre-trained entity extraction models or you can easily build your own custom named entity extractor in just a few simple steps.

- NER used in

1. **Classifying content for news providers :** News and publishing houses generate large amounts of online content on a daily basis and managing them correctly is very important to get the most use of each article.

   Named entity recognition can automatically scan entire articles and reveal which are the major people, organizations and places discussed in them.

   Knowing the relevant tags for each article help in automatically categorizing the articles in defined hierarchies and enable smooth content discovery.

2. **Efficient search algorithms :** If for every search query the algorithm ends up searching all the words in millions of articles, the process will take a lot of time. Instead, if Named Entity Recognition can be run once on all the articles and the relevant entities (tags) associated with each of those articles are stored separately, this could speed up the search process considerably.

3. **Powering content recommendations :** One of the major uses cases of named entity recognition involves automating the recommendation process.

4. **Customer support :** There are a number of ways to make the process of customer feedback handling smooth and named entity recognition could be one of them.

5. **Research papers :** An online journal or publication site holds millions of research papers and scholarly articles. There can be hundreds of papers on a single topic with slight modifications. Segregating the papers on the basis of the relevant entities it holds can save the trouble of going through the plethora of information on the subject matter.

2.12 Natural Language Generation

- Natural-Language Generation (NLG) is a software process that produces natural language output.
- Common applications of NLG methods include the production of various reports, for example weather and patient reports image captions and chat bots.
- Automated NLG can be compared to the process humans use when they turn ideas into writing or speech.

- Psycholinguists prefer the term language production for this process, which can also be described in mathematical terms, or modeled in a computer for psychological research.

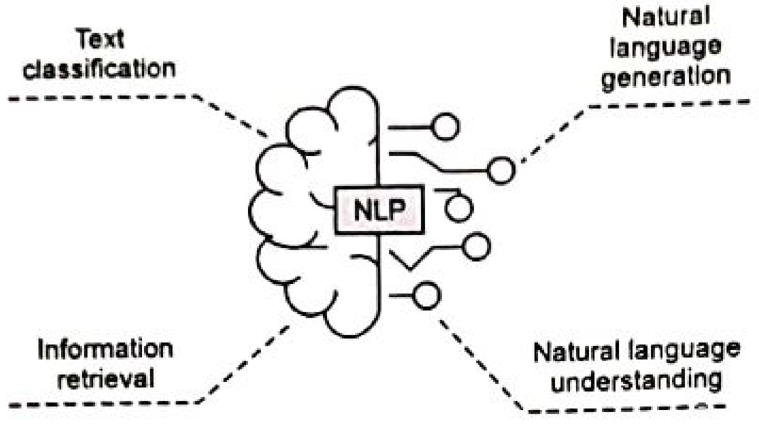


Fig. 2.12.1 Placing NLG in the context of NLP and NLU

- NLG systems can also be compared to translators of artificial computer languages, such as decompilers or transpilers, which also produce human-readable code generated from an intermediate representation.
- Human languages tend to be considerably more complex and allow for much more ambiguity and variety of expression than programming languages, which makes NLG more challenging.

NLP vs NLU vs. NLG

- Natural Language Processing (NLP) seeks to convert unstructured language data into a structured data format to enable machines to understand speech and text and formulate relevant, contextual responses. Its subtopics include natural language processing and natural language generation.
- Natural Language Understanding (NLU) focuses on machine reading comprehension through grammar and context, enabling it to determine the intended meaning of a sentence.
- Natural Language Generation (NLG) focuses on text generation, or the construction of text in English or other languages, by a machine and based on a given dataset.

Stages of NLG

- The process to generate text can be as simple as keeping a list of canned text that is copied and pasted, possibly linked with some glue text.
- The results may be satisfactory in simple domains such as horoscope machines or generators of personalised business letters.
- A sophisticated NLG system needs to include stages of planning and merging of information to enable the generation of text that looks natural and does not become repetitive.

- The typical stages of natural-language generation, as proposed by Dale and Reiterare as shown in Fig. 2.12.2 below.

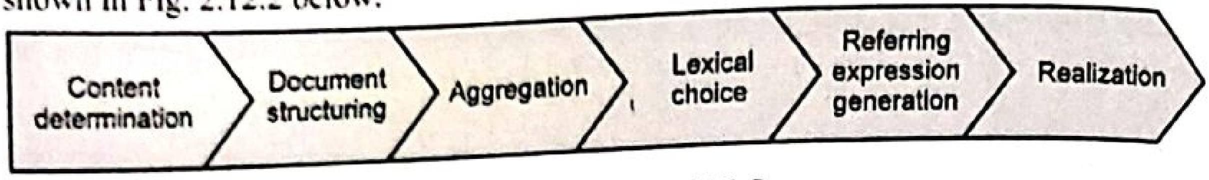


Fig. 2.12.2 Stages of NLG

- **Content determination :** Deciding what information to mention in the text.
- **Document structuring :** Overall organisation of the information to convey. For example, deciding to describe the areas with high pollen levels first, instead of the areas with low pollen levels.
- **Aggregation :** Merging of similar sentences to improve readability and naturalness. For instance, merging the two following sentences.
- **Lexical choice :** Putting words to the concepts. For example, deciding whether medium or moderate should be used when describing a pollen level of 4.
- **Referring expression generation :** Creating referring expressions that identify objects and regions. This task also includes making decisions about pronouns and other types of anaphora.
- **Realization :** Creating the actual text, which should be correct according to the rules of syntax, morphology and orthography. For example, using will be for the future tense of to be.

Alternative approach for NLG

- An alternative approach to NLG is to use "end-to-end" machine learning to build a system, without having separate stages as in Fig 2.12.2.
- An NLG system can also be built by training a machine learning algorithm (often an LSTM) on a large data set of input data and corresponding (human-written) output texts.
- Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning.
- The end-to-end approach has perhaps been most successful in image captioning,that is automatically generating a textual caption for an image.

NLG applications

- NLG makes data universally understandable making the writing of data-driven financial reports, product descriptions, meeting memos and more much easier and faster.
- It can take the burden of summarizing the data from analysts to automatically write reports that would be tailored to the audience.

- The main practical present-day applications of NLG are, therefore, connected with writing analysis or communicating necessary information to customers.
- NLG has more theoretical applications that make it a valuable tool not only in computer science and engineering, but also in cognitive science and psycholinguistics.
- Refer Fig 2.12.3 for practical and theoretical applications of NLG

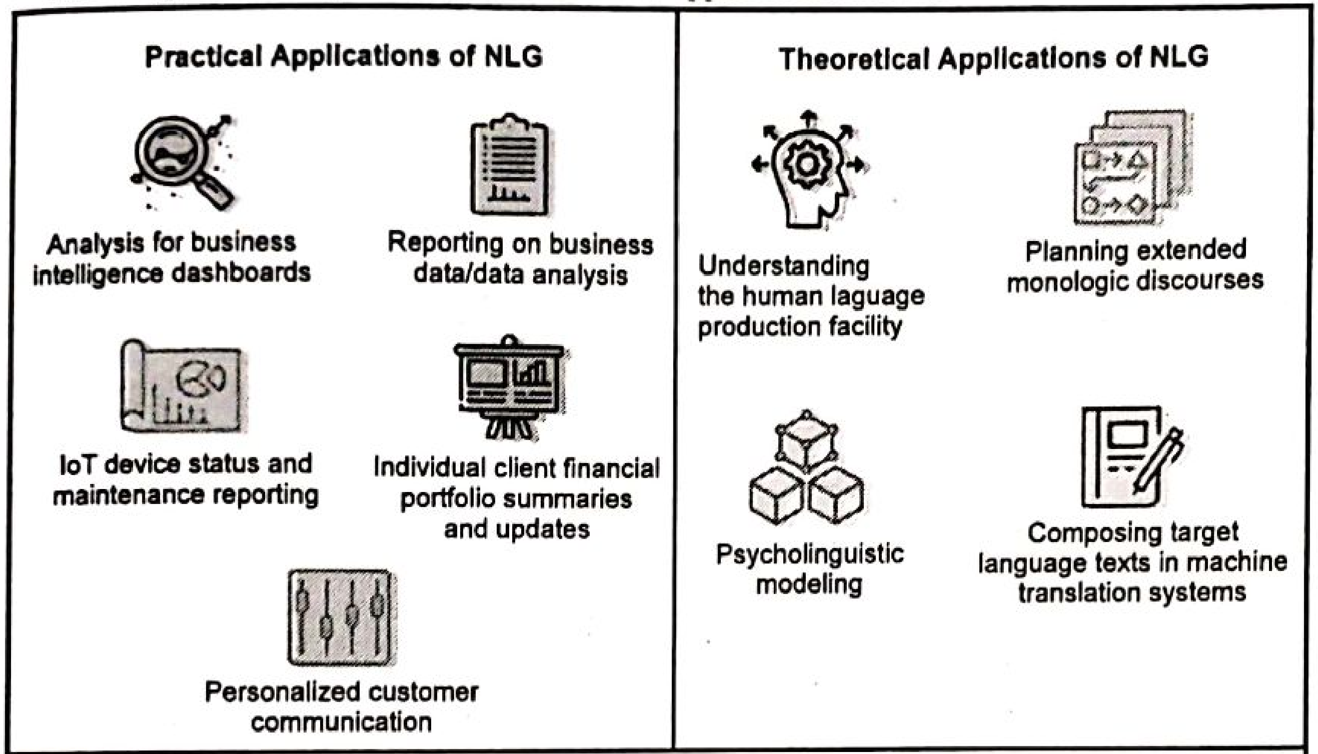

Practical Applications of NLG
Analysis for business intelligence dashboards
Reporting on business data/data analysis
IoT device status and maintenance reporting
Individual client financial portfolio summaries and updates
Personalized customer communication
Theoretical Applications of NLG
Understanding the human laguage production facility
Planning extended monologic discourses
Psycholinguistic modeling
Composing target language texts in machine translation systems

Fig. 2.12.3 Applications of NLG

NLG tools

- One can see that natural language generation is a complicated task that needs to take into account multiple aspects of language, including its structure, grammar, word usage and perception.
- Luckily, it won't build the whole NLG system from scratch as the market offers multiple ready-to-use tools, both commercial and open-source.
- Commercial NLG tools
  1. **Arria NLG PLC :** Is believed to be one of the global leaders in NLG technologies and tools and can boast the most advanced NLG engine and reports generated by NLG narratives.
  2. **AX semantics :** Offers eCommerce, journalistic and data reporting (e.g. BI or financial reporting) NLG services for over 100 languages. It is a developer-friendly product that uses AI and machine learning to train the platform's NLP engine.

3. Yseop is known for its smart customer experience across platforms like mobile, online or face-to-face. From the NLG perspective, it offers compose that can be consumed on-premises, in the cloud or as a service and offers Savvy, a plug-in for Excel and other analytics platforms.

4. **Wordsmith :** It is an automated insights product that is an NLG engine that works chiefly in the sphere of advanced template-based approaches. It allows users to convert data into text in any format or scale. Wordsmith also provides a plethora of language options for data conversion.

- Open-Source NLG tools

  1. **Simplenlg :** Is the most widely used open-source realiser, especially by system-builders. It is an open-source Java API for NLG written by the founder of Arria. It has the least functionality but also is the easiest to use and best documented.

  2. **Natural OWL :** Is an open-source toolkit which can be used to generate descriptions of OWL classes and individuals to configure an NLG framework to specific needs, without doing much programming.

Review Questions

1. Define the language modelling in NLP. Also discuss the types of language modelling in NLP.
2. Discuss the need of language modelling in NLP.
3. What is statistical language modelling ? / Discuss in brief statistical language modelling.
4. What is Unigram language model.
5. Discuss N-gram model with an example.
6. Write a short note on applications of N-gram language model.
7. Discuss Bi-gram model with an example.
8. Write a short note on applications of Bi-gram language model.
9. Write a short note on applications of Bi-gram language model.
10. Explain the concept of N-gram model and how bigram & trigram are derived.
11. What is smoothing techniques in NLP.
12. Why smoothing techniques are required in NLP.
13. Discuss the scenario when smoothing techniques are useful / List out the applicability of smoothing techniques.
14. List out smoothing techniques for language modeling and discuss any one technique.
15. Write a short note on any three of the following smoothing techniques.

a. Additive smoothing / Laplace smoothing

b. Good-Turing estimate

c. Kneser-Ney smoothing

d. Katz smoothing (backoff)

e. Absolute discounting

16. Write any three comparison points of smoothing techniques.
17. List out applications of

    a. Unigram language modeling

    b. N-gram language modeling
18. Define morphology parsing and discuss how it is used in NLP.
19. Write a note on :

    a. Morphological parser

    b. Morphological recognition using FSA
20. Explain morphological parsing with Finite-State transducers.
21. Define Finite-State transducer or FST in NLP.
22. Discuss the process of morphological parsing with FST lexicon and rules.
23. Compare use of FSA and FST in NLP.
24. Discuss POS tagging in NLP.
25. Write a note on rule based taggers in NLP.
26. Explain stochastic taggers for POS tagging.
27. What is NER in NLP ? / How named entity recognition is carried out in NLP.
28. List out various open source and propriatory NER APIs available for NLP applications.

❑❑❑

Chapter 3
Words and Word Forms

Syllabus

Bag of words, skip-gram, Continuous Bag-Of-Words, Embedding representations for words Lexical Semantics, Word Sense Disambiguation, Knowledge Based and Supervised Word Sense Disambiguation.

Contents

3.1 Bag of Words

- In NLP the input to various algorithms for different applications is in text form.
- But these machine learning algorithms cannot work on the raw text.
- The text must be converted to numbers for further processing.
- More specifically the vectors of these numbers representing text are generated.
- Bag Of Words (BOW) representation is used to convert text into fixed length vectors.
- BOW model takes into account the frequency of occurrence of words in input text document.
- It focuses on two things :
  1. Vocabulary of known words.
  2. Measure of presence of these known words.
- The word 'Bag' in bag of words is analogous to the bag of items. When we fill the bag the order of the items is discarded, similarly in bag of words model represents unordered set of words irrespective of their position in the document.
- It only considers the frequency of words in the text.
- Let's consider the following example to understand how bag of words model works stepwise.

Step 1 : Step 1 includes collection of data. Consider each of the following sentences as text documents.

- The dog sat
- The dog sat in the hat
- The dog with the hat

Step 2 :

- Step 2 includes designing the vocabulary.
- All the words present in document set are listed.
- In out example the words formed are : the, dog, sat, in, the, hat, with.

Step 3 :

- Step 3 includes computation of the frequency of the occurrence of words in the document.

- In our example it can be list as :

Document	the	dog	sat	in	hat	with
the dog sat	1	1	1	0	0	0
the dog sat in the hat	2	1	1	1	1	0
the dog with the hat	2	1	0	0	1	1

- With the 6 distinct words and their frequency of occurrence in the document, now we can write fixed length vector for each document as follows :
  - The dog sat → [1, 1, 1, 0, 0, 0]
  - The dog sat in the hat → [2, 1, 1, 1, 1, 0]
  - The dog with the hat → [2, 1, 0, 0, 1, 1]
- The drawback of BOW model is, we loose the context of the document.
- BOW model only tells what words will appear in the document with them frequency but it doesn't tell where they occurred.

