



Deep Learning

Natural Language Processing

Hugo Boulanger

M2 BDMA, CentraleSupélec, Université Paris Saclay

October 26, 2024



Plan

- 1 Introduction
- 2 Introduction to NLP concepts
- 3 NLP in the age of Large Language Models



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).
 - ...

```
26 .screen-reader-text:hover,  
27 .screen-reader-text:active,  
28 .screen-reader-text:focus {  
29   background-color: #f1f3f4;  
30   border-radius: 3px;  
31   box-shadow: 0 0 2px 2px rgba(0, 0, 0, 0.6);  
32   clip: auto !important;  
33   color: #212529;  
34   display: block;  
35   font-size: 14px;  
36   font-size: 0.875rem;  
37   font-weight: bold;  
38   height: auto;  
39   left: 5px;  
40   line-height: normal;  
41   padding: 15px 23px 14px;  
42   text-decoration: none;  
43   top: 5px;  
44   width: auto;  
45   z-index: 100000; /* Above WP toolbar. */  
46 }  
47
```






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, ...)

Is  grammarly
the Right Tool for You?



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





Challenges of NLP: Evolution

- Language is in **constant evolution** through time.
- New words (ex: youtuber, tiktok, ...), 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.





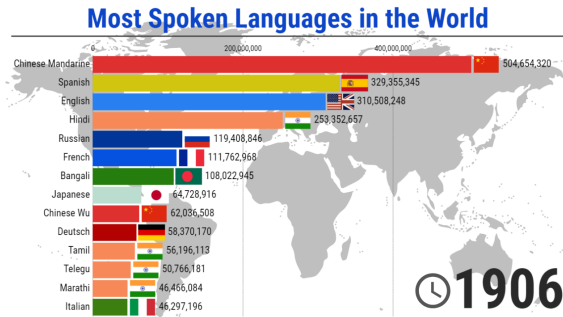
Challenges of NLP: Variation

- 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

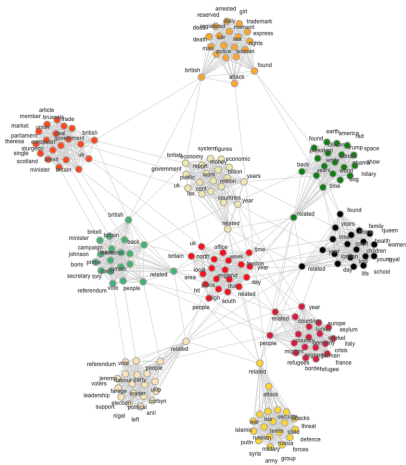
- 1 Introduction
- 2 Introduction to NLP concepts
- 3 NLP in the age of Large Language Models

Corpus

Corpus

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

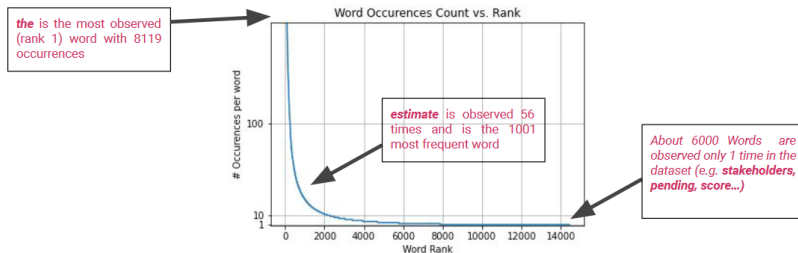
- In corpora, some words come out more than others
- We can easily find **interactions** between words by:
 - calculating their co-occurrences.
 - constructing topic graphs.
 - ...





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 occurrences.
 - 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*

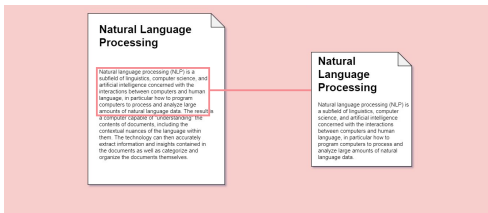
- A **semantic unit** in our case can be:
 - A word or sub-word (ex: OMG)
 - A character
 - Any sequence of characters (words, multiple words, sub-words, ...)

