Deep Learning

Natural Language Processing

Hugo Boulanger

M2 BDMA, CentraleSupélec, Université Paris Saclay

October 26, 2024

Plan

Resources and Disclaimer

- This course is based on my experience as an NLP Researcher on what is necessary to understand the current landscape of NLP.
- I have a computer engineering background, and as such, I will not cover linguistics in depth.
- If you want to both broaden and deepen your knowledge of NLP, you can take a look Speech and Language Processing from [Jurafsky and Martin, 2024].
- This course is an evolved version of Julien Denize and Tom Dupuis's 2022 course.

Why do we use Natural Language Processing (NLP) ?

- Languages are used for a **lot of different things**:
  - communicate: oral or written.
  - think: words shape our mind.
  - make science (maths, computer science, ...).
  - ...
- NLP is used on **various tasks**:
  - Communicate (translation, ASR, ...).
  - Make requests (search engine, recommender system, generative models).
  - Linguistics and Cognitives Sciences (Analyze language).
  - Code (Automatic code generation).
  - ...

Data for Natural Language Processing

- Theoretically, the whole textual data from the web can be used.
- This represents an **enormous amount** of data.
  - 1 billion website online
  - around 60M Wikipedia pages in several languages
- Data is **increasing** at a fast rate:
  - 9000 tweets / seconds
  - 3M mails / second (60% are spam)
- Have a look yourself: `https://www.internetlivestats.com`

Business interest for Natural Language Processing

- Search engines, with 2+billions of users for Google
- Social media with 3+billions of users (Facebook, Twitter, Instagram, TikTok, ...)
- Voice assistants with 100M+ users (Google Nest, Cortana, Alexia, Siri, ...)
- Translations (DeepL, Google translate, ...)
- Grammar check (Grammarly, Reverso, ...)

Natural Language Processing: definition and difficulty

Natural Language Processing

Natural language processing (NLP) is the field of science and engineering that studies computational approaches to **understand and generate human languages**.

> *"It's difficult to extract sense from strings, but they're the only communication coin we can count on. [...] Within a computer, natural language is unnatural." Alan Perlis*

IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.

Challenges of NLP: Evolution

- Language is in **constant evolution** through time.
- New words (ex: youtuber, tiktoker, ...), new grammar, new structure (ex: SMS texts).
- NLP models should **keep-up the evolution** of language which is quite difficult and require constant training.

Challenges of NLP: Ambiguity

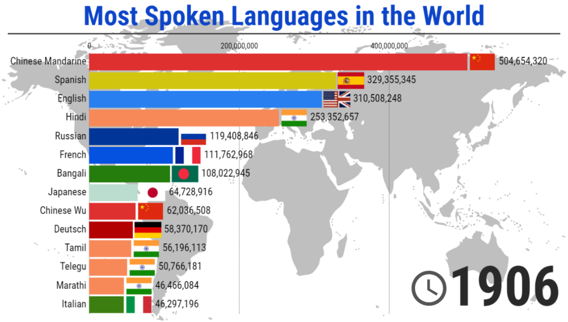
- Language is ambiguous, for humans and even more so for machine:
  - Meanings **depend** on the context.
  - Spelling errors.
  - ...
- Humans can ask for the correct interpretation or use **global context** (linguistic or not) to better communicate whereas machines cannot for each task.

Challenges of NLP: Variation

- What does **variability** affect ?
  - Accent: British, indian or american
  - Spelling: color ou colour ?
  - Syntactic and Semantic, ...
- Variability comes from **several factors**:
  - **Social context**: different meaning of words depending the social class.
  - **Geography**: variation given the country.
  - **Date**: evolution of the language through time.
  - **Topic**: different meanings of words.

