

The background features a light gray surface with several faint, overlapping circular patterns. Some of these circles have small tick marks and numbers along their perimeters, resembling a technical or scientific diagram. The numbers are small and light gray, blending into the background. The overall aesthetic is clean and modern.

Natural Language Processing

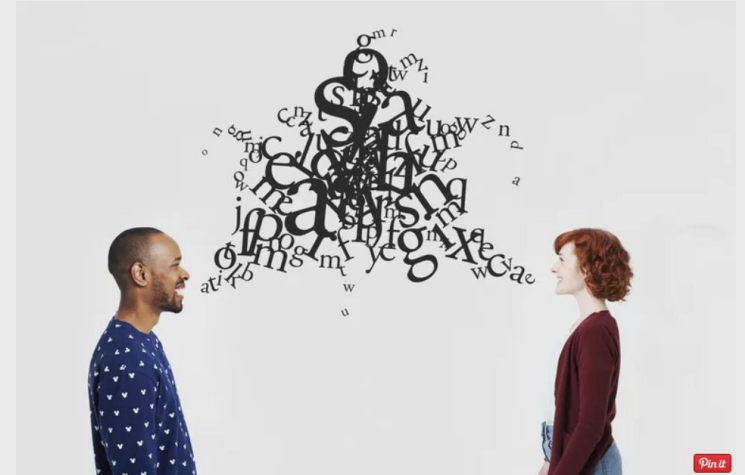
Lecture Outline:

- 1) Background of Natural Language Processing
- 2) NLP applications
- 3) Basic steps in NLP
- 4) NLP terminologies
- 5) Feature extraction in NLP
- 6) ML Model selection for NLP
- 7) Conclusion
- 8) References

Language and NLP

A language, basically is a fixed vocabulary of words which is shared by a community of humans to express and communicate their thoughts.

A *natural language* is a human **language**, such as English. On the other hand, an artificial language is a machine language, or the language of formal logic such as JAVA.



Natural language processing (also known as *computational linguistics*) is the scientific study of language from a computational perspective, with a focus on the interactions between natural (human) languages and computers.

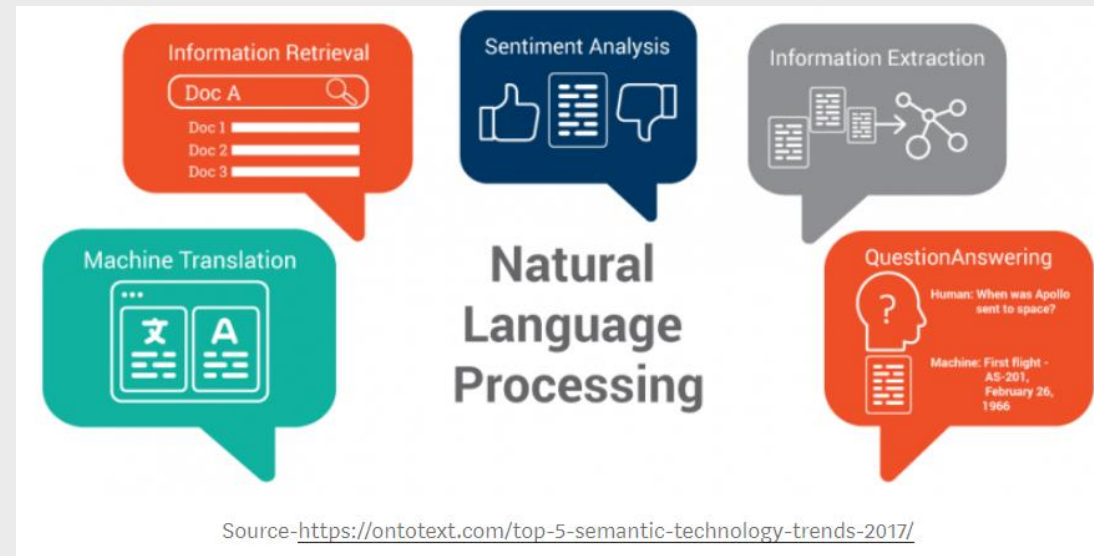
Natural Language Processing, or NLP for short, is broadly defined as the automatic manipulation of natural language, like speech and text, by software.