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.



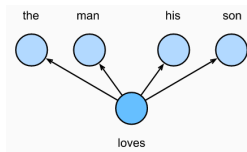
Skip-Gram

- 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:

$$P("the", "man", "his", "son" | "loves")(1)$$

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

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





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 $\mathcal{V} = \{0, 1, \dots, |\mathcal{V}| - 1\}$, is defined as the following **softmax** operation:

$$P(w_o|w_c) = \frac{\exp(\mathbf{u}_o^T \mathbf{v}_c)}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^T \mathbf{v}_c)}$$

- 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:

$$P(w_o|w_c) = \frac{\exp(\mathbf{u}_o^T \mathbf{v}_c)}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^T \mathbf{v}_c)}$$

- 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 :

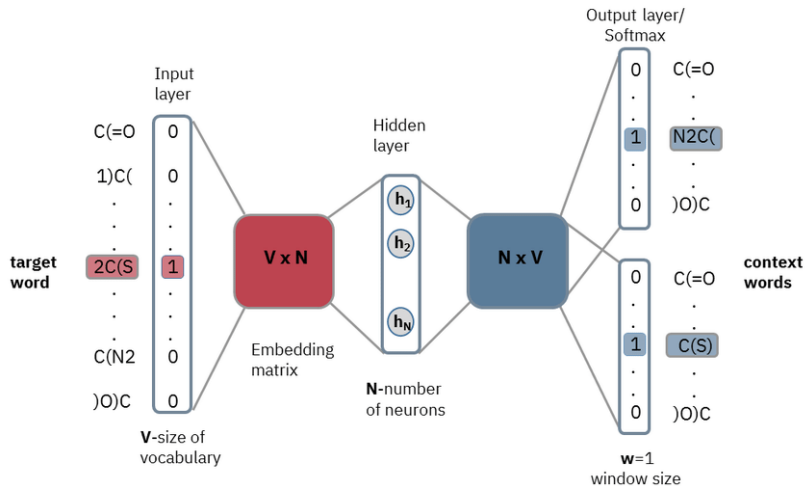
$$\log P(w_o | w_c) = \mathbf{u}_o^T \mathbf{v}_c - \log \left(\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^T \mathbf{v}_c) \right)$$

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

$$\frac{\partial \log P(w_o | w_c)}{\partial v_c}, \frac{\partial \log P(w_o | w_c)}{\partial u_o}, \forall j, \frac{\partial \log P(w_o | w_c)}{\partial u_j}$$



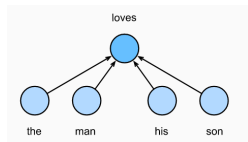
Skip-Gram: illustration



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:

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



- 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 $w_{o_1}, \dots, w_{o_{2m}}$:

$$P(w_c | w_{o_1}, \dots, w_{o_{2m}}) = \frac{\exp(\frac{1}{2m} \mathbf{u}_c^T (\mathbf{v}_{o_1} + \dots + \mathbf{v}_{o_{2m}}))}{\sum_{i \in \mathcal{V}} \exp(\frac{1}{2m} \mathbf{u}_i^T (\mathbf{v}_{o_1} + \dots + \mathbf{v}_{o_{2m}}))}$$

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

$$P(w_c | w_{o_1}, \dots, w_{o_{2m}}) = \frac{\exp(\mathbf{u}_c^T \mathbf{v}_o)}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^T \mathbf{v}_o)}$$



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:

