Natural Language Processing

R18 B.Tech. CSE (AIML) III & IV Year JNTU Hyderabad

Prepared by
K SWAYAMPRABHA
Assistance Professor

- Prerequisites: Data structures, finite automata and probability theory.

- Course Objectives:
- Introduce to some of the problems and solutions of NLP and their relation to linguistics and statistics.

- Course Outcomes:
- Show sensitivity to linguistic phenomena and an ability to model them with formal grammars.
- Understand and carry out proper experimental methodology for training and evaluating
- Empirical NLP systems
- Able to manipulate probabilities, construct statistical models over strings and trees, and
- Estimate parameters using supervised and unsupervised training methods.
- Able to design, implement, and analyze NLP algorithms
- Able to design different language modeling Techniques.

TEXT BOOKS:
- 1. Multilingual natural Language Processing Applications: From Theory to Practice – Daniel M.Bikel and Imed Zitouni, Pearson Publication
- 2. Natural Language Processing and Information Retrieval: Tanvier Siddiqui, U.S. Tiwary

- REFERENCE BOOK:
- 1. Speech and Natural Language Processing - Daniel Jurafsky & James H Martin, Pearson Publications

- **Natural Language Processing is a part of artificial intelligence that aims to teach the human language with all its complexities to computers.**

Top 7 Applications of NLP (Natural Language Processing)

1. Chatbots
Chatbots are a form of artificial intelligence that are programmed to interact with humans in such a way that they sound like humans themselves. Depending on the complexity of the chatbots, they can either just respond to specific keywords or they can even hold full conversations that make it tough to distinguish them from humans.

2. Autocomplete in Search Engines
Have you noticed that search engines tend to guess what you are typing and automatically complete your sentences? For example, On typing "game" in Google, you may get further suggestions for "game of thrones", "game of life" or if you are interested in maths then "game theory". All these suggestions are provided using autocomplete that uses Natural Language Processing to guess what you want to ask.

3. Voice Assistants
These days voice assistants are all the rage! Whether its Siri, Alexa, or Google Assistant, almost everyone uses one of these to make calls, place reminders, schedule meetings, set alarms, surf the internet etc

4. Language Translator
Want to translate a text from English to Hindi but don't know Hindi? Well, Google Translate is the tool for you! While it's not exactly 100% accurate, it is still a great tool to convert text from one language to another. Google Translate and other translation tools as well as use Sequence to sequence modeling that is a technique in Natural Language Processing.

5. Sentiment Analysis
Almost all the world is on social media these days! And companies can use sentiment analysis to understand how a particular type of user feels about a particular topic, product, etc. They can use natural language processing, computational linguistics, text analysis, etc. to understand the general sentiment of the users for their products and services and find out if the sentiment is good, bad, or neutral.

6. Grammar Checkers

7. Email Classification and Filtering

UNIT - I

- Finding the Structure of Words: Words and Their Components, Issues and Challenges, Morphological Models

- Finding the Structure of Documents: Introduction, Methods, Complexity of the Approaches, Performances of the Approaches

Words and Their Components

1 Tokens
2 Lexemes
3 Morphemes
4 Typology

1. Tokens : Tokenization is a way of separating a piece of text into smaller units called **tokens**. Here, **tokens** can be either words, characters, or subwords.

2. Lexemes: By the term word, we often denote not just the one linguistic form in the given context but also the concept behind the form and the set of alternative forms that can express it. Such sets are called lexemes or lexical items, and they constitute the lexicon of a language.

- The lexeme "play," for example, can take many forms, such as **playing, plays, played**.

3. Morpheme: is the smallest unit of a word that provides a specific meaning to a string of letters (which is called a phoneme).

There are two main types of morpheme:

i) free morphemes and
ii) bound morphemes.

- For example, "apple" is a word and also a morpheme. "Apples" is a word comprised of two morphemes, "apple" and "-s", which is used to signify the noun is plural.
- Unhappy

4. Typology is **the study the ways in which the languages of the world vary in their patterns**. It is concerned with discovering what grammatical patterns are common to many languages and which ones are rare.