NLP Applications

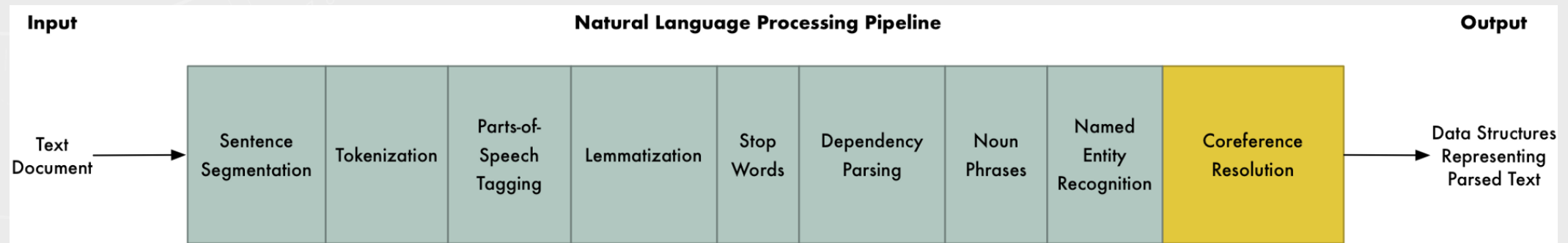
Natural Language Processing usage:

1. **Autocomplete** helps you to suggest rest of the word.
2. Google search's **predictive typing** helps you by suggesting the next word.
3. **Spell checker** in your email application saves you from stupid typing errors.
4. **Spam detection** in your mail box separates spam mails from important ones.
5. Google translator helps you to understand different languages.
6. Sentiment analyzer helps a industry to extract the views of their customer.
7. Speech recognition, voice recognition.
8. Text regeneration and voice regeneration.
9. Topic modeling and text summarization.
10. Recognizing a language from texts.
11. Search Engine
12. Semantic search and web



Basic Steps in NLP

- Step 1:** Sentence Segmentation
- Step 2:** Word Tokenization
- Step 3:** Predicting Parts of Speech for Each Token
- Step 4:** Text Lemmatization
- Step 5:** Identifying Stop Words
- Step 6:** Dependency Parsing
- Step 6b:** Finding Noun Phrases
- Step 7:** Named Entity Recognition (NER)
- Step 8:** Coreference Resolution



Sentence Segmentation

Sentence Segmentation model can be as simple as splitting apart sentences whenever you see a punctuation mark. But modern NLP pipelines often use more complex techniques that work even when a document isn't formatted cleanly.

London is the capital and most populous city of England and the United Kingdom. Standing on the River Thames in the south east of the island of Great Britain, London has been a major settlement for two millennia. It was founded by the Romans, who named it Londinium. London's ancient core, the City of London, largely retains its 1.12-square-mile (2.9 km²) medieval boundaries.

London

- Capital of United Kingdom
- Most populous city in England
- Founded by Romans

1. "London is the capital and most populous city of England and the United Kingdom."
2. "Standing on the River Thames in the south east of the island of Great Britain, London has been a major settlement for two millennia."
3. "It was founded by the Romans, who named it Londinium."

Word Tokenization

To break a sentence into separate words or *tokens* is called word *tokenization*.

Sentence:

“London is the capital and most populous city of England and the United Kingdom.”