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

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

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

with

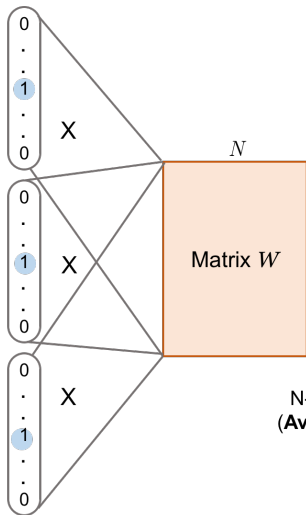
$$\log(P_{w_c} | w_{o_1}, \dots, w_{o_{2m}}) = \mathbf{u}_c^T \mathbf{v}_o - \log \left(\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^T \mathbf{v}_o) \right)$$

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

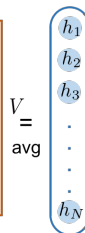


CBoW: illustration

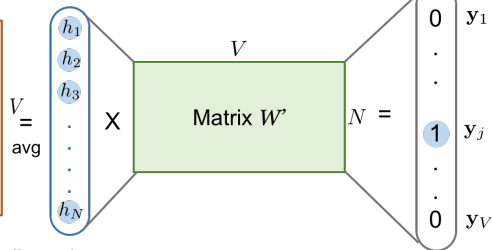
Input



Hidden



N-dimension vector
(**Average** of vectors of
all input words)

Output
softmax



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.

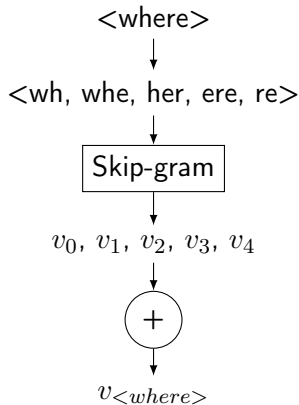


Solving Fixed Vocabulary

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]





Plan

- 1 Introduction
- 2 Introduction to NLP concepts
- 3 NLP in the age of Large Language Models



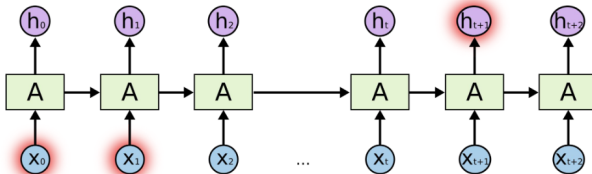
Deep Learning architectures

- To make **full use context**, a Deep Learning model needs a dedicated architecture.
- Historically it used all types of architectures with different success:
 - **MultiLayer Percetron**: quickly discarded as it lacks temporal information.
 - **Convolutional Neural Networks**: Not the best for NLP tasks.
 - **Recurrent Neural Networks**: Suitable for NLP and improved over previous architectures but lacks the **attention mechanism** to allow a better use of context.
 - **Transformers**: widely used today in NLP.
- The model learns word representation through different objectives such as for Transformers:
 - **Language modeling**:
$$P(x_1, \dots, x_T) = \prod_{t=1}^T P(x_t | x_1, \dots, x_{t-1}) .$$
 - Next sentence prediction.
 - Masked word prediction.

Recurrent Neural Network

- Recurrent Neural Network introduce a **recurrence relation** into an MLP.
- Trained by **Backpropagation Through Time (BPTT)**.
- Limits: trouble to capture **long-term dependencies**.

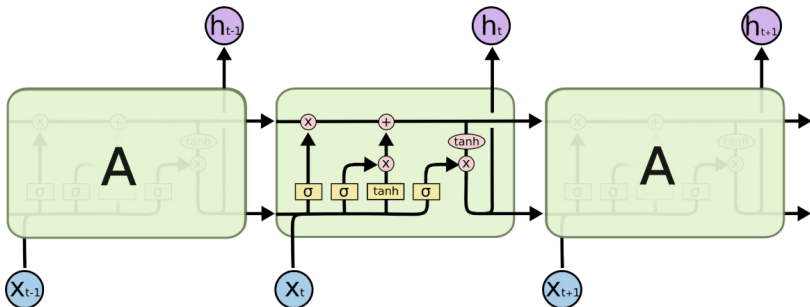
Illustration of a 1-layer Recurrent Neural Network





LSTM: Long-Short Term Memory Network

- Introduce a **memory vector** to capture long term dependencies.
- It also introduces a **forget gate** to remove some irrelevant part of the memory.
- Problem: interpret which input has an impact on the output.