They are specified according to three criteria:

- Genealogical familiarity
- Structural familiarity

- Geographic distribution

According to these criteria, the below are the important language family groups:

- Indo-European
- Sino-Tibetan
- Niger-Congo
- Afroasiatic
- Austronesian
- Altaic
- Japonic
- Austroasiatic
- Tai-Kadai

The most commonly spoken are languages in the Indo-European and Sino-Tibetan language groups. These two groups are used by 67% of the global population.

To scientifically classify languages, the following criteria are used:

- Language criteria
- Historical criteria
- Geographical criteria
- Sociopolitical criteria

The classification of languages shows us the precise connections between the languages of the world.

Applying Linguistic Typology Classifications

According to the implementation of these criteria, there are various classifications and different observations of the relationships between languages. The three basic classifications for languages of the world are:

- Genealogical
- Typological
- Areal

1. GENEALOGICAL CLASSIFICATION

This classification of linguistic typology indicates the historical connection between the languages, and it uses the historical and linguistic criteria as a basis. There are also languages that cannot be classified in to any language family group. For example, in Europe, the Basque language is called a language isolate, as it cannot relate to any other language.

2. TYPOLOGICAL CLASSIFICATION

Languages are grouped into language types on the basis of formal criteria, according to their similarities in grammatical structure. There are several types: flexile (morphological resources), agglutinative (affixes), and rooted (the root of the word as a morphological resource).

3. AREAL CLASSIFICATION

It involves geographic criteria, and covers those languages that are close by and have developed similar characteristics in terms of structure. Under the influence of intensive mutual influences, these kinds of languages are creating language unions such as the Balkan Language Union, encompassing Macedonian, Bulgarian, Serbian, and Albanian, for example.

Issues and Challenges

1 Irregularity

2 Ambiguity

3 Productivity

1. Irregularity, we mean existence of such forms and structures that are not described appropriately by a prototypical linguistic model. Some irregularities can be understood by redesigning the model and improving its rules, but other lexically dependent irregularities often cannot be generalized.

Irregular Verb Inflections

stem	eat	catch	cut	go
-s form	eats	catches	cuts	goes
-ing participle	eating	catching	cutting	going
Past	ate	caught	cut	went
-ed participle	eaten	caught	cut	gone

2. **Ambiguity** and Uncertainty in Language

- **Lexical Ambiguity**
  The ambiguity of a single word is called lexical ambiguity. For example, treating the word **silver** as a noun, an adjective, or a verb.

- **Syntactic Ambiguity**
  This kind of ambiguity occurs when a sentence is parsed in different ways. For example, the sentence "The man saw the girl with the telescope". It is ambiguous whether the man saw the girl carrying a telescope or he saw her through his telescope.

- **Semantic Ambiguity**
  This kind of ambiguity occurs when the meaning of the words themselves can be misinterpreted. In other words, semantic ambiguity happens when a sentence contains an ambiguous word or phrase.
  For example, the sentence "The car hit the pole while it was moving" is having semantic ambiguity because the interpretations can be "The car, while moving, hit the pole" and "The car hit the pole while the pole was moving".

- **Anaphoric Ambiguity**
  This kind of ambiguity arises due to the use of anaphora entities in discourse. For example, the horse ran up the hill. It was very steep. It soon got tired. Here, the anaphoric reference of "it" in two situations cause ambiguity.

- **Pragmatic ambiguity**
  Such kind of ambiguity refers to the situation where the context of a phrase gives it multiple interpretations. In simple words, we can say that pragmatic ambiguity arises when the statement is not specific.
  For example, the sentence "I like you too" can have multiple interpretations like I like you (just like you like me), I like you (just like someone else dose).