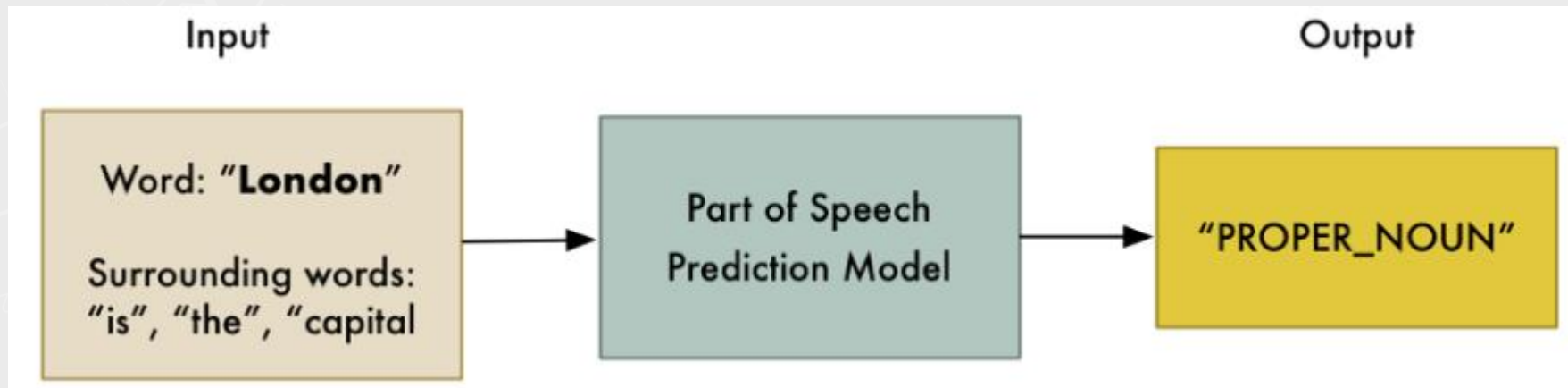
Tokens:

*“London”, “is”, “the”, “capital”, “and”, “most”, “populous”, “city”, “of”, “England”,
“and”, “the”, “United”, “Kingdom”, “.”*

POS Tagging

Inferring the part of speech (whether it is a noun, a verb, an adjective and so on) of a token is called POS tagging. Knowing the role of each word in the sentence will help us start to figure out what the sentence is talking about.

We can do this by feeding each word (and some extra words around it for context) into a pre-trained part-of-speech classification model:



London	is	the	capital	and	most	populous ...
Proper Noun	Verb	Determiner	Noun	Conjunction	Adverb	Adjective

Text Lemmatization

Figuring out the most basic form or lemma of each word in the sentence is called Lemmatization

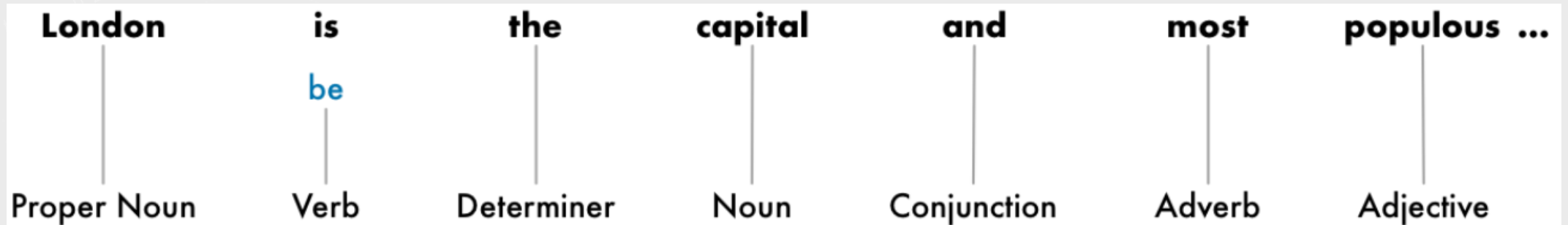
Before lemmatization:

I had a **pony**.
I had two **ponies**.

After Lemmatization:

I [have] a [pony]
I [have] two [pony]

Lemmatization is typically done by having a look-up table of the lemma forms of words based on their part of speech and possibly having some custom rules to handle words that you've never seen before.



The only change we made was turning "is" into "be".

Identifying Stop Words

Words that you might want to filter out before doing any statistical analysis are the stop words.

English has a lot of filler words that appear very frequently like “and”, “the”, and “a”. When doing statistics on text, these words introduce a lot of noise since they appear way more frequently than other words.