Challenges of NLP: Diversity

- 7000+ languages spoken in the world
- 60% of these languages exist in written form:
  - no written data for 40% of them.
  - how to deal with few data ?
  - learning a language A can be really hard in comparison with a language B



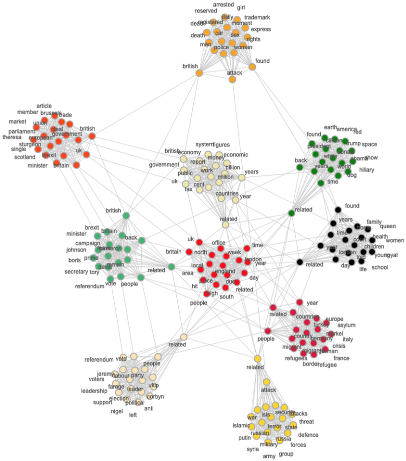
Plan

Corpus

Corpus

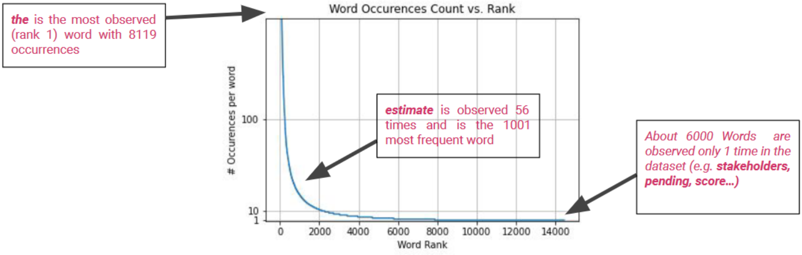
A collection of written texts, especially the entire works of a particular author or a body of writing on a particular subject.

- In corpuses, some words come out more than others
- We can easily find **interactions** between words by:
    - calculating their co-occurrences.
    - constructing topic graphs.
    - ...



Word distribution in a Corpus

- Word distribution of a 800 scientific articles corpus.



Statistical Description of a Corpus

- In a large corpus, the word distributions follows the **Zipf Law**
- For f_W the frequency of the word w and k the frequency rank of the word w

$$f_W(k) \propto \frac{1}{k^\theta}$$

- Most frequent words are **exponentially more frequent** than less frequent words.
- Consequences:
  - **Sparsity** among words: a lot of words have few occurences.
  - Some very important words are less present than non-important words (such as 'a', 'the').
  - This make NLP tasks more difficult to learn.

Token

Token

*"A **token** is an instance of a sequence of characters in some particular document that are grouped together as a useful **semantic unit** for processing" Stanford*

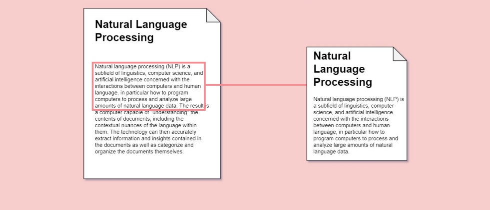
- A **semantic unit** in our case can be:
  - A word or sub-word (ex: OMG)
  - A character
  - Any sequence of characters (words, multiple words, sub-words, ...)

Document

Document

*A **document** is a sequence of semantics units.*

- The definition of document depends on the task and context and can have various length.
- Exemple of document types:
  - A tweet of 140 characters.
  - A book.
  - ...



Corpus for NLP

Corpus

*A **Corpus** is a collection of several documents.*

- It serves as an **input** for training a model:
  - Various size (large, medium, small), the bigger the better !
  - It can be **unbalanced**: fake news vs trustworthy news
  - It can be **messy**: typos, artefacts, ...
- It serves to various NLP tasks:
  - sentiment analysis.
  - topic extractor.
  - ...

Tokenization

Tokenization

***Tokenization** consists in cutting into pieces a document in **tokens**.*

- Some characters can be **thrown away** sometimes like punctuation or whitespaces.
- What are the tokens to use ?
  - Some words might not be important: "the".
  - Some words only appear once.
  - What about apostrophes: "O'neill" becomes "o" + "neill" or "o'" + "neill" or ...
- Tokenization is **language specific**.
- A class of token is sometimes called a **type**.
- The different types kept form the **vocabulary** or **index** of the corpus.

Language Modeling

Language Modeling

Language Modeling is the process of assigning probabilities to the next word(s) (or, in general the next token(s)).

Language Model

A language model, is a model assigning probabilities to the possible next word.

Given tokens, how do we build a language model?

Language Modeling: the maths

N-gram

An N-gram is a tuple of N words (ordered).