3.2 Skip Gram

- Skip-gram is one of the unsupervised learning techniques used to find the most related words for a given word.
- Skip-gram is used to predict the context word for a given target word.
- It's a reverse of CBOW algorithm.
- Here, target word is input while context words are output. As there is more than one context word to be predicted which makes this problem difficult.
- Let's consider the following example :

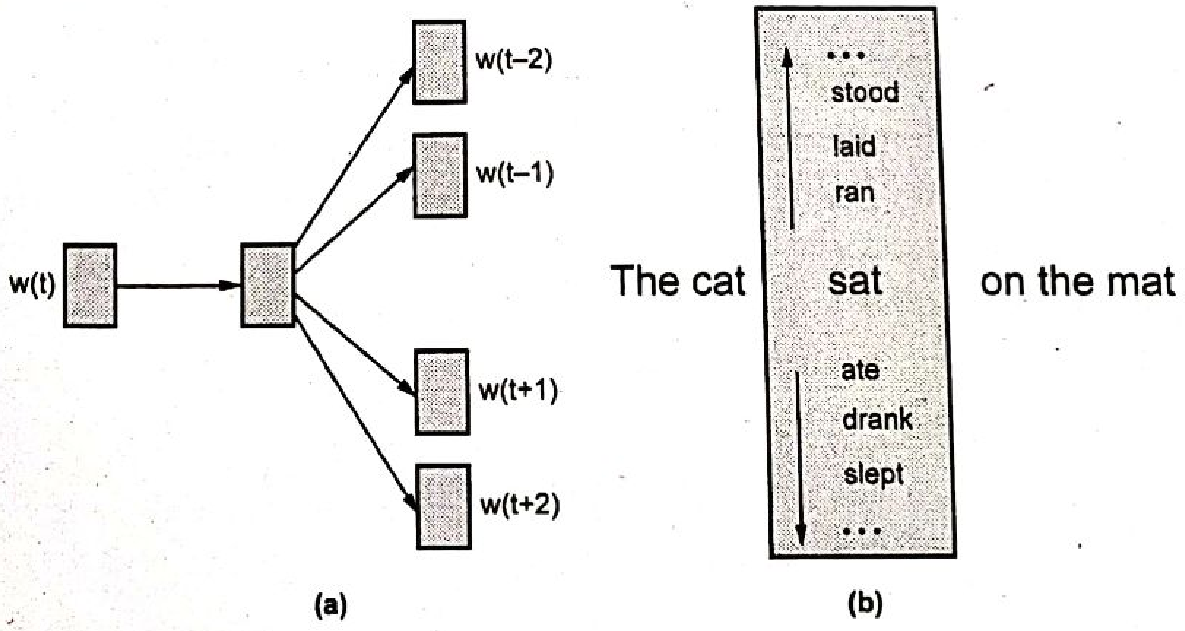


Fig. 3.2.1

Skip-gram example

The word sat will be given and we will try to predict words cat, mat at position -1 and 3 respectively given sat is at position 0. We do not predict common or stop words such as 'the'.

3.3 Continuous Bag of Words (CBOW) and Skip-Gram Model

- Word embedding is the popular representation of document vocabulary.
- Word embedding captures context, semantic and syntactic similarities of words in a document.
- They are vector representations of a particular word.
- In 2013 Google developed a technique Word2Vec to learn word embeddings using shallow neural networks.
- Word2vec is considered one of the biggest breakthroughs in the development of natural language processing.
- It is easy to understand and use.
- Word2vec is basically a word embedding technique that is used to convert the words in the dataset to vectors so that the machine understands.
- Each unique word in your data is assigned to a vector and these vectors vary in dimensions depending on the length of the word.
- Lets understand how the count vectors are generated :

Count Vector

- Let's consider
  - C is a corpus of D documents $\{d_1, d_2 \ldots .. d_D\}$.
  - N = Unique extracted tokens from C which will form our dictionary.
  - M = Count vector matrix with size D × N, where each row indicates frequency of tokens in document D(i).
- Consider the following example.
  - D_1 : He is a happy boy. She is also happy.
  - D_2 : Neeraj is a happy person.
- The dictionary created may be a list of unique tokens (words) in the corpus = ['He', 'She', 'happy', 'boy', 'Neeraj', 'person']

  Here, D = 2, N = 6

- The count matrix M of size 2 × 6 will be represented as -

	He	She	Happy	Boy	Neeraj	Person
D_1	1	1	2	1	0	0
D_2	0	0	1	0	1	1

- In the above example rows = Documents in corpus, columns = Tokens in dictionary.
- If we consider a column as a word vector, the word vector for happy can be written as [2,1]
- Some constraints and variations in this technique include :
  1. In real world applications, generally the corpus may contain millions of documents, having a huge number of unique words. The matrix generated for such applications will have a sparse structure, hence not suitable for any computation. The solution to this problem can be that instead of using every unique word as a dictionary element, only the top 10,000 most frequent words can be considered to prepare a dictionary.
  2. Either frequency of occurrence of a word in the document or the presence of the word in the document should be considered as the entry in count matrix M. Generally the frequency method is preferred.
- The word2vec model has two different architectures to create the word embeddings. They are :
  1. Continuous Bag Of Words (CBOW)
  2. Skip-gram model
- Given a set of sentences (also called corpus) the model loops on the words of each sentence and either tries to use the current word of to predict its neighbors (its context), in which case the method is called "Skip-Gram" or it uses each of these contexts to predict the current word, in which case the method is called "Continuous Bag Of Words" (CBOW).

Continuous Bag of Words (CBOW)

- The CBOW model tries to understand the context of the words and takes this as an input.
- It then tries to predict words that are contextually accurate.
- Let us consider an example for understanding this. Consider the sentence : 'It is a pleasant day' and the word 'pleasant' goes as input to the **neural network**. We are trying to predict the word 'day' here.

- The model architecture is shown below :

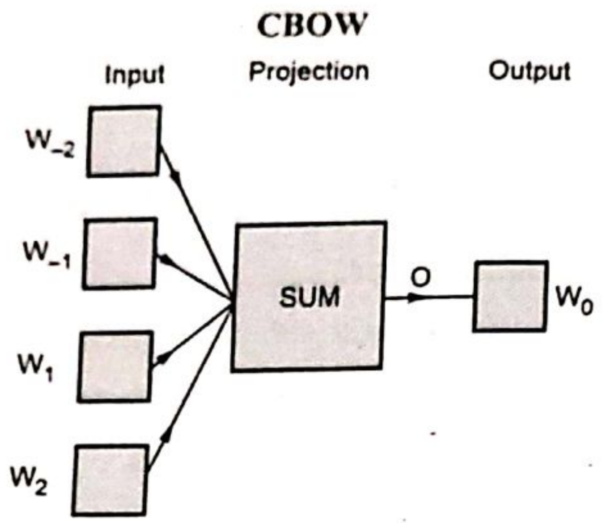


Fig. 3.3.1 The model architecture

- The model tries to predict the target word by trying to understand the context of the surrounding words.
- Consider the same sentence as above, 'It is a pleasant day'. The model converts this sentence into word pairs in the form (contextword, targetword). The user will have to set the window size. If the window for the context word is 2 then the word pairs would look like this: ([it, a], is), ([is, pleasant], a),([a, day], pleasant).
- With these word pairs, the model tries to predict the target word considered the context words.

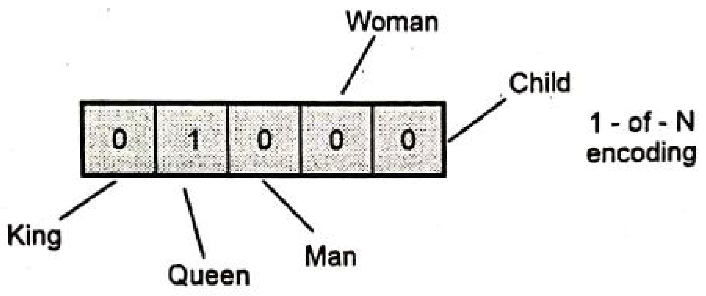
3.4 Embedding Representations for Words : Concept of Word Embedding

- As described in the BOW model we have seen how a word in a document can be represented in numerical form as a vector representation.
- In other words this numerical representation of words in called as word embedding.
- We can consider the analogy of RGB representation of colours to understand this concept.
- Section 3.1 illustrates how vector representation can be obtained from sentences in a document.
- Word embedding are classified in two types :
  1. Frequency based embedding
  2. Prediction based embedding

1. Simple frequency based embeddings

One-hot encoding

- Let's start by looking at the simplest way of converting text into a vector.
- We could have a vector of the size of our vocabulary where each element in the vector corresponds to a word from the vocabulary.
- The encoding of a given word would then simply be the vector in which the corresponding element is set to 1 and all other elements are 0.
- Suppose our vocabulary has only five words : King, Queen, Man, Woman and Child. We could encode the word 'Queen' as :



- This simple technique of creating numerical representations is called **One-Hot Encoding.**
- But this is neither scalable and nor smart.
- If we have a vocabulary of ten million words, it would mean that we would need vectors of size 10M, where each word would be represented by a single 1 and all other 9,999,999 elements would be set to 0.
- Besides the sparseness and huge memory requirement issues, this technique is also not able to capture any semantic relationships or context information.
- **Tf-Idf** is another popular technique for computing frequency based embeddings.

2. Prediction based embeddings

- Word2Vec
- Please refer section 3.1 for word2vec model.

3.5 Lexical Semantics

- Word is the smallest entity of any document.
- To know the similarity of two words, context in which they appear plays a major role.
- Similarity of context plays important role in finding out semantic similarity.

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

3.6 Word Sense Disambiguation

- To understand the concept of word sense disambiguation let's consider following example.
  1. I went to bank to withdraw money.
  2. I went to bank to fetch water.
- In the above example the same word 'bank' has appeared in two different senses and contexts.
- In computational lexical semantics it is required to examine the contextual word tokens and to figure out which sense of word is used.
- This task is known as Word Sense Disambiguation (WSD).
- WSD is the task of selecting the correct sense for a word.
- WSD is essential part of many important NLP applications like question answering, information retrieval and text classification where the use of wrong senses of words can create disasters.
- Typically any WSD algorithms, input is a word in context and fixed inventory of possible word senses. The output is a correct word sense.

- The task for which WSD is done decides nature of input and inventory of senses used.
- For example : If the task is of machine translation from English to Spanish, sense tag inventory for an English word can be set of Spanish translations. But if the task is automatic indexing of medical articles inventory can be MeSH (Medical Subject Headings) thesaurus entries.

3.7 Supervised Word Sense Disambiguation (WSD)

- The main requirement of supervised WSD is we need labeled data.
- If we have hand labeled data with proper word senses, then using supervised WSD can be used to extract features from the text.
- These features can be used to predict the senses and a classifier can be learned for assigning correct senses from the features.
- These are two broad steps in supervised WSD :
  1. Creative extraction for supervised learning.
  2. Training a classifier.

1. Feature extraction for supervised learning :

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

a) Collocational features :

- Collocation refers to a group of words that often go together or occur together.
- Some examples of collocation arc
  - To feel free → please feel free to take a seat
  - To deposit a check → I would like to deposit this check
  - Deeply regret the loss of someone → I deeply target the loss of your loved one.

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

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

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

b) Bag-Of-Words :

- Concept of bag of words is explained in detail in section 3.1.
- In WSD task using BOW concept a target word is kept as center and the context region surrounding the target word is small, symmetric, fixed size window.
- Stop words are not used as features and the vector is limited to BOW considering small number of frequently occurring words.
- Again consider the same bass example in BOW vector with 12 most frequent words feature set in written as [fishing, big, sound, player, fly, rod, pound, double, runs, playing, guitar, band] with window size 10 the above feature set is written as binary vector

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

2. Training a classifier :

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

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

- To understand formulation of Naive Bayes classifier, let's consider BOW vector for 20 words having 2^{20} possible feature vectors.

- To solve this problem, first the problem in reformulated in Bayesian manner as

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

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

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

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

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

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

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

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

2) Decision list classifier :

- In this, a sequence of tests is applied to each target word feature vector.
- A specific sense is shown by each test.
- A sense is returned with a successful test and in case of failure the next test is applied till end of list.
- Consider the disambiguation of the sense 'bass' from Fig. 3.7.1.
- As shown in Fig. 3.7.1 the first test indicates that if word fish occurs in input context then bass in correct.

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

Fig. 3.7.1 An abbreviated decision list for disambiguating the fish sense of bass from the music sense (Yarowsky, 1997)

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

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

Review Questions

1. Explain the concept of bag of words.
2. Explain skip gram technique.
3. What is continuous bag of words ?
4. What are word embeddings ?
5. Explain lexical semantics.
6. What is word sense disambiguation ?
7. What is supervised word sense disambiguation ?

Chapter 4
Text Analysis, Summarization and Extraction

Syllabus

Sentiment Mining, Text Classification, Text Summarization, Information Extraction, Named Entity Recognition, Relation Extraction, Question Answering in Multilingual Setting; NLP in Information Retrieval, Cross-Lingual IR.

Contents

4.1 Sentiment Mining

- Knowing sentiment is a very natural ability of a human being.
- Sentiment analysis (or opinion mining) is a natural language processing technique used to determine whether data is positive, negative or neutral.
- Sentiment analysis aims at getting sentiment-related knowledge especially from the huge amount of information on the internet.
- It can be generally used to understand opinion in a set of documents to help various applications like customer feedback, movie reviews, etc.
- Sentiment analysis systems focus on polarity (positive, negative, neutral), feelings and emotions (angry, happy, sad, etc), urgency (urgent, not urgent) and even intentions (interested vs. not interested).

Some popular types of sentiment analysis are :

1. Fine-grained sentiment analysis :

- If polarity precision is important in an application, we may consider expanding polarity categories to include :
  - o Very positive
  - o Positive
  - o Neutral
  - o Negative
  - o Very negative.

  This is usually referred to as fine-grained sentiment analysis, and could be used to interpret 5-star ratings in a review, for example :

  Very positive = 5 stars

  Very negative = 1 star

2. Emotion detection :

- This type of sentiment analysis aims to detect emotions, like happiness, frustration, anger, sadness, and so on. Many emotion detection systems use lexicons (i.e. lists of words and the emotions they convey) or complex machine learning algorithms.

3. Aspect-based sentiment analysis :

- To understand aspect based sentiment analysis let's consider the following example : Let's consider the sentence. "The battery life of this camera is too short". In this example we are focussing on a particular feature or aspect of a product. So an aspect-based classifier would be able to determine that the sentence expresses a negative opinion about the feature battery life.

4. **Multilingual sentiment analysis :**
   - It is a difficult task as it involves lot's of preprocessing. Many resources are also involved.
   - This can be done by making use of online sentiment lexicons or by creating translated corpora or noise detection algorithms.