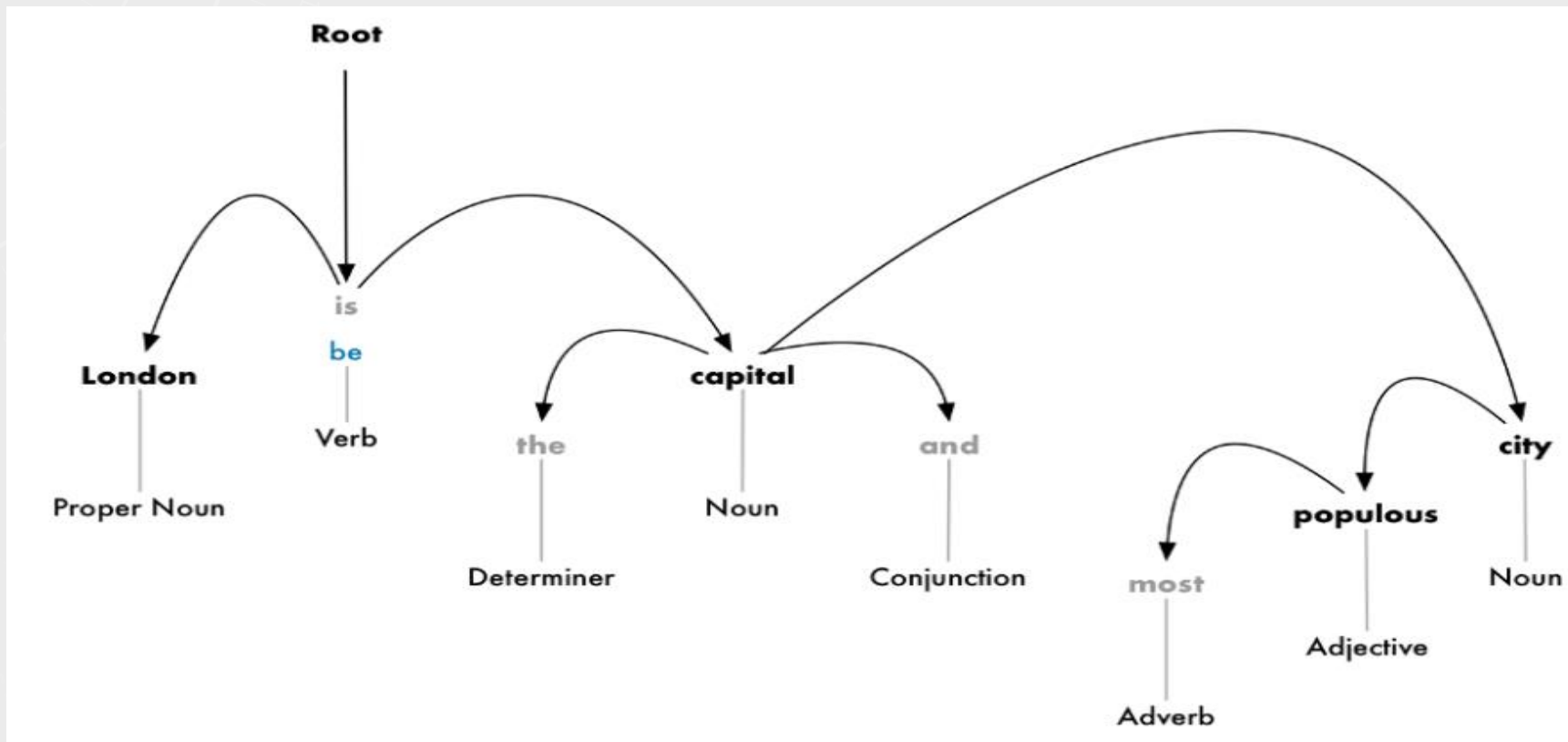
Stop words are usually identified by just by checking a hardcoded list of known stop words. But there’s no standard list of stop words that is appropriate for all applications.

London	is be	the	capital	and	most	populous ...
Proper Noun	Verb	Determiner	Noun	Conjunction	Adverb	Adjective

Dependency Parsing (contd.)

The process of figure out, how all the words in a sentence relate to each other is called *dependency parsing*.