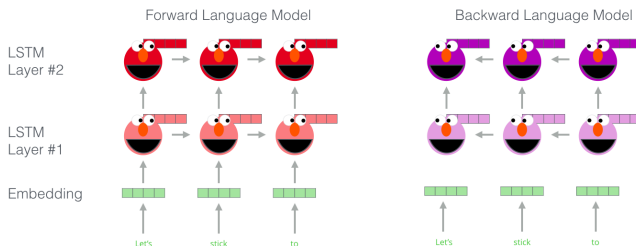




ELMo

- The input of ELMo, [Peters et al., 2018], comes from a pretrained **embedding layer**.
- ELMo takes information from context by looking at the whole sentence before assigning an embedding to solve a task.
- It uses a **bidirectional** LSTM, meaning it looks from right to left and from left to right to compute the representation.

Embedding of "stick" in "Let's stick to" - Step #1





ELMo

- It aggregates embeddings computed from the forward and backward language models that predict next words.
- For each **specific task**, a supervised architecture takes ELMo features as input, plus word embeddings from GloVe for example, to be trained.

Embedding of “stick” in “Let’s stick to” - Step #2

1- Concatenate hidden layers



2- Multiply each vector by a weight based on the task

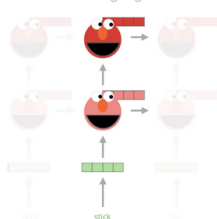


3- Sum the (now weighted) vectors

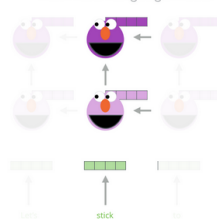


ELMo embedding of “stick” for this task in this context

Forward Language Model



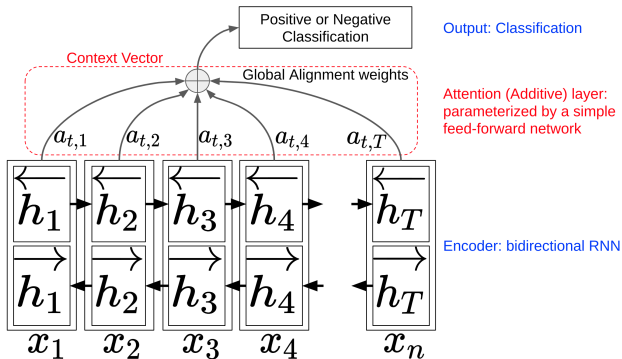
Backward Language Model





Attention mechanism

- The **attention mechanism** learn **weights** on top of hidden representations to know which hidden representation impact the final prediction.
- It can be used on top of RNN, LSTM, etc.

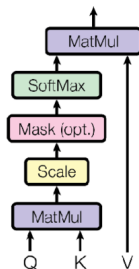




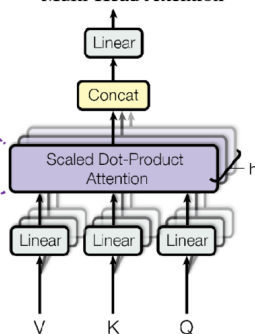
Attention mechanism

$$\text{attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Scaled Dot-Product Attention



Multi-Head Attention

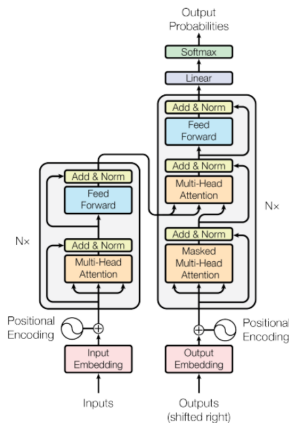


You build the attention weights with the queries (Q) and the keys (K), and apply them to the values (V).