Sentiment classification can be done by various techniques as shown in the Fig. 4.1.1 below :

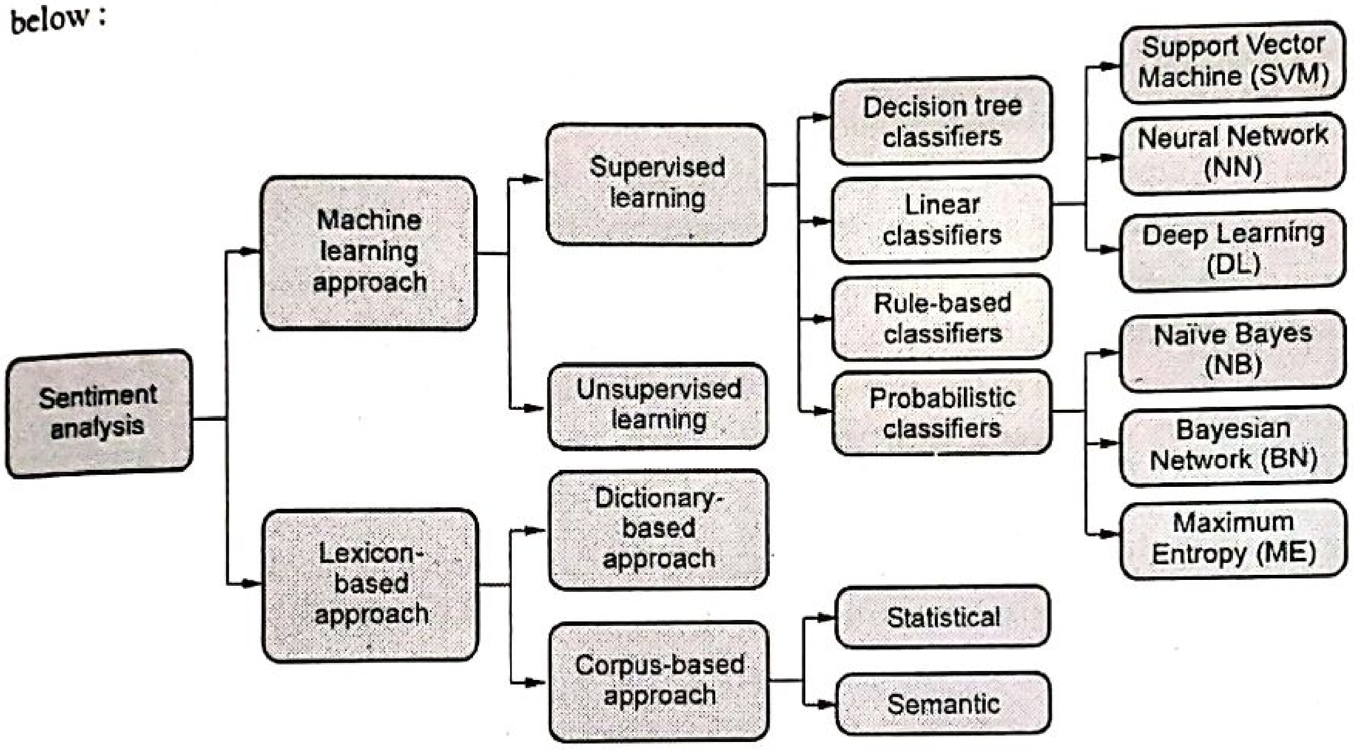


Fig. 4.1.1 Sentiment classification techniques

4.2 Text Summarization

- Any text document consists of some sentences containing the crux of the document and are contextually important and remaining are the supporting sentences.
- Not all the sentences in any document are contextually useful.
- Text summarization is an important task of NLP, which is "process of disttilling the most important information from a text to produce an abridged version for a particular task and user," according to the definition by Mani and May bury.
- Some important summaries include abstracts of scientific article, headlines of news articles, legal document summaries, summaries of medical documents etc.
- The different dimensions of summarization tasks include :

  1. **Single document versus multiple document summarization :**
     - In the first type the summarization goal is restricted to single document like headline generales whereas multiple document summarization focuses on summary

generation of an entire group of documents, like summarization of series of news stories of same event.

2. **Generic summarization vs query focused summarization :**
   - In generic summarization a particular user or need is not considered. This summary contains only important information from the documents.
   - Whereas in query focused or focused or topic based or user focused summarization the summary is a result of user query.

- There are two ways to address the task of text summarization,
  1. **Extractive summarization :** In this the key sentences or key phrases from the original document appears in the summary without any alteration. The knowledge of NL understanding is sufficient for extractive summarization.
  2. **Abstractive summarization :** In this approach the summary contains newly generated sentences not identical to the sentences present in original document. This is a difficult problem and needs knowledge of Natural Language Understanding (NLU) as well as Natural Language Generation (NLG) tasks.

Single document summarization :

- Extractive summary for a single document can be written using three stages :
  1. **Content selection :** Locating and finding the candidate sentences from the document for the summary.
  2. **Information ordering :** Find out the order in which these candidate sentences should appear in the summary.
  3. **Sentence realization :** Clean these sentences by removing non-essential phrases from each sentence, combining multiple sentences to one and resolving coherence problems.
- This process is shown in Fig. 4.2.1.

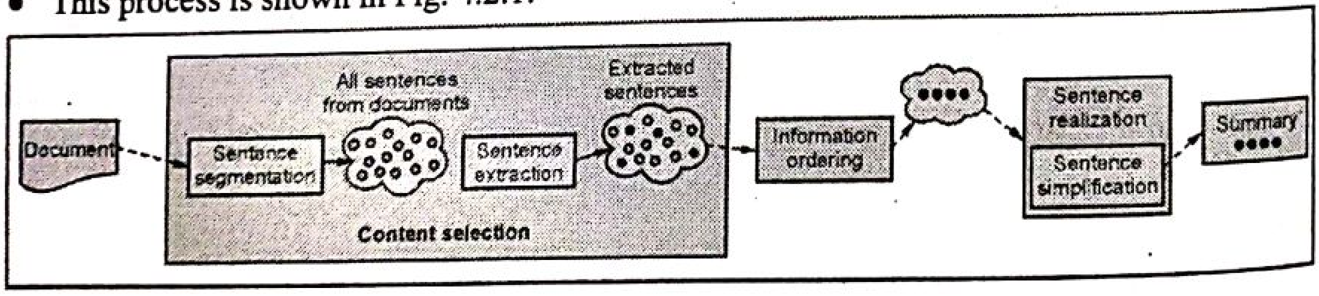


Fig. 4.2.1 The basic architecture of a generic single-document summarizer

Unsupervised content selection :

- In content selection phase the classifier labels each sentence in an input document by binary label : Important vs unimportant.

- In unsupervised approach the algorithm chooses more salient or informative words by computing topic signature which is a set of signature or salient terms having saliency score greater than some threshold θ.
- Weighting schemes like tf-idf or log likelihood ratio are used for measuring saliency.

$$\text{Weight}(w_i) = tf_{i,j} + idf_i$$

where

$i \rightarrow$ Each term i occurring in the sentence to be evaluated.

$j \rightarrow$ Current document

$tf_{i,j} \rightarrow$ Count of i in j

$idf_i \rightarrow$ Inverse document frequency

- In Log - Likelihood Ratio (LLR) the weight can be calculated as,

$$\text{Weight}(w_i) = \begin{cases} 1 & \text{if} -2\log(\lambda(w_i)) > 10 \\ 0 & \text{otherwise} \end{cases}$$

where the quantity $-2 \log (\lambda)$ is asymptotically approximated by x^2 distribution indicating that word appears in the input significantly more often than background corpus.

- The above equation assigns weight of 1 or 0 to each word.
- The score of sentence s_i = Average weight of its non-stop words i.e.

$$\text{Weight}(s_i) = \sum_{w \in s_i} \frac{\text{Weight}(w)}{|\{w | w \in s_i\}|}$$

- LLR algorithm falls in centroid based summarization.

Unsupervised summarization based on Rhetorical parsing :

- The above algorithm depends only on a single shallow feature and does not focus on the high level sentence features like discourse.
- The summarization algorithm based on coherence relations like Rhetorical Structure Theory (RST) uses the concept of a satellite and a nucleus.
- Typically a RST JUSTIFICATION relation in which first discourse unit justify the second is used.
- The example is shown in Fig. 4.2.2.

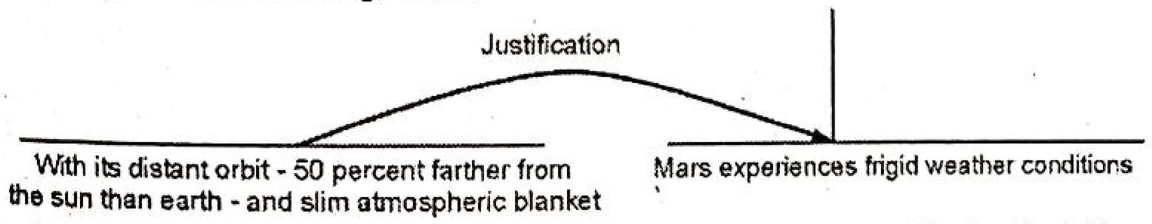

Justification
With its distant orbit - 50 percent farther from the sun than earth - and slim atmospheric blanket
Mars experiences frigid weather conditions

Fig. 4.2.2 The justification relation between two discourse units, a satellite (on the left) and a nucleus (on the right)

- This can be achieved by using a discourse parser to compute coherence relations in the first step.
- In the next step parse tree is generated and nuclear units can be used for summary.

Supervised content selection :

- The supervised machine learning a training set of documents is used in pairing with human created summary extracts.
- Each sentence is given label 1 if it appears in an extract and 0 if it is not.
- The classifier is built by choosing the features which predict that the sentences are good.
- The probabilistic classifiers like naive Bayes or MaxEnt computes the probability that sentence s is good for extraction from set of features $f_1, \ldots f_n$.
- Any sentences having probability > 0.5 is chosen for extraction.
- The probability can be given as,
  P (extract-worthy (s) | $f_1, f_2, f_3, \ldots. f_n$)
- The drawback of this algorithm is for each document a training summary is needed containing only extracted sentences.

Sentence simplification :

- After extraction and ordering of the sentences the final step in summarization is sentence realization.
- Sentence realization can be done by sentence simplification.
- Following example shows how sentences can be simplified.

Original sentence : The V-chip will give parents a device to block out programs they do not want their children to see.

Simplified sentence by humans : The V-chip will give parents a device to block out programs they do not want their children to see.

The simplest algorithms for sentence simplification use rules to select parts of the sentence to prune or keep, often by running a parser of partial parser over the sentences. Some representative rules from Zaijic et al. (2007), Conroy et al. (2006) and Vanderwende et al. (2007) remove the following :

appositives	Rajam, found the inspiration in the back of city magazines.
attribution clauses	Rebels agreed to talks with government officials.
PPs without named entities	The commercial fishing restrictions in Washington will not be lifted [SBAR unless the salmon population 329 increases [PP].
initial adverbials	"For example", "On the other hand", "As a matter of fact", "At this point".

- The algorithm for sentence selection uses rules to select parts of the sentence to keep.
- The parser is used for this purpose.

4.3 Information Extraction

4.3.1 Named Entity Recognition

- Anything which is referred by a proper name is called as **named entity.**
- Table 4.3.1 shows entity types and their categories.

Type	Tag	Sample categories
People	PER	Individuals, fictional characters, small groups.
Organization	ORG	Companies, agencies, political parties, religious groups, sports teams.
Location	LOC	Physical extents, mountains, lakes, seas.
Geo-political entity	GPE	Countries, states, provinces, counties.
Facility	FAC	Bridges, buildings, airports.
Vehicles	VEH	Planes, trains, and automobiles.

Table 4.3.1 A list of generic named entity types with the kinds of entities they refer to

- Named Entity Recognition (NER) is a process of finding the part of text containing proper names and to identify and classify the entitie's types.
- NER systems take into account various entity types like people, places, organizations, commercial products, weapons, biological entities and many more.
- Sometimes the proper names are signaled based on their surrounding contexts.

  For example :

  If in a text word Dr. appears then we can say that the next word is a name of a person.
- Concept of named entity is further extended to the entities of practical importance known as temporal, for example, dates, expressions, times, named events, etc. Some represent numerical expressions like measurements, counts, prices etc.

Ambiguity in named entity recognition :

- Two types of ambiguities exist in NER.

  1] The ambiguities which arise due to reference resolution problem :

  - In some cases it may happen that same name can be used for different entities.
  - **For example :** JFK can be used to address former president or his son.

2] The ambiguities which arise due to cross type confusion :

- It may happen that same name can refer to a person as well as some school name or airport name.
- **For example :** In extension with previous example along with name of person JFK can also refer to a school name or airport name.

NER as sequence labeling :

- To address the problem of NER word by word sequence labeling task is used.
- In this approach the assigned tags can capture both the boundary and type of detected entities.
- The classifiers are trained for labelling the tokens in a text with tags. These tags indicate the presence of specific kinds of named entities.

Evaluation of named entity recognition :

- Recall, precision and F, measures matrices are used for evaluation of NER.

where

Recall ratio of the number of correctly labelled responses to total responses eligible for labelling

Precision = Ratio of the number of correctly labelled responses to total labelled

F measure = Way to combine above two measures into single matrix.

So, $$F_\beta = \frac{(\beta^2+1)\,PR}{\beta^2 P + R}$$

Where $\beta \rightarrow$ Weights the importance of recall of precision.

$\beta > 1$ Favor recall

$\beta < 1$ Favor precision

$\beta = 1$ Recall and precision are balanced

In this case f can be written as $f_1 = \frac{2\,PR}{P+R}$

4.3.2 Relation Extraction (Relation Detection and Classification)

- To find out the relationships between the entities in the text is a very crucial task.
- To understand this let's consider the example :

$[\text{ORG}^{\text{Indian Airlines}}]$, a unit of $[\text{ORG}^{\text{ABC corp.}}]$ increased fares by $[\text{MONEY}^{5000 ₹}]$ on $[\text{TIME}^{\text{Friday}}]$.

Now from this example we can easily figure out that Indian Airlines is unit of ABC corp.

- There are many such relations exists like 'part of' or 'employs' in the text.
- Some of such semantic relationships and their NER types are mentioned in Table 4.3.2.

Relations		Examples	Types
Affiliations			
	Personal	married to, mother of	PER → PER
	Organizational	spokesman for, president of	PER → ORG
	Artifactual	owns, invented, produces	(PER|ORG) → ART
Geospatial			
	Proximity	near, on outskirts	LOC → LOC
	Directional	southeast of	LOC → LOC
Part-Of			
	Organizational	a unit of, parent of	ORG → ORG
	Political	annexed, acquired	GPE → GPE

Table 4.3.2 Semantic relations with examples and the named entity types they involve

- To detect and classify these relations is the process called as **relation extraction.**
- There are two approaches by which relation analysis can be done :
  1. Supervised learning approaches to relation analysis.
  2. Lightly supervised approaches to relation analysis.

1. Supervised learning approaches to relation analysis :

- In supervised learning approaches for relation detection and classification, in the first step first human analysis annotate the text and choose the relation from fixed small set.
- These text pieces are then used to train the system to generate similar annotation on unknown texts.
- In this approach initially the problem is divided in two subtasks,
  1. Relation detection between two entities
  2. Classification of detected relation.
- In the first phase training of classifiers is done to decide if given pair of named entities participate in the relation or not.

- Positive example extraction is done from annotated corpus and negative examples are generated by using non-annotated sentence entity pairs.
- In second phase training of classifier is done for labelling the relations between candidate entity pairs.
- Techniques like decision trees, naive Bayes, MaxEnt etc. are used for labelling task.
- Set of classifiers are trained, one for each label as a positive class and others labels are classified as negative class.
- In the final phase each instance for which labelling is to be done is passed to all classifiers and then lable is chosen with classifier having most confidence.
- The approach is stated in Fig. 4.3.3.

```
function FINDRELATIONS(word) returns relations
    relations ← nil
    entities ← FINDENTITIES(words)
    forall entity pairs ⟨e1, e2⟩ in entities do
        if RELATED?(e1, e2)
            relations ← relations + CLASSIFYRELATION(e1, e2)
```

Fig. 4.3.3 Finding and classifying the relations among entities in a text

2. **Lightly supervised approaches to relation analysis :**
   - In this approach regular expression patterns are used to match text segments which have probability of containing expressions of the relations which are of our interest.
   - Lets understand this with the help of example, consider that the search engine is closing phrasal search.
   - Now if we want to list all the hub cities which are utilized by previous airlines then we can enter something like /* has a hub at */as a query.
   - It is observed that if sufficient material is accessible then we get fairly good amount of correct answers.
   - **For example :** 1. Indigo has a hub at Mumbai.
     2. Jet airways has a hub at Kolkata. etc.
   - But this approach may fail sometimes and give the result as,
     1. Airline j has a hub at airport k.
     2. A star topology often has a hub at its center etc.

- These features can be eliminated by replacing unrestricted operator with named entity class restriction for example :

  /[ORG] has a hub at [LOC]/

- There is also possibility that some relevant results are not returned due to our right specific pattern of phrasal search.

  For example : Indigo also has a continental hub at Andaman.

- This problem can be eliminated by two ways :

  1. To generalize the pattern to capture the expressions containing the information of our interest.
  2. To expand our set of specific high precision patterns.