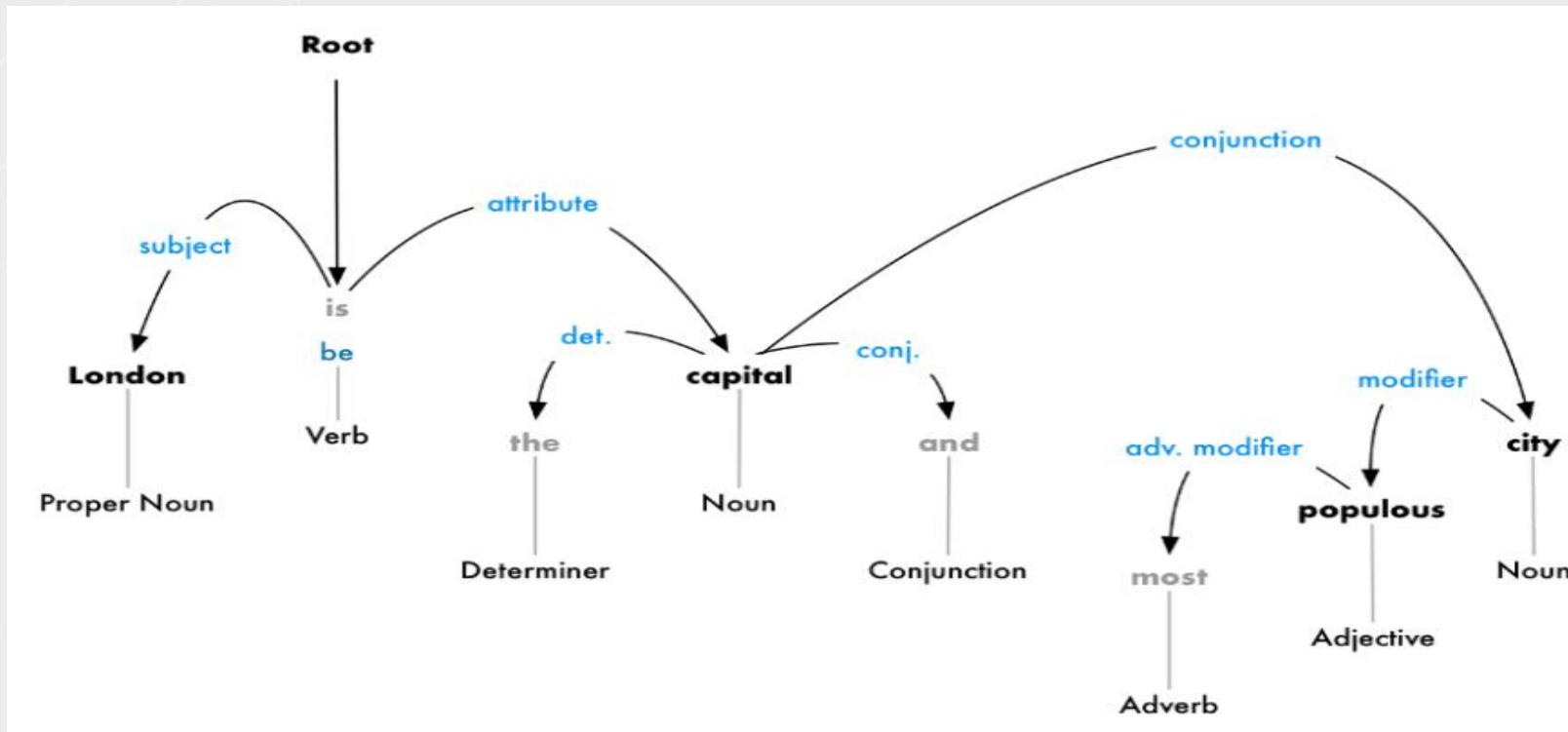
The goal is to build a tree that assigns a single **parent** word to each word in the sentence. The root of the tree will be the main verb in the sentence.



Dependency Parsing (contd.)

In addition to identifying the parent word of each word, the type of relationship that exists between those two words are also predicted:

This parse tree shows us that the subject of the sentence is the noun “*London*” and it has a “*be*” relationship with “*capital*”. We finally know something useful — *London* is a *capital*! And if we followed the complete parse tree for the sentence, we would even found out that London is the capital of the *United Kingdom*



Dependency Parsing (contd.)

Dependency parsing also works by feeding words into a machine learning model and outputting a result.

In 2016, Google released a new dependency parser called *Parsey McParseface* which outperformed previous benchmarks using a new deep learning approach which quickly spread throughout the industry. Then a year later, they released an even newer model called *ParseySaurus* which improved things further. In other words, parsing techniques are still an active area of research and constantly changing and improving.