Transformers

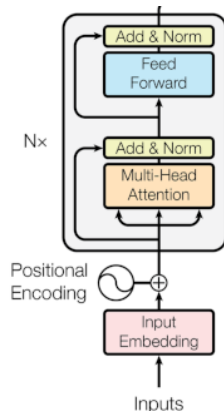
- *Attention is all you need*, [Vaswani et al., 2017]
- Use only the **attention mechanism** in the model architecture.
- Widely used transformer encoder-decoder architecture.





Transformers: encoder

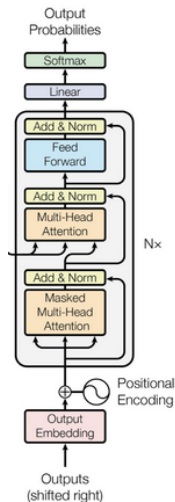
- Input: the word embeddings from a sentence.
- Stack of multi-head attention and positionwise feed-forward networks.





Transformers: decoder

- Input: the output embeddings from the already generated sentence.
- Injection of the encoder outputs.
- Stack of (masked) multi-head attention and feed-forward networks.
- Masked Multi Self Attention for the tokens not already generated.





Generative Pre-Training (GPT)

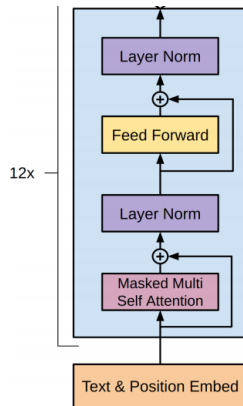
- GPT, [Radford et al., 2018], is a transformer-based architecture that seeks to learn a **task-agnostic model** with context-sensitive representations.
- Its architecture is built on a **transformer decoder**.
- Two-steps:
 - Pre-training the whole model.
 - Fine-tuning of all parameters for each specific task.

GPT: pretraining

- GPT is a generative model, it seeks to maximize the following **language modeling objective** for the corpus of tokens $\mathcal{U} = \{u_1, \dots, u_n\}$:

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \dots, u_{i-1}; \Theta)$$

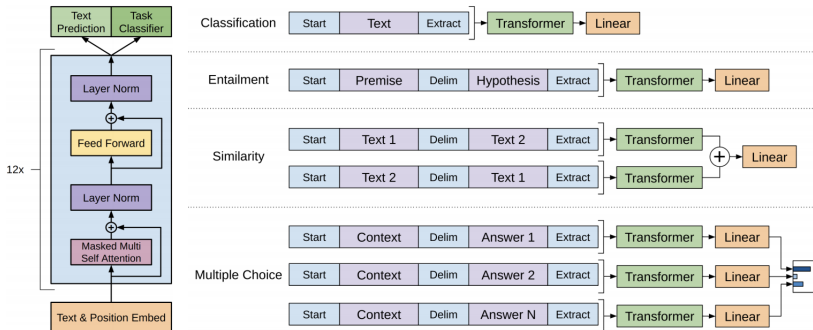
- It is an **auto-regressive model** thanks to the masked multi-head attention layers.
- It predicts representations sequentially from left to right.





GPT: finetuning

- For each specific task, the GPT pretrained model is finetuned.