     For example : If we have found that Indigo has a hub at Andaman, then this information can be used to discover new patterns in our corpus. This can be done by finding various mentions of this relation in our corpus, i.e. searching can be carried out for the terms Indigo, Andaman and hub in some proximity.

4.4 Question Answering in Multilingual Setting

- The Question Answering (QA) system automatically provides an answer to the user's query.
- To do this task the system needs access to the database or knowledge base.
- However it is observed that most QA systems rely on information obtained from various collections of natural language documents which are in most cases unstructured, e.g. collection of reference texts, news reports, world wide web pages, etc.
- Multilingual question answering is a sub-field of question answering or QA.
- It blends the field of information retrieval and natural language processing.
- It focuses on creating systems that do not only answer questions but also allow the user to access information in other languages than that used for asking questions.
- However, several multilingual QA systems have been built in the last few years, all seeking to overcome not only the challenges of monolingual QA but also to overcome the issue of accessing and retrieving information in multiple languages.
- The majority, however, are based on translating relevant sections of the question - usually with the aid of machine translation system - which are then used to access to a collection containing relevant information.
- Most commonly used multilingual QA systems also rely heavily on the monolingual system to answer the question.

- As shown in Fig. 4.4.1 the question answering system has three stages,

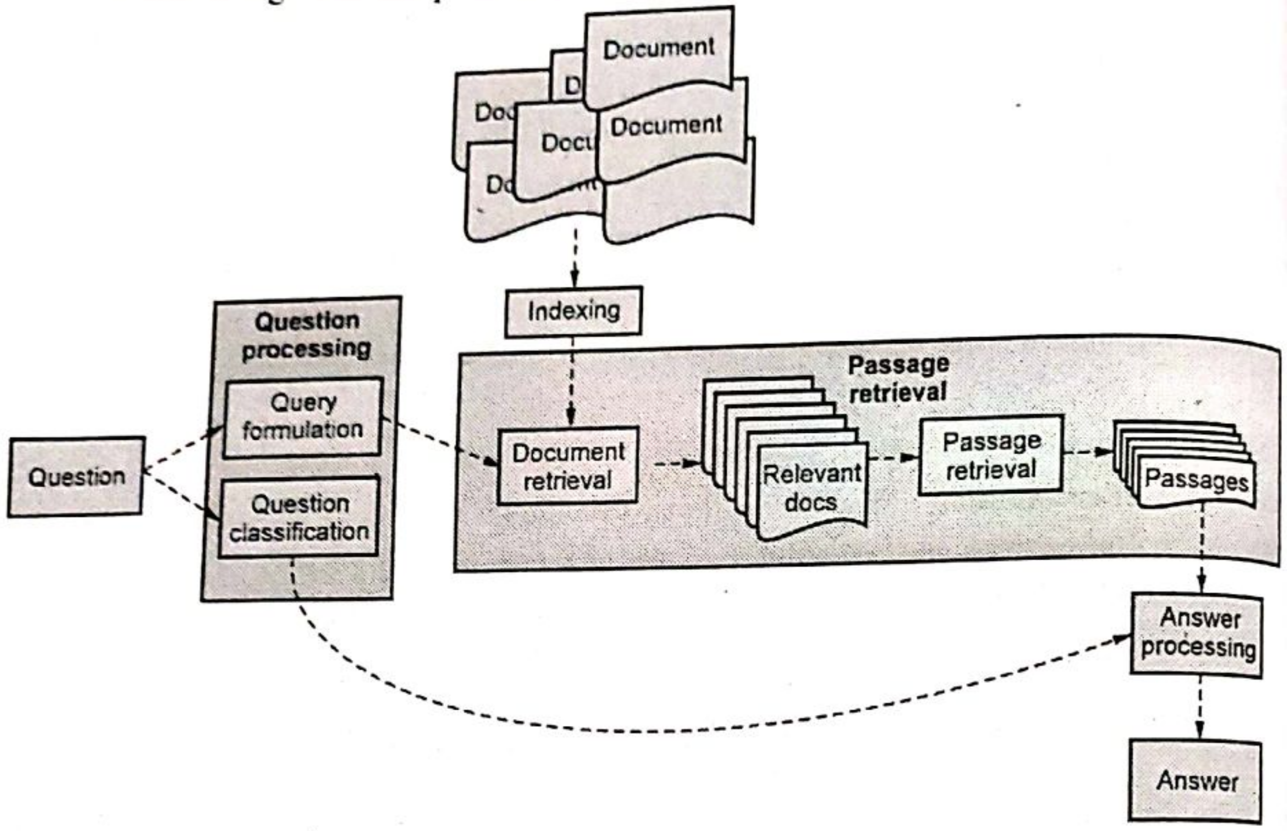


Fig. 4.4.1 Question answering has three stages : Question processing, passage retrieval and answer processing

1] Question processing

- In this phase two things are extracted from the question :
  - o A keyword query which will be appropriate as an input to the IR system.
  - o An answer type, a specification which will act as a reasonable answer.

2] Passage retrieval

- The generated query from the previous phase is then used in the information retrieval system.
- IR system can be,
  - o General IR engine over proprietary set of indexed documents or
  - o Web search engine
  - o Next, the set of potential answer passages is extracted from the set of potential answer passages.
  - o These passages are filtered and ranked based on how likely they are to contain answers to the question.
  - o This ranking is done either through hand crafted rules or by supervised training with machine learning techniques.

3] Answer processing

- In this phase a specific answer is retrieved from the passage.
- This answer extraction can be done by using two classes of algorithm : One based on answer type pattern extraction and another on N gram tiling.

4.5 NLP in Information Retrieval

- Information retrieval field mainly contain two broad tasks, first is storage and second is retrieval of all manner of media.
- IR task focuses on storage of text documents and their retrieval according to user's request.
- With IR context, a word in document refer to unit of text in the system, which is indexed and available for retrieval.
- Documents can be of different types for example, newspaper, articles, encyclopedia entries or simply small paragraphs and sentences.
- Some terminologies used in IR systems are,

  collection → Set of documents to satisfy user queries.

  term → Lexical item in collection.

  query → User's information need in set of terms.
- The architecture of Ad hoc IR system is shown in Fig. 4.5.1.

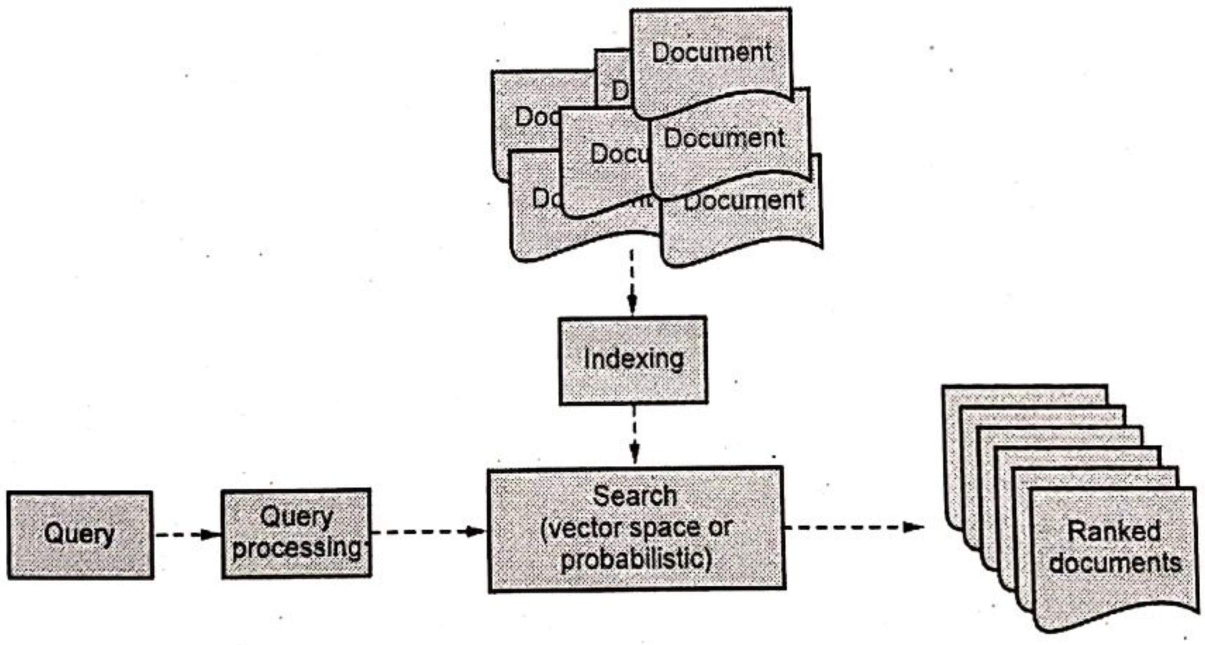


Fig. 4.5.1 The architecture of an Ad hoc IR system

The vector space model of information retrieval :

- In this the documents and queries are represented as vectors.
- Vector consists of features which represent the words present in the collection.

- Consider the example of Potato fries out recipe on internet oil in this recipe if the term Potato, fires and salt occur term frequencies 8, 2, 7 and 4 respectively then the vector ca be written as

$$\vec{d}_j = (8, 2, 7, 4)$$

- So in general for document d_j, a vector is written as,

$$\vec{d}_j = (w_{1,j}, w_{2,j}, w_{3,j}, \dots, w_{n,j})$$

where $\vec{d}_j \rightarrow$ A document

$w_{n,j} \rightarrow$ Weight that term n in document j.

- Query can be written in same way

so query q for Potato fries can be written as,

$$\vec{q} = (1, 1, 0, 0)$$

- In general it is written as,

$$\vec{q} = (w_{1,q}, w_{2,q}, w_{3,q}, \dots, w_{n,q})$$

N $\rightarrow$ Dimensions in vector = Total number of terms in complete collection.

- Now consider another document recipe for Potato curry. Let's consider the vector as,

$$\vec{d}_k = (6, 0, 0, 0)$$

- If the query is written for Potato fries then it should match document d_j then to d_k.
- If we use features and queries representing a document as dimensions in multidimension space then feature weights are used to locate documents in that particular space.
- A point in this space represents user's query translated into vector.
- This is shown graphically in Fig. 4.5.2.

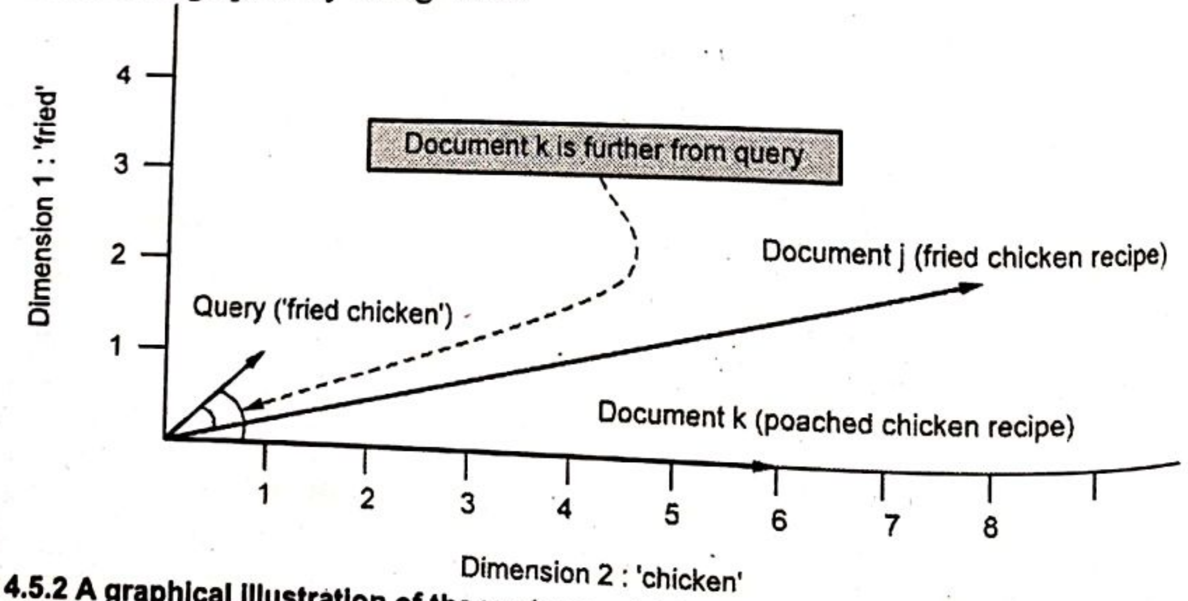


Fig. 4.5.2 A graphical illustration of the vector model for information retrieval, showing the first two dimensions (fried and chicken) and assuming that we use raw frequency in the document as the feature weights

- The similarity between vectors is measured by angle between the vectors.
- As shown in the Fig. 4.5.2 the query is considered more similar to d_j than to d_k as the angle between query and d_j is smaller.
- Typically a cosine metric is used in this type of retrieval.
- The distance between two documents is measured by cosine of the angle between them.
- For identical document cosine value is 1.
- If the documents don't share common terms, the value of cosine is 0.
- The equation for cosine is written as,

$$S_{lm}(\vec{q}, \vec{d_j}) = \frac{\left(\sum_{i=1}^{N} w_{i,q} \times w_{i,j}\right)}{\sqrt{\sum_{i=1}^{N} w_{i,q}^2} \times \sqrt{\sum_{i=1}^{N} w_{i,j}^2}}$$

- Cosine can also be considered as normalized dot product i.e. it is a dot product between two vectors divided by lengths of two vectors.
- The numerator of cosine is dot product given by formula,

$$\text{Dot - product } (\vec{x}, \vec{y}) = \vec{x} \cdot \vec{y} = \sum_{i=1}^{N} x:\times y:$$

- The denominator of cosine is represented by lengths of vector, where vector length is given as,

$$|\vec{x}| = \sqrt{\sum_{i=1}^{N} x_i^2}$$

- So a document retrieval system simply accepts user's query, creates vector representation, compare it with vector representing all the documents and finally the result is sorted, which is list of ranked documents according to their similarity with query.

Evaluation of IR systems :

- The basic tools for measuring performance IR system are,

  1. Precision 2. Recall.
- It is assumed that returned item is divided in two categories : The relevant results and irrelevant results.
- So precision can be stated as fraction of returned relevant documents.
- Recall is fraction of all possible relevant documents contained in return set.

- Let's consider,

  T → Total ranked documents as a result of given query

  R → Relevant documents from T

  N → Remaining irrelevant documents

  U → Relevant documents in the collection as a whole for given query.

  Now the precision and recall measures can be given as,

$$\text{Friction} = \frac{|R|}{|T|}$$

$$\text{Recall} = \frac{|R|}{|U|}$$

- The drawback is, these measures are not sufficient to measure the performance of the ranked retrival of the system as these are not dependent on rank.

4.6 Cross Lingual IR

- Humans have the unique ability to express themselves through different mediums of communication like : Oral, written, through images and pictures, through gestures and sign language, etc.
- Out of these, oral and written communication in natural language facilitates the fastest way of conveying the ideas, so they are very popular.
- Written communication is advantageous as it preserves the information manually as well as electronically.
- Around 7000 languages exist in the world, out of which very few are used across the globe.
- English language has the greatest internet presence, even though it is not the most spoken language in the world.
- With the advancements in technology, a humongous amount of e-text is getting generated across the world.
- The e-text may contain :
  1. Documents in different languages
  2. Multilingual documents
  3. Images with captions in different languages.
- People with linguistic diversity have started using their mother tongue for searching the documents present in various domains and languages.
- While using search engines, query based information retrieval is a common way.

- With this context in simple words Cross-Lingual Information Retrieval (CLIR) is retrieving information in one language based on the query written in another language.
- The users can search document databases in multiple languages and retrieve information in a form that is useful to them, even though they don't have linguistic competence in the target languages.
- CLIR is important for countries like India where a very large fraction of people are not conversant with English and thus don't have access to the vast store of information on the web.
- Searching distributed, unstructured, heterogeneous, multilingual data is the goal of CLIR system.