Some other parsers are:

- Stanford PCFG
- Redshift
- spaCy v0.89
- Stanford NN
- Parser.py

Finding Noun Phrases

The process of grouping together the words that represent a single idea or thing is called finding Noun phrases.

We can use the information from the dependency parse tree to automatically group together words that are all talking about the same thing.



We can group the noun phrases to generate this:



Named Entity Recognition (NER)

The goal of *Named Entity Recognition*, or *NER*, is to detect and label these nouns with the real-world concepts that they represent.

In our sentence, we have the following nouns:

London is the **capital** and most populous **city** of **England** and the **United Kingdom**.

Here's what our sentence looks like after running each token through our NER tagging model:

London is the capital and most populous city of **England** and the **United Kingdom**.

Geographic
Entity

Geographic
Entity

Geographic
Entity

Named Entity Recognition (NER)(Contd.)

NER systems aren't just doing a simple dictionary lookup. Instead, they are using the context of how a word appears in the sentence and a statistical model to guess which type of noun a word represents. A good NER system can tell the difference between "*Brooklyn Decker*" the person and the place "*Brooklyn*" using context clues.

Here are just some of the kinds of objects that a typical NER system can tag:

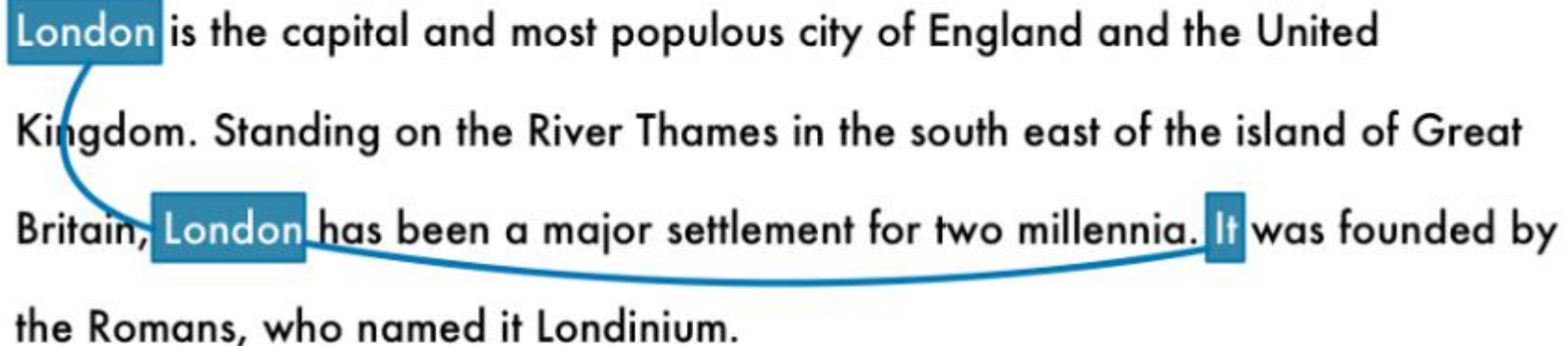
- People's names
- Company names
- Geographic locations (Both physical and political)
- Product names
- Dates and times
- Amounts of money
- Names of events

Coreference Resolution

The goal of coreference resolution is to figure out the mapping by tracking pronouns across sentences. We want to figure out all the words that are referring to the same entity..

English is full of pronouns — words like *he*, *she*, and *it*. These are shortcuts that we use instead of writing out names over and over in each sentence. Humans can keep track of what these words represent based on context. But our NLP model doesn't know what pronouns mean because it only examines one sentence at a time.

Here's the result of running coreference resolution on our document for the word "London":



The diagram illustrates coreference resolution for the word "London". It shows two sentences. In the first sentence, "London" is highlighted with a blue box. A blue line starts from this box and curves down to another blue box highlighting "London" in the second sentence. A second blue line starts from the "London" box in the second sentence and curves to a blue box highlighting the pronoun "It" in the same sentence. This visualizes how the model identifies that both "London" and "It" refer to the same entity.

London is the capital and most populous city of England and the United Kingdom. Standing on the River Thames in the south east of the island of Great Britain, London has been a major settlement for two millennia. It was founded by the Romans, who named it Londinium.