- For w_n the n^{th} word in the sentence:
  $P(w_{1:n}) = P(w_1)P(w_2|w_1)...P(w_n|w_{1:n-1})$
  $P(w_{1:n}) = \prod_{i=1}^{n} P(w_i|w_{1:i-1})$
- We can approximate using the Markov assumption (N-gram):
  $P(w_n|w_{1:n-1}) = P(w_n|w_{n-N+1:n-1})$
- Now : $P(w_{1:n}) = \prod_{i=1}^{n} P(w_i|w_{n-N+1:i-1})$

N-gram Models

N-gram models build their probabilities by counting the occurrences of the N-grams:

$$P(w_n|w_{n-N+1:n-1}) = \frac{C(w_{n-N+1:n})}{\sum_w C(w_{n-N+1:n-1}w)}$$

The denominator is equal to the count of the N-1-gram

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

Word Embeddings

Word Embeddings

Word vectors are vectors representing words. They can also be called **features or representations of words**.

- Word Embeddings allow for a dense representation of words instead of discrete values from tokenization.
- Tokens are at word-level (can be sub-word level, as we'll see later)

Prediction-based model

- **Learn dense vectors** to represent words through an **embedding matrix** later used to **convert** tokens into vectors.
- **Distributional Hypothesis**: use context to build the vectors.
- Parametrize words as dense vectors.
- Use parametrization to **predict the context** and learn the representation.

Self-Supervised Word2vec

- Word2vec **maps** each word to a vector of a certain dimension.
- It can be made from **two models** [Mikolov et al., 2013] detailed after:
  - Skip-gram.
  - Continuous Bag of Words (CBoW).
- The training seeks to **predict words in the context**, defined as surrounding words, from a word.
- Therefore Word2vec **does not require labels** and uses its own data for supervision, also called self-supervised learning.

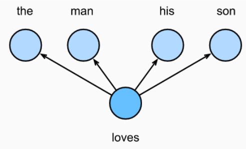
Skip-Gram

- Take the sentence: "The", "man", "loves", "his", "son".
- The center word is defined as "loves" and the context window is 2 (all other words for this specific sentence).
- Skip-Gram seeks to predict the conditional probability to **generate** the context:

$$P(\text{"}the\text{"}, \text{"}man\text{"}, \text{"}his\text{"}, \text{"}son\text{"} | \text{"}loves\text{"})(1)$$

- It assumes that context words are samples independently, therefore:

$$(1) = P(\text{"}the\text{"}|\text{"}loves) \cdot P(\text{"}man\text{"}|\text{"}loves) \cdot P(\text{"}his\text{"}|\text{"}loves) \cdot P(\text{"}son\text{"}|\text{"}loves)$$



Skip-Gram

- Each word w_i is associated to **two learned** representations:
  - $v_i \in \mathbb{R}^d$ **center** word vector
  - $u_i \in \mathbb{R}^d$ **context** word vector
- The probability to generate the context word w_o **conditionally** to the word w_c from the vocabulary index set $\mathcal{V} = \{0, 1, \ldots, |\mathcal{V}| - 1\}$, is defined as the following **softmax** operation:

$$P(w_o|w_c) = \frac{exp(\mathbf{u_o^T v_c})}{\sum_{i \in \mathcal{V}} exp(\mathbf{u_i^T v_c})}$$

- All words used in denominator are considered as **negatives** that the true context word, **positive**, should be contrasted with.
- Contrastive learning in images took inspiration from this !

Skip-Gram

- The probability to generate the context word w_o **conditionally** to the word w_c is:

$$P(w_o|w_c) = \frac{exp(\mathbf{u_o^T v_c})}{\sum_{i \in \mathcal{V}} exp(\mathbf{u_i^T v_c})}$$

- For a context window of size m the **likelihood function** is the probability to generate all context words from a text sequence of length T as follows:

$$\prod_{t=1}^{T} \prod_{-m \leq j \leq m,\ j \neq 0} P(w^{t+j}|w^t)$$

- For training we seek to **maximize the log-likelihood function** of the skip-gram model which is equivalent to minimizing:

$$-\sum_{t=1}^{T} \sum_{-m \leq j \leq m,\ j \neq 0} \log P(w^{t+j}|w^t)$$

Skip-Gram: training

- For training we seek to **maximize the log-likelihood function** of the skip-gram model which is equivalent to minimizing:

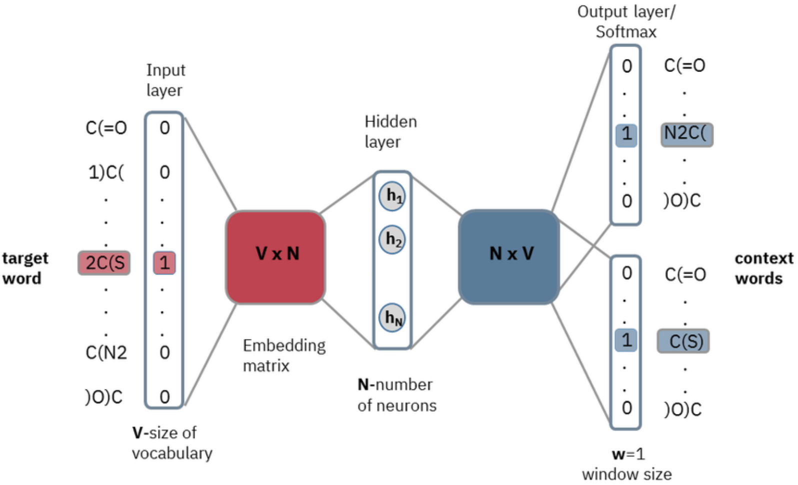
$$-\sum_{t=1}^{T} \sum_{-m \leq j \leq m,\ j \neq 0} \log P(w^{t+j} | w^t)$$

- We need to compute each log conditional probability for any pair w_c and w_o:

$$\log P(w_o | w_c) = \mathbf{u_o^T v_c} - \log \left(\sum_{i\mathcal{V}} exp(\mathbf{u_i^T v_c}) \right)$$

- To train using **gradient-descent**, we compute the **partial derivatives**: $\frac{\partial \log P(w_o|w_c)}{\partial v_c}$, $\frac{\partial \log P(w_o|w_c)}{\partial u_o}$, $\forall j$, $\frac{\partial \log P(w_o|w_c)}{\partial u_j}$

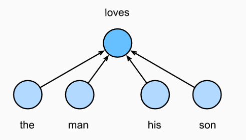
Skip-Gram: illustration



Continuous Bag of Words (CBOW)

- Similar to Skip-Gram model.
- Continuous Bag of Words (CBOW) seeks to predict the conditional probability to **generate** the center word given its context is:

$$P(\text{"}loves\text{"} | \text{"}the\text{"}, \text{"}man\text{"}, \text{"}his\text{"}, \text{"}son\text{"})$$



- The conditional probability to generate the word w_c given its context $w_{o_1}, \dots, w_{o_{2m}}$ is:

$$P(w_c | w_{o_1}, \dots, w_{o_{2m}}) = \frac{\exp(\frac{1}{2m} \mathbf{u_c^T} (\mathbf{v_{o_1}} + \dots + \mathbf{v_{o_{2m}}}))}{\sum_{i \in \mathcal{V}} \exp(\frac{1}{2m} \mathbf{u_i^T} (\mathbf{v_{o_1}} + \dots + \mathbf{v_{o_{2m}}}))}$$

Continuous Bag of Words (CBOW)

- The conditional probability to generate the word w_c given its context $w_{o_1}, \ldots, w_{o_2m}$:

$$P(w_c | w_{o_1}, \ldots, w_{o_{2m}}) = \frac{\exp(\frac{1}{2m}\mathbf{u_c^T}(\mathbf{v_{o_1}} + \cdots + \mathbf{v_{o_{2m}}}))}{\sum_{i \in \mathcal{V}} \exp(\frac{1}{2m}\mathbf{u_i^T}(\mathbf{v_{o_1}} + \cdots + \mathbf{v_{o_{2m}}}))}$$

- Let's define $v_o = \frac{1}{2m}(v_{o_1} + \cdots + v_{o_{2m}})$ then:

$$P(w_c | w_{o_1}, \ldots, w_{o_2m}) = \frac{\exp(\mathbf{u_c^T v_o})}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u_i^T v_o})}$$