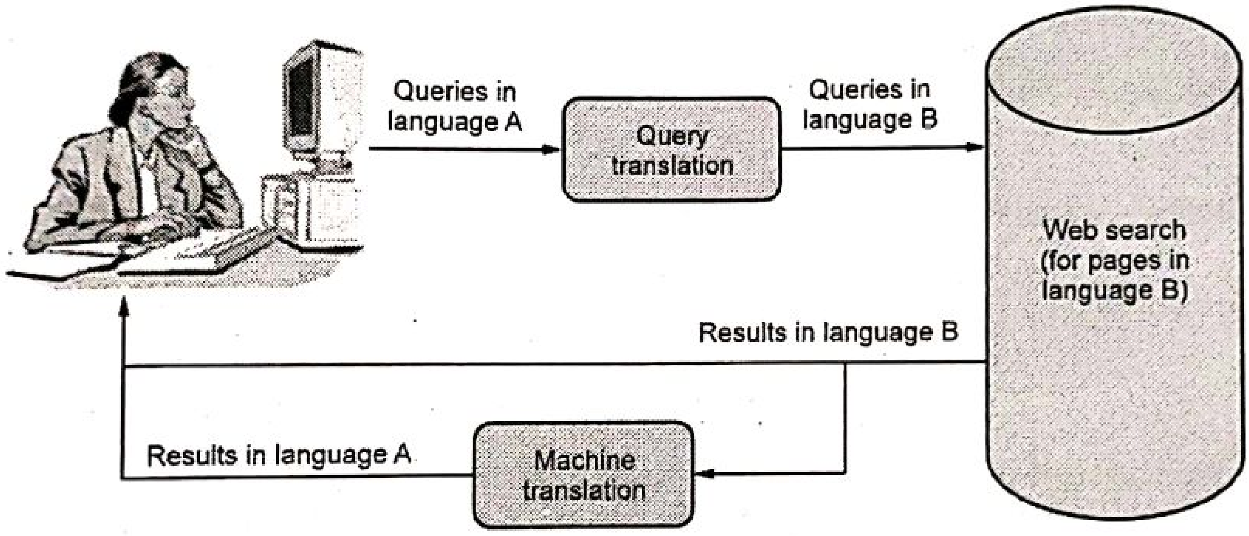


Fig. 4.6.1 CLIR system

- As shown in the Fig. 4.6.1, a user can enter a query in any language. For example, Hindi. If the relevant information is present in language B (For example English), the query is first translated to language B. After the web search the retrieved information in language B is again translated to language A, and results are displayed to the user.

Different approaches of CLIR are :

1. **Query translation approach :**
   - As shown in the Fig. 4.6.2, the query is translated to the target document language.
   - This is the most appropriate approach, as the query is shorter and fast to translate than a complete document.
   - One disadvantage can be, query translation can suffer from translation ambiguity due to the limited context.

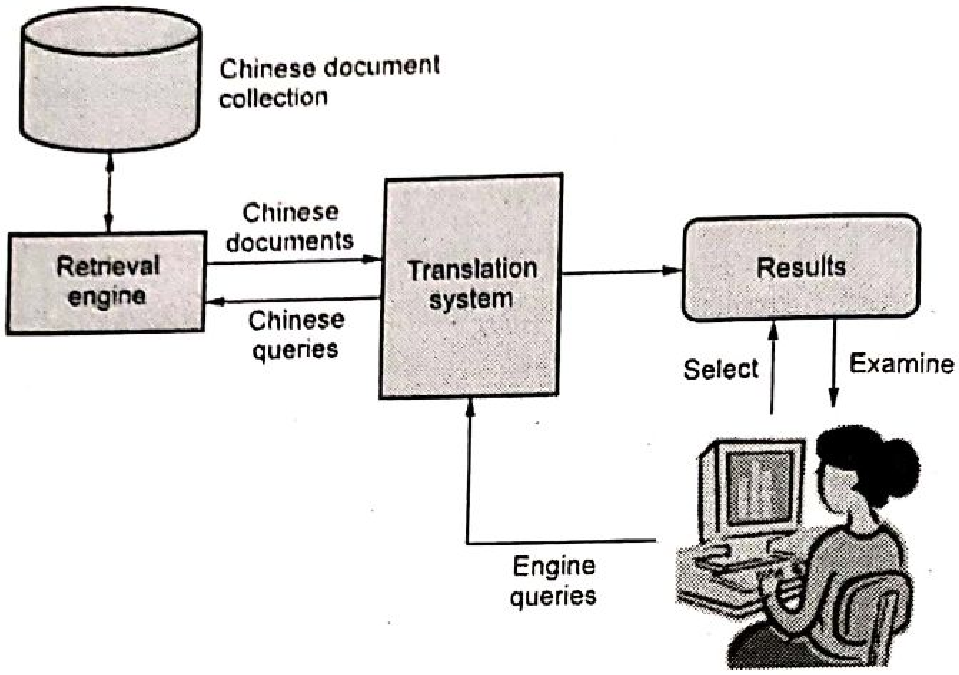


Fig. 4.6.2

2. Document translation approach :

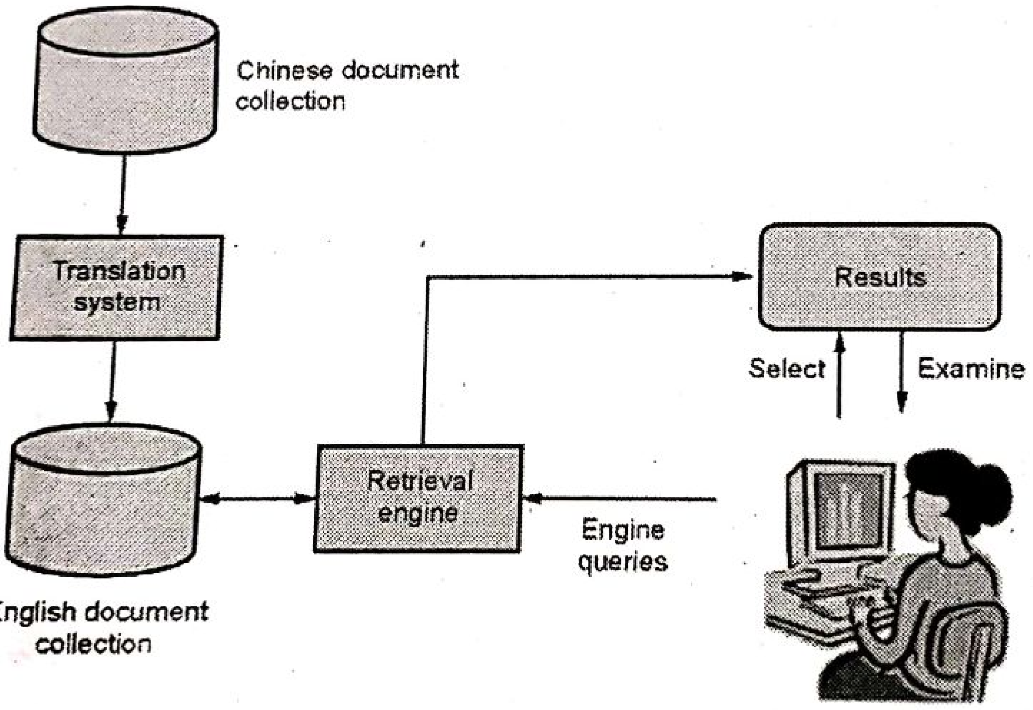


Fig. 4.6.3

- As shown in the Fig. 4.6.3, the documents in target language are translated to source language.
- Document translation can provide more accurate translation due to richer contexts.
- Advantage of document translation is, it's easy for the user to understand the document easily once it is retrieved.

3. **Interlingua based approach :**

- In this the documents and the query are both translated into some common third Interlingua (like UNL).
- This approach generally requires huge resources as the translation needs to be done online.

Challenges in CLIR :

1. **Translation ambiguity :**
   - Due to ambiguities in various phases of NLP, more than one translation is possible of the source sentence.
   - **For example :** Word "Pooja" can have two meanings, it can be the name of a girl or the second meaning can be worshipping.
2. **Phrase identification :**
   - In some languages like Japanese the words are not separated by white spaces. This may lead to the incorrect translation.
3. **Dictionary coverage :**
   - The linguistic resources in the dictionary can prove as a constraint in performance of the system.
   - Out-of-Vocabulary (OOV) problems.
   - Daily new words are added to the language, which may not be recognized by the system.
   - Apart from traditional CLIR systems, recently cross-lingual word embeddings and neural network based information retrieval systems are becoming increasingly popular.
   - Cross-lingual word embeddings can represent words in different languages in the same vector space by learning a mapping from monolingual embeddings.
   - Neural network based information retrieval can produce better representations for documents and queries. In this case the ranking is done directly from relevant labels.

Review Questions

1. What is fine grained sentiment analysis ?
2. Explain aspect based sentiment analysis.
3. Explain multilingual sentiment analysis.
4. What is text summarization ?
5. Explain single document summarization.

Chapter 5

Machine Translation

Syllabus

Need of MT, Problems of Machine Translation, MT Approaches, Direct Machine Translations, Rule-Based Machine Translation, Knowledge Based MT System, Statistical Machine Translation (SMT), Parameter learning in SMT (IBM models) using EM, Encoder-decoder architecture, Neural Machine Translation.

Contents

5.1 Need of Machine Translation

- According to research firm Common Sense Advisory, 72.1 percent of the consumers spend most or all of their time on sites in their own language, 72.4 percent say they would be more likely to buy a product with information in their own language and 56.2 percent say that the ability to obtain information in their own language is more important than price.
- These are just a few of the many reasons that translation has become essential in the modern and ever more globalized world that we live in.
- Machine translation is a tool that can help businesses and individuals in many ways. While machine translation is unlikely to totally replace human beings in any application where quality is really important, there are a growing number of cases that show how effective and useful machine translation can be.
- Machine translation is useful in many areas, including :
  - Highly repetitive content where productivity gains from machine translation can dramatically exceed what is possible with just using translation memories alone (e.g. automotive manuals.)
  - Content that is similar to translation memories but not exactly the same (e.g. government policy documents.)
  - Content that would not get translated otherwise due to cost, scale and volume limitations of human translations (e.g. millions of Chinese, Japanese, Korean and other language patents made available in English.)
  - High-value content that is changing every hour and every day there is time sensitivity (e.g. stock market news.)
  - Content that does not need to be perfect but just approximately understandable (e.g. any website for a quick review.)
  - Knowledge content that facilitates and enhances the global spread of critical knowledge (e.g. customer support.)
  - Content that is created to enhance and accelerate communication with global customers who prefer a self-service model (e.g. knowledgebase.)
  - Content that would normally be too expensive or too slow to translate with a human only translation approach (e.g. many projects that have an insufficient budget for a human only approach.)
  - Increasing the amount of content that can be translated within a budget (e.g. Dell has doubled the amount of content they translate without increasing their translation budget.)

- o Real-time communications where it would not be practical for a human to translate (e.g. chat and email.)

5.2 Problems of Machine Translation

- The languages can be considered similar of different based on various aspects.
- There may be syntactic differences which can be modelled in general way.
- There can be unusual use of meanings of certain words which makes sense to only those who are familiar with the situation known as idiosyncratic words and also there are lexical differences.
- All these types of differences in languages are known as translation divergences which makes task of machine translation difficult.
- Few intricacies of languages which makes the task of machine translation hard are described below :

1. Typology :

- If we listen to some foreign language speech then it is difficult to understand that, but still we can figure out some universal aspects like words referring to people, being polite also there are nouns and verbs in every language.
- Some systematic cross linguistic similarities and differences in different languages.
- This study is called as typology.

Typology includes following aspects

1. Morphological aspects :

The languages has two types of morphological variation.

a. Number of morphemes per word :

- o Some languages are isolating languages in which each word load one morpheme. For example : Vietnamese, Korean, Nihali language spoken by Nihal tribe near Tapti river in Maharashtra.
- o Some languages are polysynthetic languages in which words are composed of many morphemes. For example : Siberian Yupik ("Eskimo"), Inuktitut, American Indian Language etc.

b. Degree of segmentation of morphemes :

- o Some languages are agglutinative languages. Agglutination is a grammatical process in which words are composed of sequence of morphemes, each of which represents no more than single grammatical category.
- o In agglutinative languages morphemes have clean boundaries. For example : Languages like Turkish, Tamil, etc.

- Other type of languages are fusion languages. Unlike agglutinative languages fusion languages use single inflectional morpheme for denoting multiple grammatical, syntactic or semantic features. For example : Languages like Sanskrit, Bengali, Punjabi, Latin, French, etc.

2. Syntactic aspects :

- Typically all languages differ in the basic word order of verbs, subjects and objects in declarative clauses.
- Some languages like German, French, English, etc. SVO (Subject-Verb-Object) where verb is in between subject and object. These have prepositions.
- Some languages like Hindi, Japanese, etc. are SOV (Subject-Object-Verb) languages where verb comes at the end of clause. They have postpositions.
- Some languages like Irish, Arabic and Biblical Hebrew are VSO (Verb-Subject-Object).
- For example, in English sentence,

  He adores listening to music

  argument VP (listening to music) follows verb adores. Verb listening is followed by PP to music and preposition to is followed by its argument music.

  Whereas in Japanese language, verbs are preceded by arguments and postposition follows its argument i.e. it follows reverse order as compared with English.

  i.e.

  English : He adores listening to music.

Japanese :	Kare	ha	ongaku	wo	kiku	no ga	daisuki desu
	he		music	to	listening		adores.

3. Arguments structure and linking aspects and head marking and dependent marking languages :

- Head marking languages are the once in which relation between head and head dependents is shown.
- Whereas in dependent marking languages the relation on non head is depicted.
- The example of head marking language is Hungarian and English is a non head language.
- Some other types of languages are :
  - Verb framed languages in which dissection of melion is marked on verb. For example : Spanish, Japanese, Tamil.

- o Satellite framed languages with direction of motion on satellite. Example : English, Hindi, Chinese, Russian, etc.
- o Prop drop languages which omit pronouns. Japanese, Chinese omit the pronouns more than Spanish. This parameter of frequency of omission of pronouns is referred to as referential density.

- The languages which uses more pronouns are known as referentially dense languages. For example : Spanish.
- These are also known as hot languages and are more explicit and easy to hear. The languages like Chinese or Japanese are known as referentially sparse language. These are called as cold languages.

4. **Other structural divergences :**

- Due to typological differences structural divergences between languages exists. For example : In English language adjective appears before noun.

  For example : green witch

  whereas in French and Spanish adjectives follow nouns.

  For example : maison bleue → French

  house blue

- Dates also appear in various formats

  For example : DD/MM/YY in British English

  YYMMDD in Japanese

5. **Lexical divergences :**

- Lexical divergences make the task of translation very difficult.
- The problem like word sense disambiguation is crucial in translation process.

  For example : English word bass appears in Spanish as fish lubine or instrument bajo.
- Lexical divergences can be grammatical also. We can get the translation of source language words to a different part of speech in target language.
- Also there can be word choice constraints on grammatical rules.
- One more problem is of lexical gap i.e. no word or phrase exists which expresses the meaning of word in other language.

5.3 Machine Translation Approaches

- There are two approaches for machine translation,
  1. Classical machine translation
  2. Statistical machine translation

- In this section we will understand various pre-statistical classical machine translation approaches.
- Three classical MT approaches include,

1. **Direct translation :**
   - Each word is translated as it appears in the source language text.
   - It requires large bilingual dictionary.
2. **Transfer approach :**
   - First input text is parsed and then rules are applied to transform source to target language.
   - Then target language sentence is generated from the parse tree.
3. **Interlingua approach :**
   - Source language text is converted to abstract meaning representation known as interlingua.
   - Afterwards target language is generated from this interlingual representation.