NLP terminologies

Tokenization :

Tokenization is a process to split longer strings into smaller pieces. Large documents can be tokenized into paragraphs, Paragraphs can be tokenized into sentences and sentences can be tokenized into phrases, words or letters.

Corpus or Corpora :

A large structured collection of texts is known as corpus(plural corpora)

Stemming :

Stemming is a process to eliminate affixes (prefix,suffix,infix,circumfix) from a word in order to obtain a word stem or root word.

going -> go , happily -> happy , am/are/is -> be.

Stemming cuts off the end or beginning of the word,taking into account a list of common prefixes and suffixes

Form : Studies Suffix : es Stem : Studi

Form : Studying Suffix : ing Stem : Study

NLP terminologies (Contd.)

Lemmatization:

A common term associated with stemming is Lemmatization. There is a slight difference between stemming and Lemmatization

Lemmatization takes into consideration morphological analysis of the words.

Form : Studies Lemma : Study

Form : Studying Lemma : Study

Lemmatization definitely has an edge over stemming but building a Stemmer is far easier than the latter as deep linguistic knowledge is required to look for the proper form of word.

Stop Words :

Consider words like a, an, the, be etc. These words don't add any extra information in a sentence. Such words can often create noise while modeling. Such words are known as **Stop Words**.

NLP terminologies (Contd.)

Disambiguation :

One of the major challenges in NLP is disambiguation of content. One word can have multiple meanings which at times becomes challenging for the machines to interpret. For example lead can be used in two different contexts

A pencil is made up of **lead**.

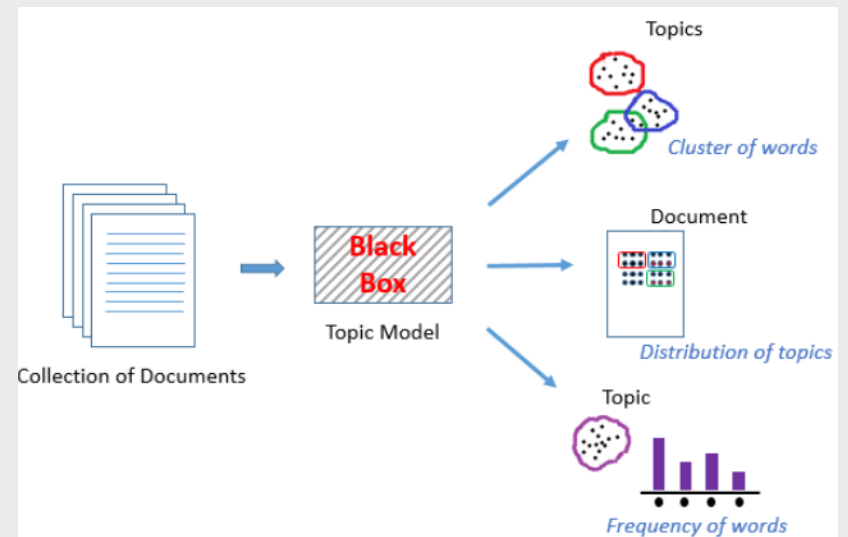
Prime Minister would **lead** the rally on Sunday.

Topic Models :

In machine learning or NLP topic models are type of statistical models which help in discovering the abstract topic that occur in the collection of documents

Word Boundaries:

In written form of languages punctuation marks help us to determine the end of a sentence or paragraph. But in Verbal communication Word Boundary detection plays an important role. Since there is no sign of start of the word, end of the word and number of words in the spoken utterance of any natural language, one must study the intonation pattern of a particular language.



NLP Feature Extraction (BoW)

Bag of Words :

A bag-of-words is a representation of text that describes the occurrence of words within a document.

It involves two things:

- 1) A vocabulary of known words.
- 2) A measure of the presence of known words.

Its called a “*bag*” of words, because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.

NLP Feature Extraction (BOW) [Contd.]

The following models a text document using bag-of-words. Here are two simple text documents:

- (1) John likes to watch movies. Mary likes movies too.
(2) John also likes to watch football games.