3. Productivity

1. Set Objectives

The simplest way of implementing a form of NLP in your work environment is to make sure everyone is working towards goals. By setting objectives you are giving your team a direction and something to work for. If employees feel they are expected to achieve these firm objectives they will naturally work harder to make sure they do. They

will also automatically be taking more responsibility over their role and the work they do.
This increases productivity on an individual level. Incentives for successfully achieving objectives can also be specified in order to motivate staff to succeed and thrive in the work environment.

2. Boost Staff Morale

NLP is a great way to make employees more engaged and content in the workplace and NLP training is a valuable investment. Committed and engaged employees will perform better than other employees in the business – what are you doing to maintain commitment and engagement in your business? Learning NLP techniques through tailored training courses and coaching will enable team members to reach high levels of performance by over coming barriers in the workplace.
Staff morale is an ongoing factor that a team leader or manager considers. Treating employees with respect, listening to their ideas and making them feel included on a daily basis will keep their self-esteem and confidence high.

3. Better Communication

Internal communications and client relationships are vital for a productive and efficient working environment. Making people aware of how they come across when interacting with others is a key aspect of using NLP to improve communication. NLP will help to identify adverse behaviours such as body language. Body language such as avoiding eye contact or slouching shoulders is generally a subconscious behaviour.
Once the negative behaviour has been recognised, the individual can work to change and improve. As the individual becomes more self-aware, they also become more aware of other people. Effective communication requires an understanding to others' thought processes as well as an awareness of yourself. See yourself in the way that you would like others to see you.

4. Learning and Development

NLP is all about bringing together an individual's innermost skills and highlighting their hidden, concealed ability. NLP unlocks the potential for a wealth of knowledge. Employees will be eager to learn and advance in their own professional development. This enables employees to take control of their own career and requires them to be proactive about doing so. A proactive, engaged and progressing team will be highly motivated and productive.

NLP helps the individual to improve in their job role by taking a highly performing team member and using their behaviour and work ethic as a model for others to adopt. You could have a team that is as strong as your strongest employee. This would have a huge impact on productivity as well as give your business a competitive edge.

5. Changing Behaviour

The main objective of NLP is to reverse negative behaviours and habits. How an individual interprets their workplace has little to do with the actual working environment and more to do with the individual. Employees have completely different experiences at work, even though the work environment is the same for everyone. NLP makes employees aware that the problems they face at work are usually internal, not external. Making employees self-aware of their attitudes and behaviours is the first step towards a positive change.
success, no matter what your goals are.

Morphological Models

1 Dictionary Lookup
2 Finite-State Morphology
3 Unification-Based Morphology
4 Functional Morphology
5 Morphology Induction

Morphological Models

There are many possible approaches to designing and implementing morphological models. Over time, computational linguistics has witnessed the development of a number of formalisms and frameworks, in particular grammars of different kinds and expressive power, with which to address whole classes of problems in processing natural as well as formal languages.

Various domain-specific programming languages have been created that allow us to implement the theoretical problem using hopefully intuitive and minimal programming effort.

1. Dictionary Lookup

Morphological parsing is a process by which word forms of a language are associated with corresponding linguistic descriptions.

In this context, a dictionary is understood as a data structure that directly enables obtaining some precomputed results, in our case word analyses. The data structure can be optimized for efficient lookup, and the results can be shared. Lookup operations are relatively simple and usually quick. Dictionaries can be implemented, for instance, as lists, binary search trees, tries, hash tables, and so on.

Because the set of associations between word forms and their desired descriptions is declared by plain enumeration, the coverage of the model is finite and the generative potential of the language is not exploited. Developing as well as verifying the association list is tedious, liable to errors, and likely inefficient and inaccurate unless the data are retrieved automatically from large and reliable linguistic resources.

For instance, dictionary-based approaches to Korean depend on a large dictionary of all possible combinations of allomorphs and morphological alternations. These approaches do not allow development of reusable morphological rules.