- These three approaches can be represented using vauquois triangle as shown in Fig. 5.3.1.

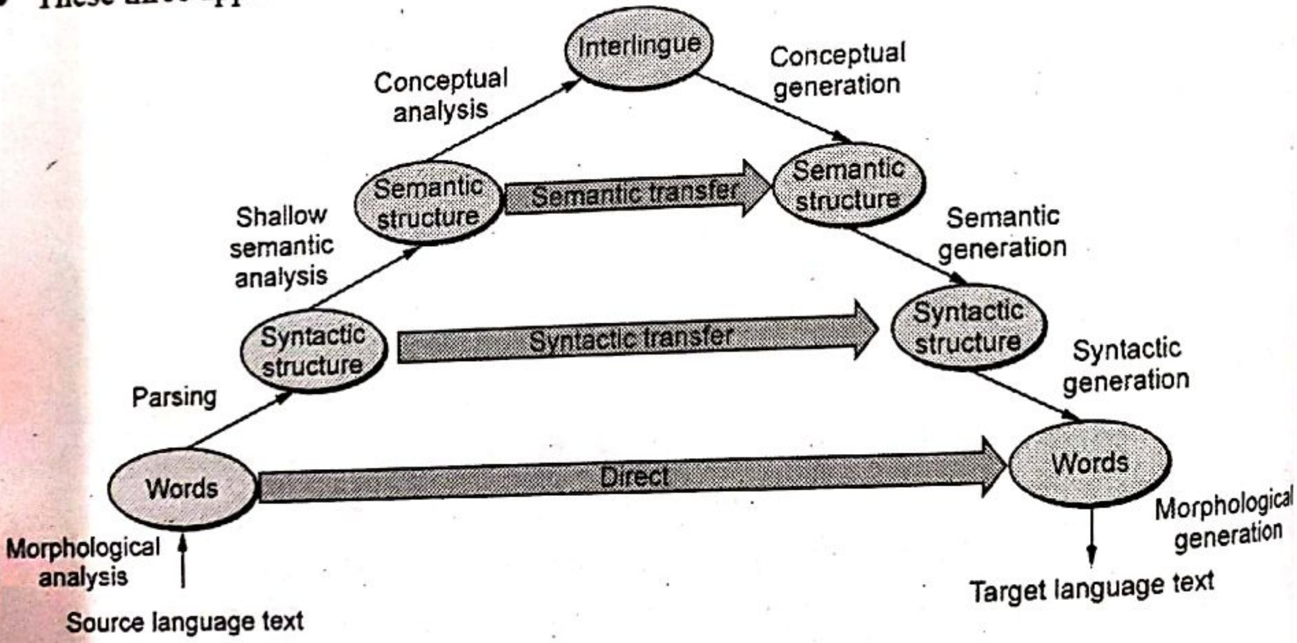


Fig. 5.3.1 The Vauquois (1968) triangle

- Let's try to understand these approaches one by one.

5.4 Direct Translation

- In direct translation word by word translation of source language text is done in the order of appearance of words.

- Every word is mapped on target word directly.
- To accomplish this there is a need of large bilingual dictionary.
- Each entry in this dictionary can be considered as a small program for translation of one word.
- After translation reordering can be done if required, for example : moving adjectives after nouns in English to French translation.
- Direct translation process can be understood from the Fig. 5.4.1.
- The four steps outlined in Fig. 5.4.1 would proceed as shown in Fig. 5.4.2.

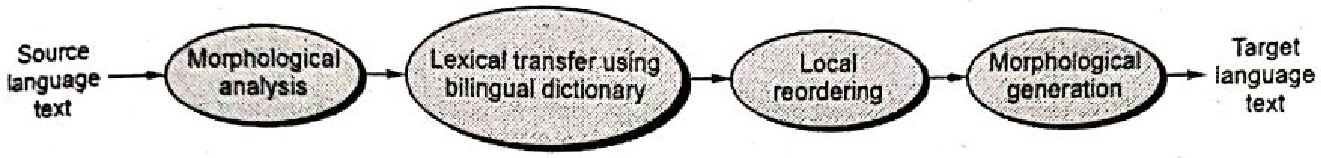


Fig. 5.4.1 Direct machine translation. The major component, indicated by size here, is the bilingual dictionary

- Let's consider the example,

  Mary didn't slap the green witch

  Maria no dió una bofetada a la bruja verde

  Mary not gave a slap to the witch green

- As per Fig. 5.4.1 after 1st step of morphological analysis we get the output as shown in Fig. 5.4.2.

Input :	Mary didn't slap the green witch
After 1 : Morphology	Mary DO-PAST not slap the green witch
After 2 : Lexical transfer	Maria PAST no dar una bofetada a la verde bruja
After 3 : Local recording	Maria no dar PAST una bofetada a la bruja verde
After 4 : Morphology	Maria no dió una bofetada a la bruja verde

Fig. 5.4.2 An example of processing in a direct system

- In step 2 it is considered that bilingual dictionary has the phrase dar una bofetada a which is a Spanish translation of English word slap.
- In step 3 reordering will be done and adjective noun is switched from green witch to bruja verde. The dictionary entries are usually complex One of the sample dictionary is shown in Fig. 5.4.3.

```
function DIRECT_TRANSLATE_MUCH/MANY(word) returns Russian translation
If preceding word is how return skol'ko
else if preceding word is as return stol'ko zhe
else if word is much
    If preceding word is very return nil
    else if following word is a noun return mnogo
else /* word is many */
    If preceding word is a preposition and following word is a noun return mngii
    else return mnogo
```

Fig. 5.4.3 A procedure for translation much and many into Russian, adapted from Hutchins (1986, pg. 133) discussion of Panov (1960). Note the similarity to decision list algorithms for word sense disambiguation

- Some of the drawbacks of direct translation approach are :
  - Knowledge about phrasing or grammatical structure in the source or target language is missing, making it difficult to handle long distance reordering. For example : As shown in Fig. 5.4.4 the translation of English to German language which is near to English.

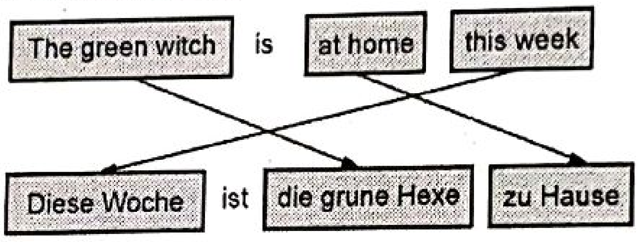


Fig. 5.4.4 Complex reorderings necessary when translating from English to German. German often puts adverbs in initial position that English would more naturally put later. German tensed verbs often occur in second position in the sentence, causing the subject and verb to be inverted

  - More complex reordering is witnessed in the translation of the languages where SVO to SOV reordering is needed like in the example below of translation of English-Japanese.

    He adores listening to music

    Kare ha ongaku wo kiku no ga daisuki desu

  - From these examples we can see that direct approach is too focused on individual words. To avoid these problems inclusion of phrasal and structural knowledge can be beneficial.

5.5 Transfer

- Each language has a specific structure which is unique in its own way.
- To overcome these differences while doing MT contrastive knowledge can be applied, which is knowledge of the differences between two languages.
- This is the basis of transfer model.
- In transfer model there are three phrases,
  1) Analysis
  2) Transfer
  3) Generation.
- The first step is parsing of a source language.
- The parse for MT can be different than parsing for other applications as there is no need of finding out the attachment of prepositional phrases for the sentence 'John saw the girl with binoculars' in transforming it to French.
- After parsing rules and syntactic transfer and lexical transfer need to be applied.
- In syntactic transfer rules source parse tree is modified so that it will resemble target parse tree as shown in Fig. 5.5.1.

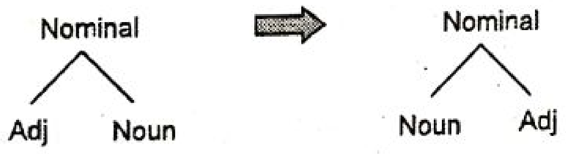


Fig. 5.5.1 A simple transformation that reorders adjectives and nouns

- As shown is Fig. 5.5.1 syntactic transformations operations map one tree structure to another.
- With this mapping we also assume that morphological processing is done where it is understood that didn't is formed by do-PAST plus not, where PAST is a VP. So that in the lexical transfer do is removed not is changed to no and slap is turned to dar una bofetada a as shown in Fig. 5.5.2.
- The second part of transfer based systems is lexical transfer.
- It is uses bilingual dictionary similar to direct transfer.
- May times a word has many different possible translations in the target language where bilingual dictionary can do this job naturally, for example : English word Home as different possible translations in German like House (going home), Hein (Home game), Zu House (being at Home) where through bilingual dictionary it can be figured out that Zu House is most probable translation.

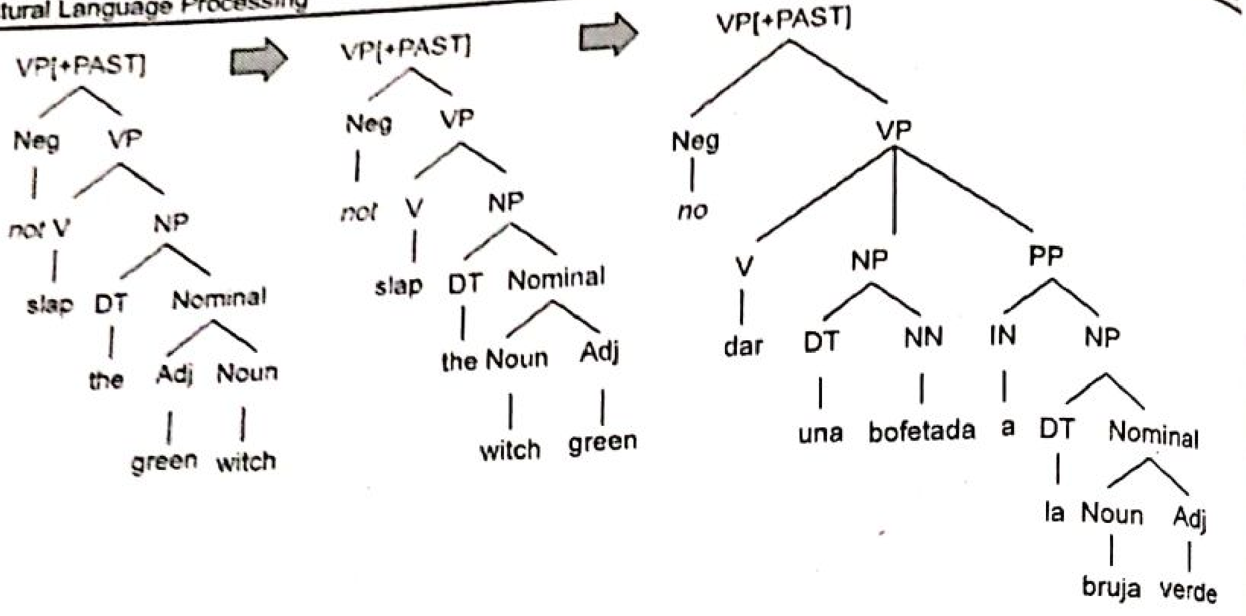


Fig. 5.5.2 A further sketch of the transfer approach

5.6 Interleague (Meaning) based MT

- In rule based transfer model i.e. MT we need distinct set of transfer rules for each pair of languages which is sometimes a difficult task for some languages.
- Interlingua approach focuses on the method of extracting meaning of input and expressing it to target language.
- So this can be considered as knowledge based MT, where like contrastive knowledge used in rule based system only standard interpreter and generator can be used for the language.
- Interlingua is a language independent canonical form.
- The basic idea of interlingua is all the sentences in any language carry the same intended meaning.
- Translation is done by doing deep semantic analysis on input from language X and generating interlingual representation which is then transformed to language Y.
- The example interlingual representation is shown in Fig. 5.6.1.

$$\begin{bmatrix} \text{EVENT} & \text{SLAPPING} \\ \text{AGENT} & \text{MARY} \\ \text{TENSE} & \text{PAST} \\ \text{POLARITY} & \text{NEGATIVE} \\ \text{THEME} & \begin{bmatrix} \text{WITCH} & & \\ \text{DIFFINITENESS} & \text{DEF} & \\ \text{ATTRIBUTES} & [\text{HAS-COLOR} & \text{GEEN}] \end{bmatrix} \end{bmatrix}$$

Fig. 5.6.1 Interlingual representation of Mary did not slap the green witch

- The example represents even based representation where small fixed set of semantic roles are used to link events with their arguments.
- Interlingual representation can be created from source language text by using semantic analyzer techniques.
- For example in the event Mary and slap semantic role labeler can figure out the AGENT relation.
- Unlike transfer model which need only syntactic parsing more analysis work is needed in interlingua.
- In interlingua most of the translation process can be done using general language processing techniques and modules.
- Some of the drawbacks of this approach are :
    - It requires exhaustive analysis of the semantics involved in a domain and formalize it to ontology which is an additional work.

      For example : In translating of sentence from Japanese to Chinese, universal interlingua should include the concepts like ELDER-BROTHER and YOUNGER-BROTHER.

5.7 Statistical MT

- In statistical MT the result is focussed and not the process like the classical MT approaches explained earlier.
- When we wish to translate a sentence from one language to another it is always a challenging task to map all the culture specific concepts metaphors in languages, a word or a tense parallely to other language.
- All these constraints makes true translation of one language to other an very difficult or sometimes, impossible task.
- We can view MT task as a model to produce an output which maximises some value function representing importance of faithfulness and fluency.
- In statistical MT probabilistic models of faithfulness and fluency are built and later these are combined to choose most probable translation.
- Consider faithfulness and fluency as quality metric translation of source model sentence S to target language $\hat{T}$ is given as :

  best - translation $\hat{T} = \text{argmax}_T$ faithfulness (T, S) fluency (T)

  which resembles a noisy channel model.

- Let's consider following information for MT task,
- Let's consider a foreign language sentence $F = f_1, f_2, \ldots, f_m$ which will be converted to English.
- French and Spanish will be considered as foreign languages but target language is English every time.
- According to probabilistic modes the English sentence $\hat{E} = e_1, e_2, \ldots, e_i$ is best one if it has highest probability $P(E \mid F)$.
- If we consider a noisy channel model then this can be rewritten using Bayes rule as

$$\begin{aligned} \hat{E} &= \text{argmax}_E \, P(E \mid F) \\ &= \text{argmax}_E \frac{P(F \mid E)\, P(E)}{P(F)} \\ &= \text{argmax}_E \, P(F \mid E)\, P(E) \end{aligned}$$

- As in argmax denominator $P(F)$ is a constant we can ignore it.
- So we get a noisy channel equation with two components : a translation model $P(F \mid E)$ and language model $P(E)$.

i.e.

$$\hat{E} = \underset{E \in \text{English}}{\text{argmax}} \quad \overset{\textit{Translation model}}{P(F \mid E)} \quad \overset{\textit{Language model}}{P(F)}$$

- Here the assumption is made that model generates corrupt version of English from foreign language input F. So our task is to find out the hidden sentence E which generated observation sentence F.
- This process is shown in Fig. 5.7.1.
- In statistical MT model there are three components for translating fresh sentence F to English sentence E :
  1) A language model to compute $P(E)$.
  2) A translation model to compute $P(F|E)$.
  3) A decoder which produces most probable E from given F.
- The first component was discussed in detail earlier.
- In the next sections we focus on translation model and decoding algorithms.

