**Recherche bibliographique**

*par Sylvain*ligne horizontale

NLP = Natural Language Processing

Ressources

* *Deep Learning book* section 12.4 page 456 | [lien](http://www.deeplearningbook.org/contents/applications.html)
* *CS224N: Natural Language Processing with Deep Learning | Winter 2019* | [lien](https://www.youtube.com/playlist?list=PLoROMvodv4rOhcuXMZkNm7j3fVwBBY42z)
* *A Simple Introduction to Natural Language Processing* | [lien](https://becominghuman.ai/a-simple-introduction-to-natural-language-processing-ea66a1747b32)
* *Understanding word vectors* | [lien](https://gist.github.com/aparrish/2f562e3737544cf29aaf1af30362f469)
* *Word2Vec – a baby step in Deep Learning but a giant leap towards Natural Language Processing* | [lien](https://medium.com/explore-artificial-intelligence/word2vec-a-baby-step-in-deep-learning-but-a-giant-leap-towards-natural-language-processing-40fe4e8602ba)

NLP : use of human languages by a computer. Example: translation from English to French.

language model: define a probability distribution over sequences of words, characters, or bytes (in a natural language) → use techniques that are specialized for processing sequential data. Because the total number of possible words is so large, word-based language models must operate on an extremely high-dimensional and sparse discrete space.

While humans can easily master a language, the ambiguity and imprecise characteristics of the natural languages are what make NLP difficult for machines to implement.

NLP entails applying algorithms to identify and extract the natural language rules such that the unstructured language data is converted into a form that computers can understand.

1) *n*-grams

language model = probability distribution over sequences of tokens (in a natural language)

token = word, a character, or even a byte

*n*-grams = fixed-length sequences of tokens

Models based on *n*-grams define the conditional probability of the *n*-th token given the preceding *n-1* tokens.

*n=1* → unigram, *n=2* → bigram, *n=3* → trigram ; Greek suffix “-gram” denoting something that is written (think of “grammar”)

→ limitation because zero probability, much information is lost

2) NLMs = Neural Language Models

NLM = class of language model designed to overcome the curse of dimensionality problem for modeling natural language sequences by using a distributed representation of words → able to recognize that two words are similar without losing the ability to encode each word as distinct from the other

word embeddings = we view the raw symbols as points in a space of dimension equal to the vocabulary size. In the embedding space, words that frequently appear in similar contexts are close to each other.

Neural networks in other domains also define embeddings. For example, a hidden layer of a convolutional network provides an “image embedding.”

3) High-Dimensional Outputs

In many natural language applications, we often want our models to produce words (rather than characters) as the fundamental unit of the output.

*… inutile de continuer à creuser dans le cadre de ce projet*

Syntactic analysis and semantic analysis are the main techniques used to complete Natural Language Processing tasks.

* Syntax refers to the arrangement of words in a sentence such that they make grammatical sense.
* Semantics refers to the meaning that is conveyed by a text.

Semantic analysis is one of the difficult aspects of NLP that has not been fully resolved yet. It involves applying computer algorithms to understand the meaning and interpretation of words and how sentences are structured. Here are some techniques in semantic analysis:

* Named entity recognition (NER): It involves determining the parts of a text that can be identified and categorized into preset groups. Examples of such groups include names of people and names of places.
* Word sense disambiguation: It involves giving meaning to a word based on the context.
* Natural language generation: It involves using databases to derive semantic intentions and convert them into human language.

Named Entity Recognition (NER) → identify named entities in text into predefined categories, such as person names, organizations, locations, date and time, etc.

Word embedding is one of the most popular representation of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc. Word2Vec is one of the most popular technique to learn word embeddings using deep neural network. It was developed by Tomas Mikolov in 2013 at Google.

Word2vec is word embedding which consider similarity comes from neighbor words. Voir le cours de Parisa Rastin sur [Word2Vec](https://drive.google.com/drive/folders/1XXei8aOiW-CXWY12mcRtK-8ard0rTNDc) slide 36.