The word list or dictionary-based approach has been used frequently in various ad hoc implementations for many languages. We could assume that with the availability of immense online data, extracting a high-coverage vocabulary of word forms is feasible these days. The question remains how the associated annotations are constructed and how informative and accurate they are. References to the literature on the unsupervised learning and induction of morphology, which are methods resulting in structured and therefore nonenumerative models.

2. Finite-State Morphology

By finite-state morphological models, we mean those in which the specifications written by human programmers are directly compiled into finite-state transducers.

The two most popular tools supporting this approach, which have been cited in literature and for which example implementations for multiple languages are available online, i) XFST (Xerox Finite-State Tool) ii) LexTools

Finite-state transducers are computational devices extending the power of finite-state automata. They consist of a finite set of nodes connected by directed edges labeled with pairs of input and output symbols. In such a network or graph, nodes are also called states, while edges are called arcs. Traversing the network from the set of initial states to the set of final states along the arcs is equivalent to reading the sequences of encountered input symbols and writing the sequences of corresponding output symbols.

The set of possible sequences accepted by the transducer defines the input language; the set of possible sequences emitted by the transducer defines the output language. For example, a finite-state transducer could translate the infinite regular language consisting of the words *vnuk*, *pravnuk*, *prapravnuk*, *...* to the matching words in the infinite regular language defined by *grandson*, *great-grandson*, *great-great-grandson*, *...*

The role of finite-state transducers is to capture and compute **regular relations** on sets [38, 9, 11].[6] That is, transducers specify relations between the input and output languages. In fact, it is possible to invert the domain and the range of a relation, that is, exchange the input and the output. In finite-state computational morphology, it is common to refer to the input word forms as **surface strings** and to the output descriptions as **lexical strings**, if the transducer is used for morphological analysis, or vice versa, if it is used for morphological generation.

Let us have a relation R, and let us denote by $[\Sigma]$ the set of all sequences over some set of symbols Σ, so that the domain and the range of R are subsets of $[\Sigma]$. We can then consider R as a function mapping an input string into a set of output strings, formally denoted by this type signature, where $[\Sigma]$ equals *String*:

Finite-state tools are available in most general-purpose programming languages in the form of support for regular expression matching and substitution. While these may not be the ultimate choice for building full-fledged morphological analyzers or generators of a natural language, they are very suitable for developing tokenizers and morphological guessers capable of suggesting at least some structure for words that are formed

correctly but cannot be identified with concrete lexemes during full morphological parsing.

3. Unification-Based Morphology

Unification-based approaches to morphology have been inspired by advances in various formal linguistic frameworks aiming at enabling complete grammatical descriptions of human languages, especially head-driven phrase structure grammar (HPSG), and by development of languages for lexical knowledge representation, especially DATR. The concepts and methods of these formalisms are often closely connected to those of logic programming. In finite-state morphological models, both surface and lexical forms are by themselves unstructured strings of atomic symbols. In higher-level approaches, linguistic information is expressed by more appropriate data structures that can include complex values or can be recursively nested if needed. Morphological parsing *P* thus associates linear forms φ with alternatives of structured content ψ, cf. (1.1):

$$\mathcal{P} :: \phi \rightarrow \{\psi\} \qquad \mathcal{P} :: form \rightarrow \{content\} \qquad (1.2)$$

In either case, information in a model can be efficiently shared and reused by means of inheritance hierarchies defined on the feature structure types.

Morphological models of this kind are typically formulated as logic programs, and unification is used to solve the system of constraints imposed by the model. Advantages of this approach include better abstraction possibilities for developing a morphological grammar as well as elimination of redundant information from it.

Unification-based models have been implemented for Russian [58], Czech [59], Slovene [53], Persian [60], Hebrew [61], Arabic [62, 63], and other languages. Some rely on DATR; some adopt, adapt, or develop other unification engines.

4. Functional Morphology

This group of morphological models includes not only the ones following the methodology of functional morphology , but even those related to it, such as morphological resource grammars of Grammatical Framework . Functional morphology defines its models using principles of functional programming

and type theory. It treats morphological operations and processes as pure mathematical functions and organizes the linguistic as well as abstract elements of a model into distinct types of values and type classes.