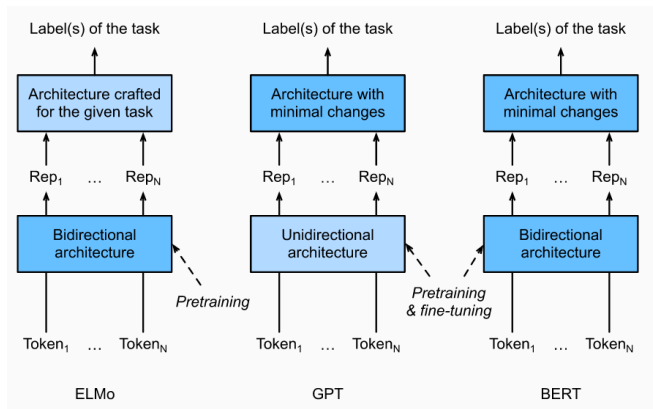


Limitations from previous methods

- ELMo encodes **bidirectionally** which allows a better use of context.
- However ELMo uses task-specific architectures.
- GPT is **task-agnostic** and is easily fine-tuned to various tasks.
- However GPT is an autoregressive model and cannot have representations sensitive to the **full context** of words.

BERT: combining the best of both worlds.

- BERT, [Devlin et al., 2019], stands for **Bidirectional Encoder Representations from Transformers**.
- It encodes context bidirectionally and its whole model is finetuned for specific tasks.





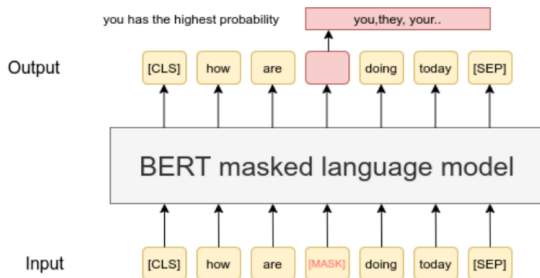
BERT: pretraining tasks

- As GPT, BERT first pretrain its representations that are later fine-tuned for specific tasks.
- BERT has **two different pretraining tasks**:
 - Masked Language Modeling.
 - Next Sentence Prediction
- It is based on the transformer encoder architecture.



BERT: Masked Language Modeling

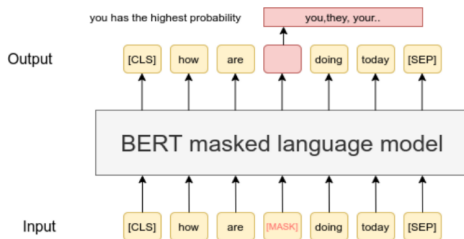
- In this pretraining stage, the goal of BERT is to predict some masked tokens from its bidirectional context.
- 15% of all tokens are selected randomly and masked by placing the token "<mask>" instead of the right token.





BERT: Masked Language Modeling

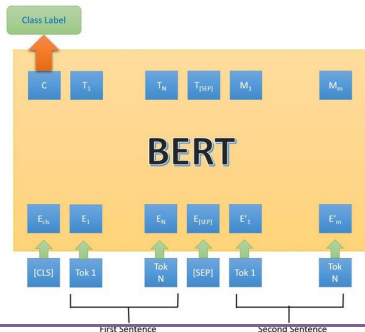
- Because the token "<mask>" does not exist in finetuning and to avoid **discrepancy** between pretraining and finetuning, the input is sometimes replaced by other tokens following these proportions:
 - 80% the "<mask>" special token is used.
 - 10% a **random token** is inserted to avoid overfitting on the "<mask>" token.
 - 10% the **unchanged label**.





BERT: Next Sentence Prediction

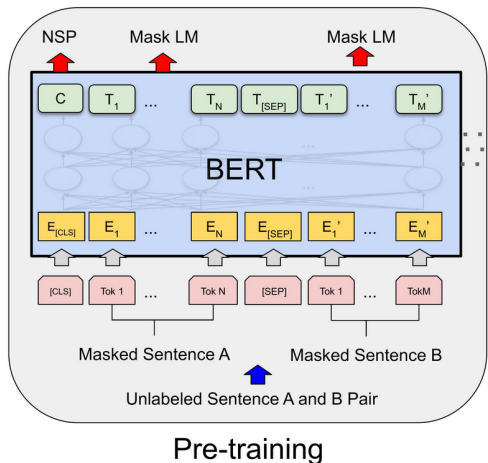
- Masked Language Modeling lacks the **logical relation** between text pairs.
- BERT adds a **next sentence prediction** objective.
- A pair of sentences are given: sometimes **consecutive** (True), sometimes **random** (False).
- The next sentence prediction is a **binary classification task** learned thanks to the "<cls>" token.