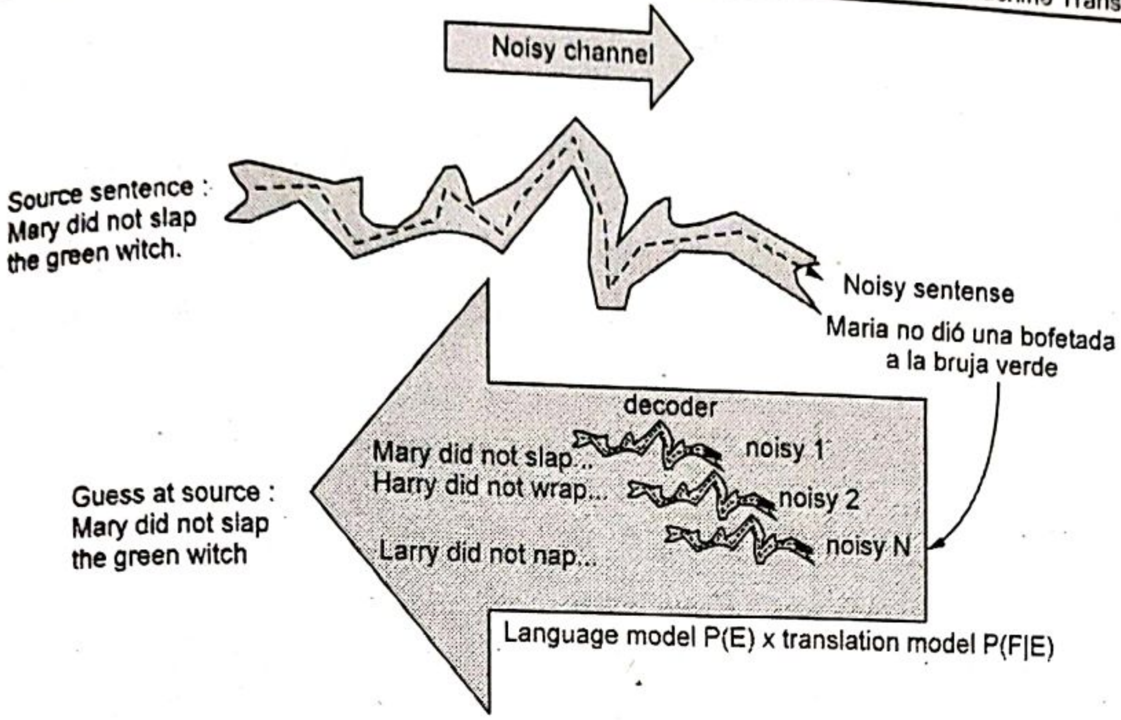

Noisy channel
Source sentence : Mary did not slap the green witch.
Noisy sentense
Maria no dió una bofetada a la bruja verde
decoder
Mary did not slap...
Harry did not wrap...
Larry did not nap...
noisy 1
noisy 2
noisy N
Guess at source : Mary did not slap the green witch
Language model P(E) x translation model P(F|E)

Fig. 5.7.1 The noisy channel model of statistical MT. To translate Spanish (source language) to English (target language), we instead think of "sources" and "targets" backwards. We build a model of the generation process from an English sentence through a channel to a Spanish sentence. Now given a Spanish sentence to translate, we pretend it is the output of an English sentence going through the noisy channel and search for the best possible "source" English sentence

The phrase-based translation model P(F|E) :

- The task of phrase based translation model is assigning a probability that English sentence E generates foreign sentence F.
- Instead of focusing on translation of individual word, modern statistical MT focuses on behavior of phrases.
- Phrase based statistical MT model uses phrases as well as single words as fundamental units of translation.
- This can be understood from the example in Fig. 5.7.2

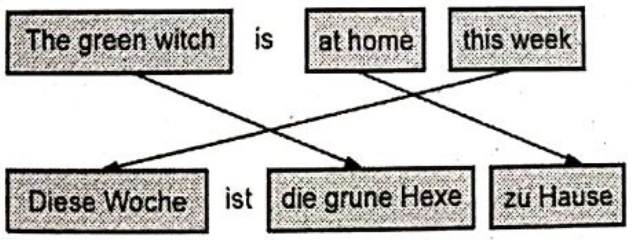


Fig. 5.7.2 Phrasal reorderings necessary when generating German from English : repeated from Fig. 5.4.5

- The phrase based translation model follows three steps :
  1) In the first step English source words are grouped into phrases $\bar{e}_1, \bar{e}_2, - \bar{e}_I$.
  2) In the second step each English phrase $\bar{e}_i$ is translated to Spanish phrase $\bar{f}_i$.
  3) In the third step each of the Spanish phrase is recorded.
- Phrase based translation uses probability model based on translation probability and distortion probability where :

  Factor $\phi(\bar{f}_i \mid \bar{e}_i) \rightarrow$ translation probability of generating Spanish phrase $\bar{f}_i$ from English phrase $\bar{e}_i$.

  $d \rightarrow$ distortion probability for reordering of Spanish phrases.

  Distortion is positional difference of word in Spanish sentence than in English. It is measure of distance between position of phrases.
  Parameter of distortion is $d(a_i - b_{i-1})$

  where

  $a_i \rightarrow$ Start position of Spanish phrase generated by ith English phrase $\bar{e}_i$

  $b_{i-1} \rightarrow$ End position of Spanish phrase generated by i–1th English phrase $\bar{e}_i - 1$.

  A small constant α can be raised to distortion.

  So, $\quad d(a_i - b_{i-1}) = \alpha^{|a_i - b_{i-1} - 1|}$

- Final model of phrase based MT is given as,

$$P(F \mid E) = \prod_{i=1}^{I} \phi(\bar{f}_i, \bar{e}_i)\, \alpha(a_i - b_{i-1})$$

- To understand this process consider following example shown in Fig. 5.7.3.

Position	1	2	3	4	5
English	Mary	did not	Slap	the	green witch
Spanish	Maria	no	dió una bofetada	a la	bruja verde

Fig. 5.7.3

- In this example the distortions are 1 and probability P(F | E) is computed as,

$$\begin{aligned} P(F \mid E) = {} & P(\text{Maria, Mary}) \times d(1) \times P(\text{no} \mid \text{did not}) \times \\ & d(1) \times P(\text{dió una bofetada} \mid \text{slap}) \\ & \times d(1) \times P(\text{a la} \mid \text{the}) \times d(1) \\ & \times P(\text{bruja verde} \mid \text{green witch}) \times d(1) \end{aligned}$$

- For using phrase based mode we need two more models :
  1. **Model of decoding :** For finding hidden English string from surface Spanish string
  2. **Model of training :** To learn parameters i.e. set of phrase translation probabilities

$$\phi(\bar{f}_i \mid \bar{e}_i)$$

- The parameters for training and distortion constant α are set if we have large bilingual training set.
- The training set should have pairing of each Spanish sentence with English to understand which Spanish sentence phrase is translated by English sentence phrase.
- This mapping is called as phrase alignment.
- With large training set with sentence labelled with phrase alignment, the number of times each phrase pair occurred is counted and normalized to get probabilities as,

$$\phi(\bar{f}, \bar{e}) = \frac{\text{count}(\bar{f}, \bar{e})}{\Sigma \bar{f} \text{ count } (\bar{f}, \bar{e})}$$

- Each phrase pair $(\bar{f}, \bar{e})$ along with probability $\phi(\bar{f}, \bar{e})$ is stored in phrase translation table.
- One more way of extracting exact phrases as word alignment.
- Example of word alignment is shown in Fig. 5.7.4 and word alignment matrix in Fig. 5.7.5.

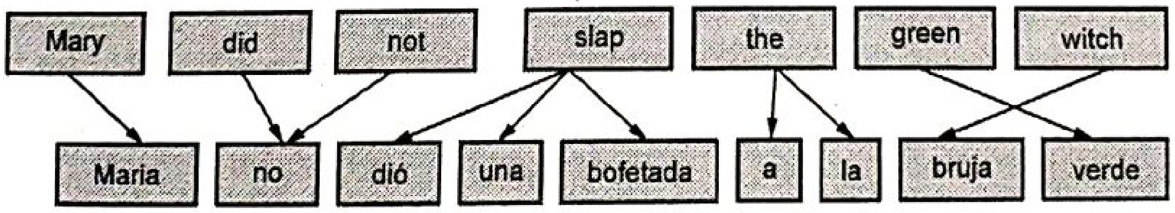


Fig. 5.7.4 A graphical model representation of a word alignment between the English and Spanish sentences.

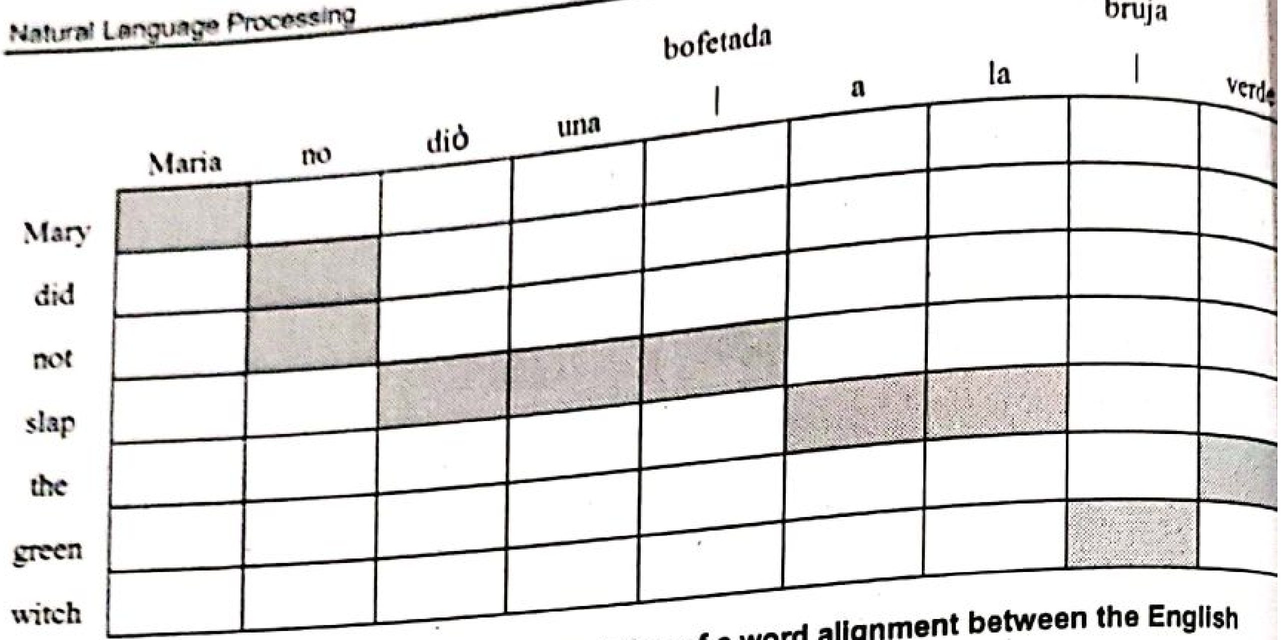


Fig. 5.7.5 An alignment matrix representation of a word alignment between the English and Spanish sentences.

Alignment in MT :

- Word alignment is the basic concept used in statistical translation model.
- A word alignment is a mapping between the source words and the target words in a set o parallel sentences.
- To understand word alignment consider the example shown in Fig. 5.7.6.

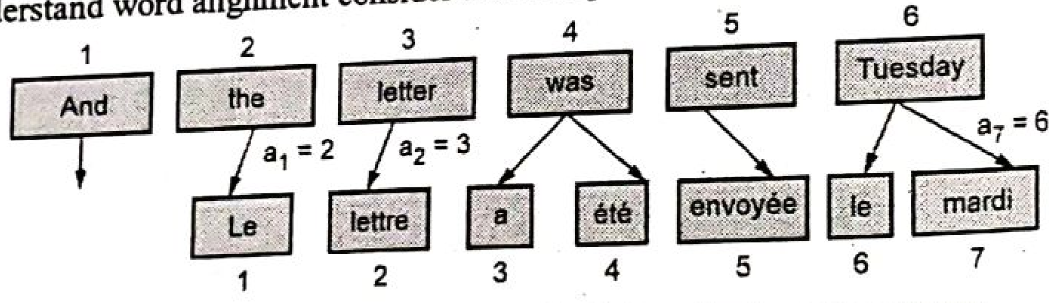


Fig. 5.7.6 An alignment between an English and a French sentence. Each French word aligns to a single English word.

- In this section we will discuss IBM models for word alignment.
- IBM models consider that each French word comes from exactly one English word.
- By this assumption we can assign index number of the English word that the French word comes from.
- So the alignment in Fig. 5.7.6 can be given as A = 2, 3, 4, 4, 5, 6, 6

5.8 Parameter Learning in SMT (IBM Models) using EM

- IBM models are the statistical alignment algorithms.

- IBM model 1 for generating a Spanish sentence from English sentence $E = e_1, e_2, \dots e_I$ of length I is described as follows :
  1. Choose a length J for the Spanish sentence,
  $$F = f_1, f_2, \dots f_J$$
  2. $A = a_1, a_2, \dots a_J$ is the alignment between English and Spanish sentences.
  3. For each position j in the Spanish sentence choose a Spanish word f_j by translating English word aligned to it.

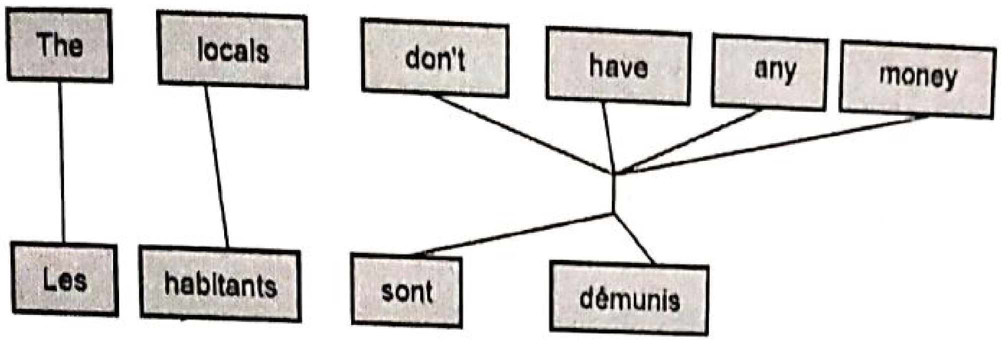


Fig. 5.8.1 An alignment between an English and a French sentence, in which there is a many-to-many alignment between English and French words

- This process is described in Fig. 5.8.2.