Functional morphology implementations are intended to be reused as programming libraries capable of handling the complete morphology of a language and to be incorporated into various kinds of applications. Morphological parsing is just one usage of the system, the others being morphological generation, lexicon browsing, and so on. Next to parsing (1.2), we can describe inflection *I*, derivation *D*, and lookup *L* as functions of these generic types:

$$\mathcal{I} :: lexeme \rightarrow \{parameter\} \rightarrow \{form\} \quad (1.3)$$

$$\mathcal{D} :: lexeme \rightarrow \{parameter\} \rightarrow \{lexeme\} \quad (1.4)$$

$$\mathcal{D} :: lexeme \rightarrow \{parameter\} \rightarrow \{lexeme\} \quad (1.4)$$

A functional morphology model can be compiled into finite-state transducers if needed, but can also be used interactively in an interpreted mode, for instance. Computation within a model may exploit lazy evaluation and employ alternative methods of efficient parsing, lookup, and so on .

Many functional morphology implementations are embedded in a general-purpose programming language, which gives programmers more freedom with advanced programming techniques and allows them to develop full-featured, real-world applications for their models. The Zen toolkit for Sanskrit morphology [67, 68] is written in OCaml. It influenced the functional morphology framework [64] in Haskell, with which morphologies of Latin, Swedish, Spanish, Urdu [69], and other languages have been implemented.

5. Morphology Induction

We have focused on finding the structure of words in diverse languages supposing we know what we are looking for. We have not considered the problem of discovering and inducing word structure without the human insight (i.e., in an unsupervised or semi-supervised manner). The motivation for such approaches lies in the fact that for many languages, linguistic expertise might be unavailable or limited, and implementations adequate to

a purpose may not exist at all. Automated acquisition of morphological and lexical information, even if not perfect, can be reused for bootstrapping and improving the classical morphological models, too.

There are several challenging issues about deducing word structure just from the forms and their context. They are caused by ambiguity [76] and irregularity [75] in morphology, as well as by orthographic and phonological alternations [85] and nonlinear morphological processes [86, 87].

Finding the Structure of Documents

Document Structuring is a subtask of Natural language generation, which involves **deciding the order and grouping (for example into paragraphs) of sentences in a generated text**. It is closely related to the Content determination NLG task.

1. Sentence Boundary Detection

2. Topic Boundary Detection

Sentence boundary detection is the method of detecting where one sentence ends and another begins. If you are thinking that this sounds pretty easy, as a period (.) or a question mark (?) denotes the end of a sentence and the beginning of another sentence, then you are wrong. There can also be instances where the letters of acronyms are separated by full stops, for instance. Various analyses need to be performed at a sentence level; detecting the boundaries of sentences is essential.

Exercise 1.11: Sentence Boundary Detection

In this exercise, we will extract sentences from a paragraph. To do so, we'll be using the sent_tokenize() method, which is used to detect sentence boundaries. The following steps need to be performed:

step 1: Open a Jupyter Notebook.
step 2: Insert a new cell and add the following code to import the necessary libraries

```
import nltk
```

```
from nltk.tokenize import sent_tokenize
```

step 3: Use the sent_tokenize() method to detect sentences in some given text. Insert a new cell and add the following code to implement this:

```
def get_sentences(text):

return sent_tokenize(text)

get_sentences("We are reading a book. Do you know who is "\

"the publisher? It is Packt. Packt is based "\

"out of Birmingham.")
```

step 4: This code generates the following output:

```
['We are reading a book.'
'Do you know who is the publisher?'
'It is Packt.',
'Packt is based out of Birmingham.']
```

step 5: Use the sent_tokenize() method for text that contains periods (.) other than those found at the ends of sentences:

```
get_sentences("Mr. Donald John Trump is the current "\
"president of the USA. Before joining "\
"politics, he was a businessman.")
```

step 6: The code will generate the following output:

```
['Mr. Donald John Trump is the current president of the USA.',

'Before joining politics, he was a businessman.']
```

As you can see in the code, the sent_tokenize method is able to differentiate between the period (.) after "Mr" and the one used to end the sentence. We have covered all the preprocessing steps that are involved in NLP.