Continuous Bag of Words (CBoW) training

- For a context window of size m the likelihood function is the probability to **generate** the center word given its context from a text sequence of length T, and defined as follows:

$$\prod_{t=1}^{T} P(w^t | w^{t-m}, \ldots, w^{t-1}, w^{t+1}, \ldots, w^{t+m})$$

- For training we seek to maximize the log-likelihood function of the CBOW model which is equivalent to minimizing:

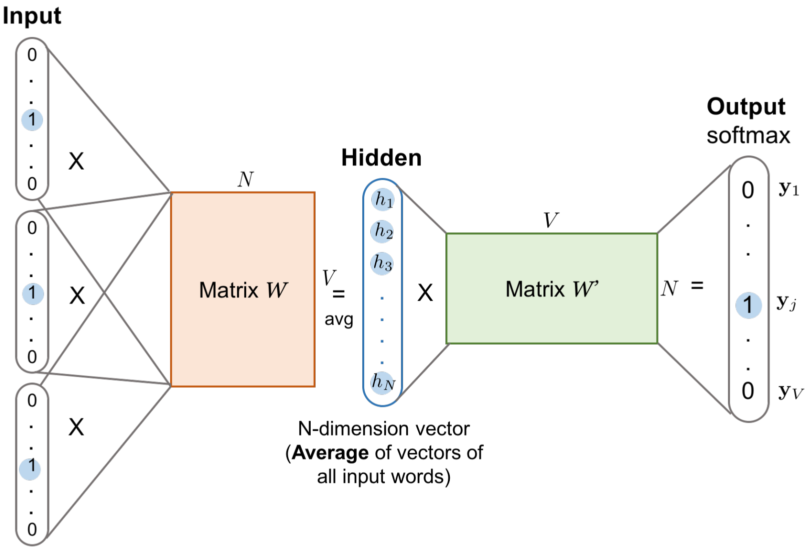
$$-\sum_{t=1}^{T} \log P(w^t | w^{t-m}, \ldots, w^{t-1}, w^{t+1}, \ldots, w^{t+m})$$

with

$$\log(P_{w_c} | w_{o_1}, \ldots, w_{o_2 m}) = \mathbf{u_c^T v_o} - \log\left(\sum_{i \in \mathcal{V}} exp(\mathbf{u_i^T v_o})\right)$$

- To train using **gradient-descent** we compute the partial derivatives.

CBoW: illustration



Training Word2vec

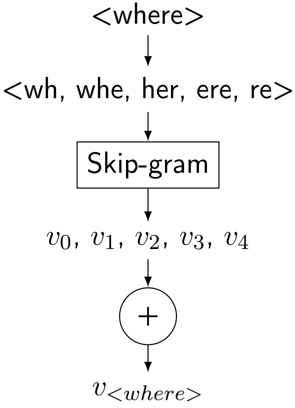
- Word2vec output is the matrix embedding from **either** Skip-gram or CBoW.
- In practice CBoW is quicker to train but Skip-gram has better results.
- Improvements:
  - **Negative sampling**: draw negatives from a predefined distribution.
  - **Hierarchical softmax**: uses binary tree.
  - **Global Statistics**: Additional use of global statistics led to GloVe[Pennington et al., 2014]
- Problems:
  - **Fixed Vocabulary**: unknown tokens are not treated.
  - **Fixed Context**: static embeddings don't react to context.

Solving Fixed Vocabulary

Words

Words are made of sub-words.

- Words are sequences of **n-grams of characters**.
- You can learn a model on this sequence instead.
- You obtain word representation by summing the sub-word representations.
- This technique is called **Fast-Text** [Bojanowski et al., 2016]



Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016).
Enriching word vectors with subword information.
arXiv preprint arXiv:1607.04606.

Jurafsky, D. and Martin, J. H. (2024).
Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models.
Stanford University, 3rd edition.
Online manuscript released August 20, 2024.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013).
Distributed representations of words and phrases and their compositionality.
In *27th Annual Conference on Neural Information Processing Systems*, pages 3111–3119.

Pennington, J., Socher, R., and Manning, C. D. (2014).
Glove: Global vectors for word representation.
In *EMNLP*, pages 1532–1543.