BERT: pretraining

- BERT learns both Masked Language Modeling and Next Sentence Prediction during pretraining.





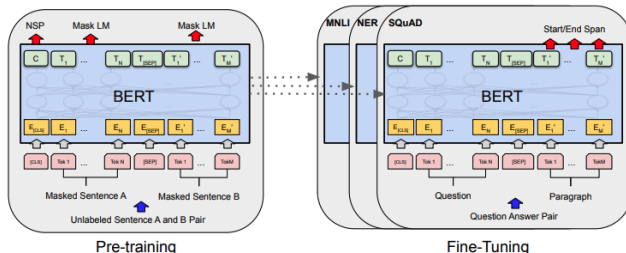
How to improve ?

- NLP progress did not stop at BERT.
- Some models with better language modeling math background have been proposed such as MPNet, [Song et al., 2020].
- Increasing the model size and the dataset greatly improve results.
 - For example, new GPT versions are often released that only enhance the size of the datasets and models.



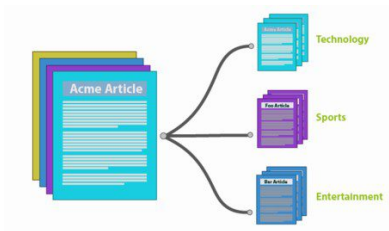
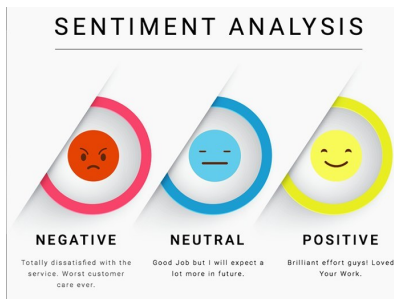
Make our transformer pretrained models useful

- We have seen various models to learn word embeddings with or without context representations.
- We now want to use our models on **various tasks**.
- For that we need to model the task we want to deal with and use our models **to learn to solve this task**.
- GPT and BERT have the advantage to be able to transfer to specific tasks with minor architecture changes.



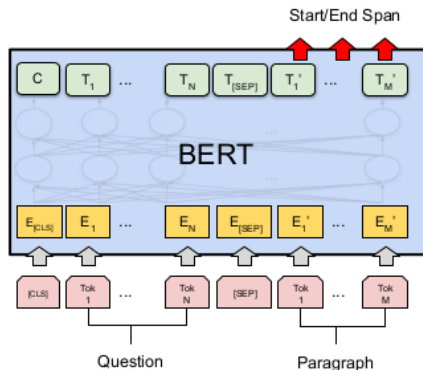
Classification tasks

- Classification tasks cover several subjects such as **document classification**, **sentiment analysis**.
- It can be learned via a "<cls>" token in transformer based architecture just like for next sentence prediction in BERT.



Question Answering

- Question Answering parameters take as an input a question and **generates an answer**.
- It requires the model to know **two more tokens**:
 - "<sep>" between question and answer.
 - "<end>" the end of the generated answer.



Named Entity Recognition

- **Named Entity Recognition (NER)** seeks to retrieve various named entities from a text.
- Using BERT, NER model can be trained by feeding the output representation of each word to classification layer.