PGxCorpus = a Manually Annotated Corpus for Pharmacogenomics

Ressources

* article scientifique sur [bioRxiv](https://www.biorxiv.org/content/10.1101/534388v3)

Remarques

* PGxCorpus contient **10 classes** décrites sur la figure 4 page 9.
* On essaie d'améliorer les résultats de la table 11 page 14 avec (Bio)BERT. Pour des raisons de facilité, on ne s'intéresse qu'à la première colonne *exact no* : les autres variables sont trop difficiles à calculer → on ne regarde que les résultats exacts et sans hiérarchie. La hiérarchie signifie qu’une *genomic\_variation* est aussi une *genomic\_factor*, ainsi si on a l’annotation *genomic\_factor* au lieu de *genomic\_varation*, ce n’est pas entièrement faux.
* Dans le cadre de ce projet, il n’est pas utile de creuser l’article trop profond : il explique la méthode d’annotation.
* Dans la page 3, “nested” signifie “imbriqué” et “discontiguous” signifie “discontinu”. Imbriqué est lorsqu’un mot a une annotation au sein d’une phrase qui a elle-même une annotation. Dans l’exemple, “l’effet du médicament et du traitement”, le mot “traitement” est relié au mot “effet” mais de manière discontinue parce qu’il y a des termes entre les deux. **Dans le cadre de ce projet, on ne prend pas compte les *nested* et les *discontiguous***, ainsi le type *gene\_or\_protein* est le plus simple car a priori il n’a pas de *nested* ni *discontiguous*. Il faut réparer nous-mêmes les *nested* tandis que les *discontiguous* sont identifiables par les points-virgules.
* Le code pour la baseline est en Lua avec Torch, c’est l'équivalent de PyTorch pour Python.
* Pourquoi le choix du format BRAT ? Il existe un outil pour annoter en ligne et de manière collaborative, ce qui rend les choses simples.

BERT = Bidirectional Encoder Representations from Transformers

Ressources

* [présentation](https://fr.wikipedia.org/wiki/BERT_(traitement_automatique_du_langage)) courte sur Wikipedia
* [dépôt](https://github.com/google-research/bert) original sur GitHub
* [article](https://arxiv.org/abs/1810.04805) scientifique original sur arXiv
* [post](https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html) sur le blog de Google
* *How BERT leverage attention mechanism and transformer to learn word contextual relations* | [post](https://towardsdatascience.com/how-bert-leverage-attention-mechanism-and-transformer-to-learn-word-contextual-relations-5bbee1b6dbdb) sur le blog Towards Data Science

Remarques

* **BERT est pré-entraîné**, c’est-à-dire que dans l’architecture du modèle, certains paramètres sont déjà fixés. Le fine-tuning signifie modifier ces paramètres pré-entraînés, l'intérêt est qu’on ne part pas de rien. Comme les modèles de la baseline contiennent également une partie pré-entraînée, notre score sera comparable.
* Dans le cadre de ce projet, il n’est pas utile de creuser l’article BERT trop profond : il utilise des” transformeurs” qui sont complexes (méthode d’attention pour les RNN).

BERT use transformer architecture to learn the text representations. BERT use bidirectional transformer (both left-to-right and right-to-left direction) rather than dictional transformer (left-to-right direction).

2 phases training is applied in BERT. Using generic data set to perform first training and fine tuning it by providing domain specific data set. In pre-training phase, sentences are retrieved from BooksCorpus (800M words) (Zhu et al., 2015) and English Wikipedia (2500M words).

BioBERT

Ressources

* Solving BioNLP problems using Bert (BioBert Pytorch) | [GitHub](https://github.com/MeRajat/SolvingAlmostAnythingWithBert/tree/master/biobert_ner)
* *BioBERT: a pre-trained biomedical language representation model for biomedical text mining* | [article](https://arxiv.org/abs/1901.08746) scientifique sur arXiv
* *Some examples of applying BERT in specific domain* | Towards Data Science | [lien](https://towardsdatascience.com/how-to-apply-bert-in-scientific-domain-2d9db0480bd9)

BioBERT follow BERT model architecture which is multi bidirectional transformer and learning text representation by predicting masked token and next sentence. You can notice that BioBERT outperform BERT on domain specific dataset.