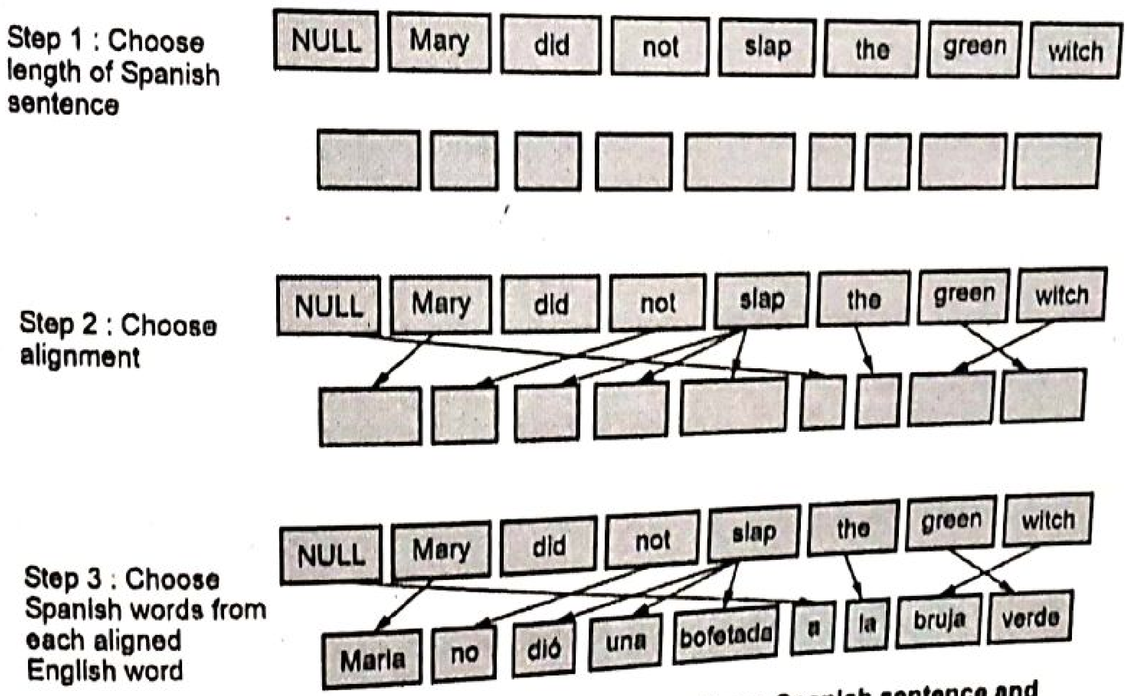

Step 1 : Choose length of Spanish sentence
NULL
Mary
did
not
slap
the
green
witch
Step 2 : Choose alignment
Step 3 : Choose Spanish words from each aligned English word
Maria
no
dió
una
bofetada
a
la
bruja
verde

Fig. 5.8.2 The three steps of IBM Model 1 generating a Spanish sentence and alignment from an English sentence

- Let's understand how generative probability model for model 1 is formulated for assigning a probability to each possible Spanish sentence F.

- Consider P(F|E) is the probability of generating Spanish sentence F from English sentence E.
- Following terminology is used :

  $e_{aj} \rightarrow$ English word aligned to Spanish word f_j

  $t(f_x|e_y) \rightarrow$ Probability of translating e_y by f_x i.e. $P(f_x | e_y)$
- According to step 3 in algorithm, assume that length J, alignment A and English source E is already known. So the probability of Spanish sentence can be given as,

$$P(F | E, A) = \prod_{j=1}^{I} t(f_j|e_{aj})$$

- Now consider step 1 and 2 of algorithm.

  Consider English sentence has length I, Spanish sentence has length J.
- Let's assume that each alignment in equally likely i.e. each Spanish and come from one of I English words so there are $(I + 1)^J$ possible alignments.
- Let's assume that probability of choosing length J is small constant $\in$.
- So the probability P(A | E) of alignment A of length J given English sentence E is given as,

$$P(A | E) = \frac{\in}{(I + 1)^J}$$

which is a combined probability of choosing length J and $(I + 1)^J$ is one of the possible alignments.

- These probabilities can be combined as,

$$P(F, A | E) = P(F | E, A) \times P(A | E)$$
$$= \frac{\in}{(I + 1)^J} \prod_{j=1}^{J} t(f_j | e_{aj})$$

where $P(F, A | E)$ = Probability of generating Spanish sentence F through a particular alignment.

- Total probability P(F | E) of generating F is given by summing all possible alignments.

$$P(F/E) = \sum_{A} P(F, A | E) = \sum_{A} \frac{\in}{(I + 1)^J} \prod_{j=1}^{J} t(f_j | e_{aj})$$

- This equation represents the generative probability model of IBM model 1.

- The best alignment between pair of sentences F and E can be computed by Viterbi algorithm and it is given by,

$$\hat{A} = \underset{A}{\text{argmax}}\ P(F, A \mid E)$$

$$= \underset{A}{\text{argmax}}\ \frac{\in}{(I+1)^J} \prod_{j=1}^{J} t(f_j \mid e_{aj})$$

$$= \underset{a_j}{\text{argmax}}\ t(f_j \mid e_{aj}) \quad 1 < j < J$$

5.9 Neural Machine Translation

- Neural Machine Translation (also known as Neural MT, NMT, Deep Neural Machine Translation, Deep NMT, or DNMT) is a state-of-the-art machine translation approach that utilizes neural network techniques to predict the likelihood of a set of words in sequence. This can be a text fragment, complete sentence, or with the latest advances an entire document.
- Unlike statistical machine translation, which consumes more memory and time, neural machine translation, NMT, trains its parts end-to-end to maximize performance.
- NMT uses deep neural networks and artificial intelligence to train neural models
- Unlike the traditional phrase-based translation system which consists of many small sub-components that are tuned separately, neural machine translation attempts to build and train a single, large neural network that reads a sentence and outputs a correct translation.
- NMT makes use of vector representations for words. This means that words are transcribed into a vector defined by a unique magnitude and direction.
- Compared to phrase-based models, this framework is much simpler.
- Rather than separate component like the language model and translation model, NMT uses a single sequence model that produces one word at a time.
- As shown in the Fig. 5.9.1, the NMT uses a bidirectional recurrent neural network, also called an encoder, to process a source sentence into vectors for a second recurrent neural network, called the decoder, to predict words in the target language.

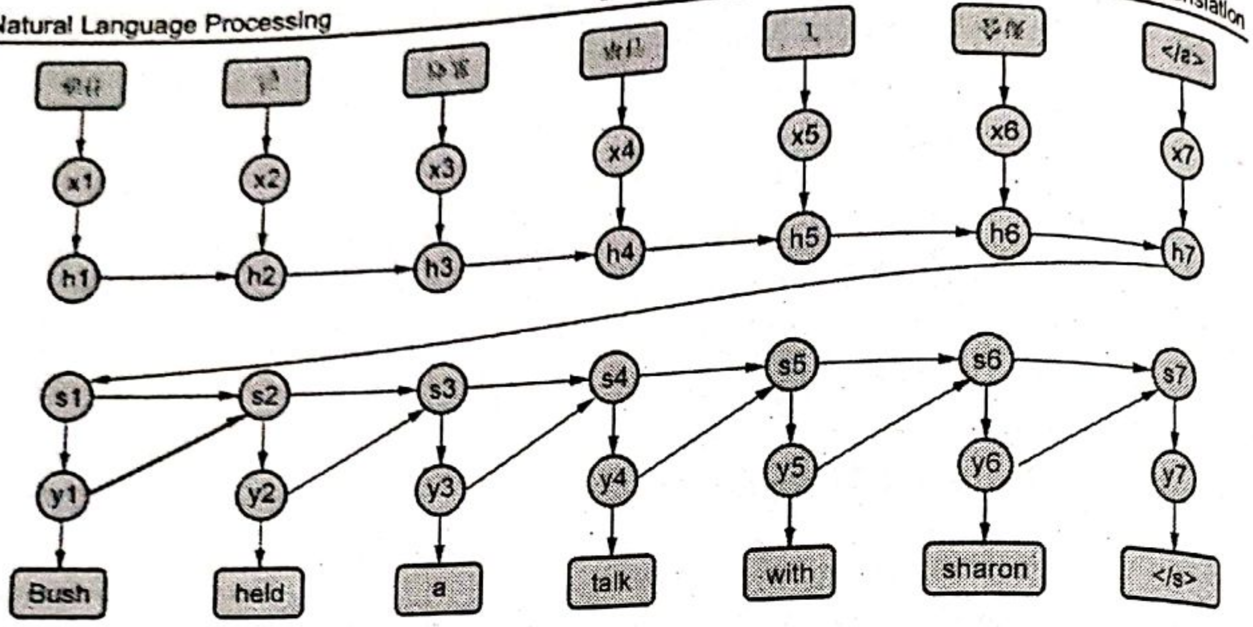


Fig. 5.9.1

5.10 Encoder and Decoder Architecture for MT

- The encoder-decoder recurrent neural network architecture with attention is currently the state-of-the-art on some benchmark problems for machine translation.
- This architecture is used in the heart of the Google Neural Machine Translation system, or GNMT, used in their Google Translate service.
- The following description will cover encoder-decoder architecture :
  - Encoder
    - The encoder is the feeding end of the system. It understands the sequence and reduces the dimension of the input sequence.
    - The sequence is written in a fixed size known as the *context vector*.
    - The context vector acts like input to the decoder, which generates an output sequence after the end token is reached.
    - These sequence models are known as encoder-decoder models.
    - This architecture can handle input and output sequences of variable length.
  - Decoder
    - If LSTM is used for the encoder, the same is used for the decoder.
    - Decoder is more complex than the encoder network.
    - The decoder is in an "aware state". It knows what words you have generated so far and what the previous hidden state was.
    - The first layer of the decoder is initialized by using the context vector 'C' from the encoder network to generate the output.

- Then a special token is applied at the start to indicate the output generation. It applies a similar token at the end.
- The first output word is generated by running the stacked LSTM layers.
- A *SoftMax* activation function applies to the last layer. Its job is to introduce non-linearity in the network. Now this word is passed through the remaining layers and the generation sequence is repeated.

- The parameters such as optimizers, cross-entropy loss, learning rate, etc., play an important role in improving the model's performance.
- Refer to the Fig. 5.10.1 below to understand the architecture of encoder decoder model.

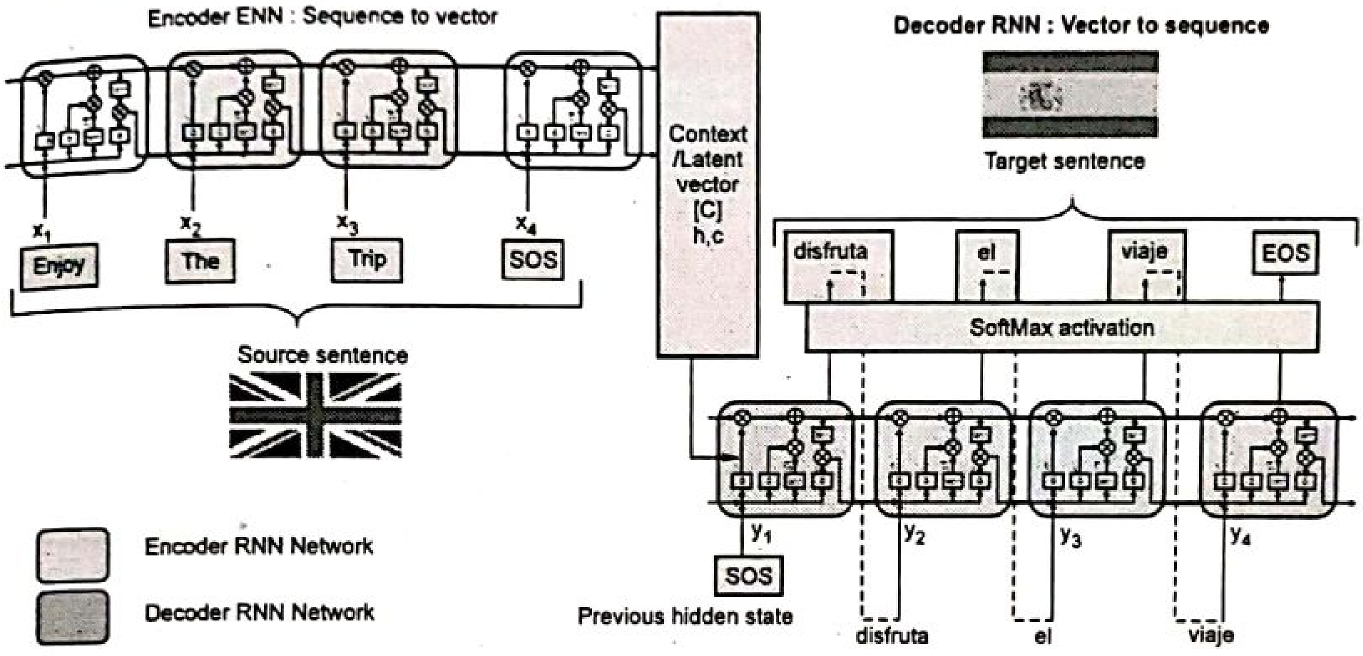


Fig. 5.10.1 Architecture of encoder decoder model

Applications of neural network translation

- One of the most popular translation systems in the world is Google translate.
- The system uses Google neural machine translation to enhance fluency and accuracy.
- The system not only applies a large data set for training its algorithms, its end-to-end design allows the system to learn over time and create better, more natural translations.
- Google neural machine translation can even process "zero-shot translations."
- For example, the translation from French to Spanish is a zero-shot translation because it is a direct translation.
- Previously, Google translate would translate the initial language into English, and then translate that English to the target language.

Review Questions

1. What is the need of machine translation ?
2. List problems of machine translation.
3. Explain typology.
4. Explain syntactic aspects.
5. Explain lexical divergences.
6. Explain MT approaches.
7. Explain direct translation.
8. Explain direct machine translation process.
9. What are the drawbacks of direct translation approach ?
10. Explain transfer model.
11. Explain interleague or meaning based MT.
12. Explain statistical MT.
13. Explain phrase based MT with example.
14. Explain alignment in MT.
15. Explain neural machine translation.
16. Explain encoder and decoder architecture for MT.
17. What are the applications of neural network translation ?

Winter-2021 [3170723]
Natural Language Processing

Semester - VII (CE / CSE) Professional Elective - VI

Gujarat Technological University
Solved Paper

Time : $2\frac{1}{2}$ Hours] [Total Marks : 70

Instructions :

1. Attempt all questions.
2. Make suitable assumptions wherever necessary.
3. Figures to the right indicate full marks.
4. Simple and non-programmable scientific calculators are allowed.

Q.1 a) *State advantages and disadvantages of Natural Language Processing.* **(Refer sections 1.4 and 1.5)** [3]

b) *Explain components of Natural Language Processing.* **(Refer section 1.6)** [4]

c) *Explain phases of Natural Language Processing in detail.* **(Refer section 1.9)** [7]

Q.2 a) *Explain Markov model in brief.* **(Not in New Syllabus)** [3]

b) *Explain Add-k smoothing with example.* **(Refer section 2.7)** [4]

c) *Describe Kneser-Ney smoothing with suitable example.* **(Refer section 2.6.4)** [7]

OR

c) *Describe Part-of-Speech Tagging with suitable example.* **(Refer section 2.10)** [7]

Q.3 a) *Give the intuition of skip-gram.* **(Refer section 3.2)** [3]

b) *Explain word embedding in brief.* **(Refer sections 3.4)** [4]

c) *Describe nearest neighbor algorithm for Word Sense Disambiguation.* **(Not in New Syllabus)** [7]

OR

Q.3 a) *Give the example of bag-of-words model.* **(Refer section 3.1)** [3]

b) *Explain knowledge-based method to WSD.* **(Refer section 3.7)** [4]

c) *Describe various relations between senses with suitable example.* **(Refer section 3.7)** [7]

Q.4 a) *Give the examples of temporal lexical triggers.* **(Not in New Syllabus)** [3]

b) *Create a confusion matrix for visualizing how well a binary classification system performs against gold standard labels.* **(Not in New Syllabus)** [4]

c) *Explain stages of IR-based question answering model.* **(Refer section 4.4)** [7]

OR

Q.4 a) *State different algorithms used for relation extraction.* **(Refer section 4.3.2)** [3]

b) *Create confusion matrix for a three-class categorization task.* **(Not in New Syllabus)** [4]

c) *Explain how Natural Language Processing is useful in Information Retrieval.* **(Refer section 4.5)** [7]

Q.5 a) *Explain translation divergence in brief.* **(Refer section 5.2)** [3]

b) *Write a short note on rule-based machine translation.* **(Refer section 5.3)** [4]

c) *Explain the training of Encoder-Decoder Model with RNNs.* **(Refer section 5.10)** [7]

OR

Q.5 a) *Explain SVO, SOV and VSO in brief.* **(Refer section 5.2, Point 2)** [3]

b) *Write a short note on direct machine translation.* **(Refer section 5.4)** [4]

c) *How does parameter learning in SMT performed ? Explain with suitable example.* **(Refer section 5.7)** [7]

❑❑❑