UniTs - University of Trieste

Faculty of Data Science and Artificial Intelligence

Department of mathematics informatics and geosciences

Natural Language Processing

Lecturer:
Prof. Alberto Cazzaniga

Author:
Christian Faccio

September 30, 2025

github.com/christianfaccio christianfaccio@outlook.it

Preface

As a student of Data Science and Artificial Intelligence, I've created these notes while attending the **Natural Language Processing** course.

The course provides a comprehensive introduction to the field of natural language processing, covering both theoretical concepts and practical applications. The notes encompass a variety of topics, including:

- ML and DL Fundamentals
- Tokenization and Text Preprocessing
- Word Embeddings
- RNN for NLP
- Transformers and Attention Mechanisms
- Language Models (e.g., BERT, GPT)
- Understanding LLMs
- Visutal Language Models

While these notes were primarily created for my personal study, they may serve as a valuable resource for fellow students and professionals interested in natural language processing.

Contents

1 Tokenization

1.1 Basics

NLP deals with computer readable text.

Problem. computer readable text is of variable format:

1. ”This is an interesting lecture”
2. ”This lecture is cool :)#nlp proc”
3. ”This lecture is also inter-interesting”
4. The same sentence written in chinese, japanese, arabic, etc.

There is the need for **standardization**. Regular Expressions ($\sim$ 1950 Kleen) are a way to describe patterns in strings.

Example: *Find all sentences starting with ”the”*

1. `/the/` $\rightarrow$ matches ”the” anywhere in the string (but not ”The”)
2. `/[Tt]the/` $\rightarrow$ matches ”the” or ”The”
3. `/[Tt]he` $\rightarrow$ **MISSING:** explanation
4. `/[â-zA-Z][Tt]he[â-zA-Z]/` $\rightarrow$
5. Solve the fact that this misses beginning or end of a line

*For more information on Regular Expressions look at [1].

Basic entities Computer readable text is collected in **Corpora**.

Definition: *Corpus*

The **Corpus** is a collector of computer readable text (or speech). See for example a document, a novel, ...

Corpora may vary for different elements:

- Length: # bytes to store it;
- Language: phonemic (It, En, ...), syllabic (Arabic, Japanese), morphosyllabic (Chinese):
  - Different languages with same character system;
  - Same language, same character systems but different dialect or slang (ex. ”I don’t” $\rightarrow$ ”Iont”);
- Genre: written and spoken genre;
- Demographic characteristics;
- Time;

Standardization comes with **Datasheets** (T.Gebru) or **Data Statement** (Bendor). Important is the *motivation* for that collection, the *situation*, the *language-variety*, *speaker/writer demographics*, the *collection process* and the *annotation*. This is done to make it **reproducible** for others.

The unit we would like to focus on are **WORDS**. A word is the single distinct meaningful element of speech or writing, typically shown with spaces on either side when written or printed and saw with others in a portion of text.

Again, **problems** arise:

- Punctuation: usually considered as a word (",", ".", ":", ...);
- Utterance: like "uh", "um", "ah", ...;

Actual elementary parts of a corpus will be defined/deduced but he corpus itself.

Word Types are the distinct words in a corpus ($\mathscr{V}$ is the vocaboulary). Capitalization is sometimes considered and sometimes not, depends on the situation. **Word Instances** represent a measure of length, indicating how many total words in a corpus there are (N). **Word Counting** indicates how many times a single and distinct word is repeated in a corpus.

	$\lvert\mathscr{V}\rvert$	N
Shakespeare	31k	884k
Google n-grams	13M	1B
Wikipedia	?	4.9B

Definition: ***Herdant's Law***

$|\mathscr{V}| = k|\mathscr{C}|^{\beta}$ with $k > 0$ and $0 < \beta < 1$.

A big problem is the cardinality of the vocaboulary, making tasks like next token prediction almost impossible to solve.

1. A first solution is to reduce the vocaboulary size by using **Stemming** or **Lemmatization**: Stemming reduces words to their base or root form (e.g., "running" $\rightarrow$ "run"), while lemmatization considers the context and converts words to their meaningful base form (e.g., "better" $\rightarrow$ "good").
2. A better solution is to use **Subword Tokenization**, breaking words into smaller units (subwords) that can be combined to form words. This approach helps manage the vocabulary size while still capturing meaningful components of words. Examples include Byte Pair Encoding (BPE) and WordPiece.

Definition: ***BPE***

Byte-Pair-Encoding is a procedure that, given a new sentence, divides it into subwords and then encodes it as a sequence of tokens. The algorithm starts with a base vocabulary of characters and iteratively merges the most frequent pairs of tokens to create new subword tokens, until a predefined vocabulary size is reached.
Example: corpus is made of (A,B,D,C,A,B,E,C,A,B).
- (AB D C AB E C AB), $\mathscr{V} = \{A,B,C,D,E,AB\}$;
- (AB D CAB E CAB), $\mathscr{V} = \{A,B,C,D,E,AB,CAB\}$;
- ...
- **k** times;

Once we have that, how do we perform **inference**?

- Start from characters ($|\mathscr{C}_{test}|$)
- Merge them into subwords
- Merge subwords into other subwords
- ...

Distance The usual measure of distance used in NLP is the **Edit Distance** (or Levenshtein distance). It is defined as the minimum number of operations required to transform one string into another, where the allowed operations are insertion, deletion, or substitution of a single character.

Bibliography

[1] Daniel Jurafsky et al. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*.