Based on these two text documents, a list constructed as follows for each document:

"John", "likes", "to", "watch", "movies", "Mary", "likes", "movies", "too"

"John", "also", "likes", "to", "watch", "football", "games"

Representing each bag-of-words:

```
BoW1 = {"John":1, "likes":2, "to":1, "watch":1, "movies":2, "Mary":1, "too":1};
```

```
BoW2 = {"John":1, "also":1, "likes":1, "to":1, "watch":1, "football":1, "games":1};
```

Creating a dictionary:

```
BoW_Dictionary = {"John":2,"likes":3,"to":2,"watch":2,  
"movies":2,"Mary":1,"too":1,"also":1,"football":1,"games":1};
```

[illegible]

NLP Feature Extraction (BOW) [Contd.]

BoW	John	likes	to	watch	movies	Mary	too	also	football	Games
1	1	2	1	1	2	1	1	0	0	0
2	1	1	1	1	0	0	0	1	1	1
3										
4										
5										

positive

negative

positive

positive

negative

Class Label: Y

Feature Vector: X

NLP Feature Extraction (Tf-idf) [Contd.]

Tf-idf :

Short form for **term frequency-inverse document frequency** is a numerical statistic to define how important a word is to a document in a collection of documents.

Term frequency measures the frequency of a term in a document

$Tf(t)$ = Number of times term t occurs in a document / Total number of terms in a document.

Inverse document frequency (idf) measures how important a term is, whereas in calculating TF all terms are considered equally important.

However certain terms like “and”, “is”, “are” appear a lot of time but have a little importance. Thus, **rare terms should be scaled up and frequent terms should be weighed down.**

$IDF(t)$ = $\log(\text{Total number of documents} / \text{Number of documents having the term } t.)$

Document 1		Document 2	
Term	Term Count	Term	Term Count
this	1	this	1
is	1	is	1
a	2	another	2
sample	1	example	3

$$\text{tf}(\text{"this"}, d_1) = \frac{1}{5} = 0.2$$

$$\text{tf}(\text{"this"}, d_2) = \frac{1}{7} \approx 0.14$$

$$\text{idf}(\text{"this"}, D) = \log\left(\frac{2}{2}\right) = 0$$

$$\text{tfidf}(\text{"this"}, d_1, D) = 0.2 \times 0 = 0$$

$$\text{tfidf}(\text{"this"}, d_2, D) = 0.14 \times 0 = 0$$

$$\text{tf}(\text{"example"}, d_1) = \frac{0}{5} = 0$$

$$\text{tf}(\text{"example"}, d_2) = \frac{3}{7} \approx 0.429$$

$$\text{idf}(\text{"example"}, D) = \log\left(\frac{2}{1}\right) = 0.301$$

$$\text{tfidf}(\text{"example"}, d_1, D) = \text{tf}(\text{"example"}, d_1) \times \text{idf}(\text{"example"}, D) = 0 \times 0.301 = 0$$

$$\text{tfidf}(\text{"example"}, d_2, D) = \text{tf}(\text{"example"}, d_2) \times \text{idf}(\text{"example"}, D) = 0.429 \times 0.301 \approx 0.129$$

REFERENCES

- <https://cloud.google.com/natural-language/>
- <https://cloud.google.com/natural-language/docs/morphology>
- [*Parsing English in 500 Lines of Python* https://explosion.ai/blog/parsing-english-in-python?source=post_page-----](https://explosion.ai/blog/parsing-english-in-python?source=post_page-----)
- Demo on Dependency parsing https://explosion.ai/demos/displacy?source=post_page-----
- Demo on NER https://explosion.ai/demos/displacy-ent?source=post_page-----
- Coreference resolution <https://medium.com/huggingface/state-of-the-art-neural-coreference-resolution-for-chatbots-3302365dcf30>
- Coreference resolution https://huggingface.co/coref/?source=post_page-----
- Natural Language Processing is a Fun <https://medium.com/@ageitgey/natural-language-processing-is-fun-9a0bff37854e>
- Introduction to Natural Language Processing <https://medium.com/greyatom/introduction-to-natural-language-processing-78baac3c602b>
- Natural Language Processing: From Basics to Using RNN LSTM <https://towardsdatascience.com/natural-language-processing-from-basics-to-using-rnn-and-lstm-ef6779e4ae66>