Methods

1 Generative Sequence Classification Methods
2 Discriminative Local Classification Methods
3. Hybrid Approaches
4. Discriminative Sequence Classification Methods

5 Extensions for Global Modeling for Sentence Segmentation

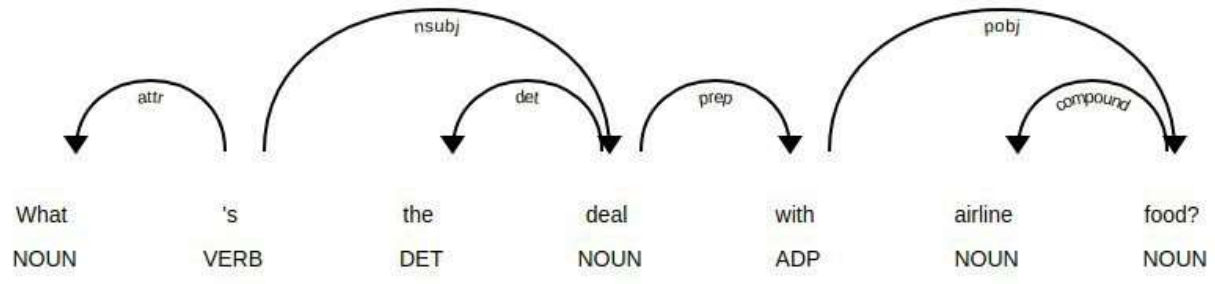
1. Generative Classifiers tries to model class, i.e., what are the features of the class. In short, it models how a particular class would generate input data. When a new observation is given to these classifiers, it tries to predict which class would have most likely generated the given observation.

2. The discriminative model refers to **a class of models used in Statistical Classification, mainly used for supervised machine learning**. These types of models are also known as conditional models since they learn the boundaries between classes or labels in a dataset.

3. A hybrid approach in NLP
NLP, ML, and human input are all part of the hybrid approach, which combines the best of rule-based and machine learning approaches. Human experience guides accurate analysis, but machine learning makes that analysis scale easily.

Approaches to NLP Tasks

There are 3 main groups of approaches to solving NLP tasks.



1. Rule-based

Rule-based approaches are the oldest approaches to NLP. have been proven to work well. Rules applied to text can offer a lot of insight: think of what you can learn about arbitrary text by finding what words are nouns, or what verbs end in -ing, or whether a pattern recognizable as Python code can be identified. Regular expressions and context free grammars are textbook examples of rule-based approaches to NLP.

Rule-based approaches:

- tend to focus on pattern-matching or parsing

- can often be thought of as "fill in the blanks" methods
- are low precision, high recall, meaning they can have high performance in specific use cases, but often suffer performance degradation when generalized

2. "Traditional" Machine Learning

"Traditional" machine learning approaches include probabilistic modeling, likelihood maximization, and linear classifiers.

Traditional machine learning approaches are characterized by:

- training data - in this case, a corpus with markup
- feature engineering - word type, surrounding words, capitalized, plural, etc.
- training a model on parameters, followed by fitting on test data (typical of machine learning systems in general)
- inference (applying model to test data) characterized by finding most probable words, next word, best category, etc.
- "semantic slot filling"

3. Neural Networks

This is similar to "traditional" machine learning, but with a few differences:

- feature engineering is generally skipped, as networks will "learn" important features (this is generally one of the claimed big benefits of using neural networks for NLP)
- instead, streams of raw parameters ("words" -- actually vector representations of words) without engineered features, are fed into neural networks
- very large training corpus

Specific neural networks of use in NLP include recurrent neural networks (RNNs) and convolutional neural networks (CNNs).