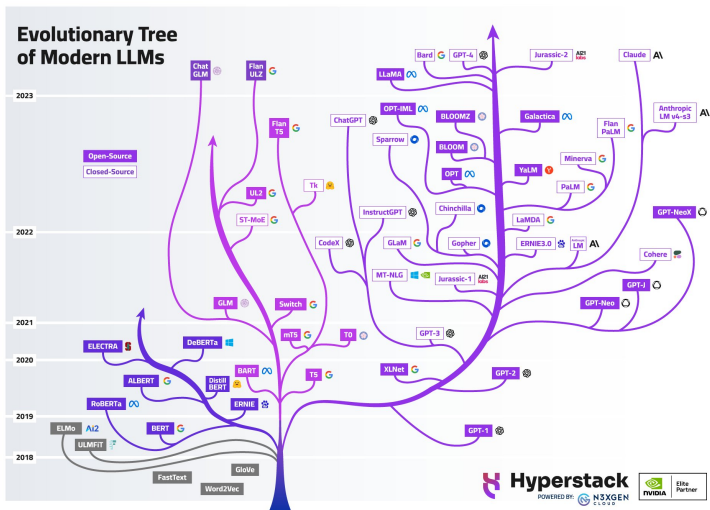
In fact, the **Chinese** **NORP** market has the **three** **CARDINAL** most influential names of the retail and tech space – **Alibaba** **OPE**, **Baidu** **ORG**, and **Tencent** **PERSON** (collectively touted as **BAT** **ORG**), and is betting big in the global **AI** **OPE** in retail industry space. The **three** **CARDINAL** giants which are claimed to have a cut-throat competition with the **U.S.** **OPE** (in terms of resources and capital) are positioning themselves to become the 'future **AI** **PERSON** platforms'. The trio is also expanding in other **Asian** **NORP** countries and investing heavily in the **U.S.** **OPE** based **AI** **OPE** startups to leverage the power of **AI** **OPE**. Backed by such powerful initiatives and presence of these conglomerates, the market in APAC AI is forecast to be the fastest-growing **one** **CARDINAL**, with an anticipated **CAGR** **PERSON** of **45%** **PERCENT** over **2018 - 2024** **DATE**.

To further elaborate on the geographical trends, **North America** **LOC** has procured **more than 50%** **PERCENT** of the global share in **2017** **DATE** and has been leading the regional landscape of **AI** **OPE** in the retail market. The **U.S.** **OPE** has a significant credit in the regional trends with **over 65%** **PERCENT** of investments (including M&As, private equity, and venture capital) in artificial intelligence technology. Additionally, the region is a huge hub for startups in tandem with the presence of tech titans, such as **Google** **ORG**, **IBM** **ORG**, and **Microsoft** **ORG**.



Large Language Models in 2024

Today's LLM landscape is dominated by Decoders (GPT-likes).









Explaining the dominance of Decoders

- Higher context window vs Encoders/Encoder-Decoders.
- Few-shot learning [Brown, 2020]:
 - Reduces the need for Fine-tuning/adding layers.
 - Every tasks can be seen as a language modeling task.
 - Prompts/Instructions to condition the generation on the task.



Conclusion

- We discussed basics of NLP:
 - definition of tokens, documents, corpus
 - tokenization procedure
- We learned about Word embeddings
 - Learned representation with Word2vec and FastText
- We introduced context in word representations with modern models:
 - ELMo, GPT, BERT
- We studied for some tasks how to finetune modern models to solve them:
 - Classification, Sentiment analysis, Q&A, ...

-  Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016).
Enriching word vectors with subword information.
arXiv preprint arXiv:1607.04606.
-  Brown, T. B. (2020).
Language models are few-shot learners.
arXiv preprint arXiv:2005.14165.
-  Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019).
BERT: pre-training of deep bidirectional transformers for
language understanding.
In *Proceedings of the 2019 Conference of the North American
Chapter of the Association for Computational Linguistics:
Human Language Technologies, NAACL-HLT*, pages
4171–4186.
-  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.



Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018).



Deep contextualized word representations.

In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 2227–2237.



Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018).

Improving language understanding by generative pre-training.
In *OpenAI*.



Song, K., Tan, X., Qin, T., Lu, J., and Liu, T. (2020).

Mpnet: Masked and permuted pre-training for language understanding.

In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems*.



Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017).



Attention is all you need.

In *Annual Conference on Neural Information Processing Systems 2017*, pages 5998–6008.