ENS Data Challenge : POSOS

Hurault Samuel
Ecole Normale Superieure Paris-Saclay
samuel.hurault@ens-paris-saclay.fr

Matthieu Cordonnier
Ecole Normale Superieure Paris-Saclay
matthieu.cordonnier@ens-paris-saclay.fr

1. Introduction

The goal of the challenge is to predict from a medical question the associated intent among 51 different classes. The main difficulty of the challenge is the fact that the input sentences contain a lot of spelling errors, abbreviations, and sometimes very technical medical vocabulary. The overall architecture of our algorithm is described in Figure 1.

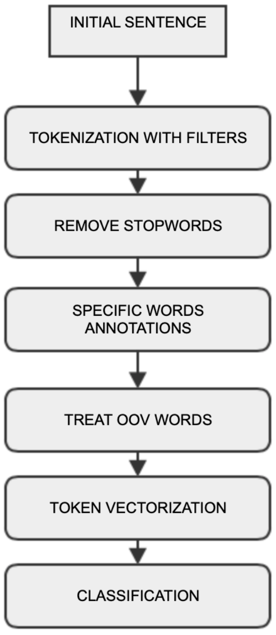


Figure 1. Algorithm structure

2. Data preprocessing

2.1. First preprocessing steps

The raw data is a list of questions written in French. The preprocessing of the data starts by "cleaning" the sentences. First by removing "stop words", punctuation and numbers. Then we replace technical words from the medical lexical field by generic words.

- **Remove stop words** : a stop word is a word which is so common that it is useless for classification. In French some stop words could be the articles "le", "la", "ce" etc... The pertinence of a word is evaluated through its distribution among a corpus of texts : if the distribution is uniform, the word is a stop word. In our case we use a list of French general stop words from the library nltk. We don't use any list of medical stop words because we think that in that specific problem, medical terms are discriminant for our classification.

- **Specific word annotations** : Some words belonging to a same concept are not useful for the classification task because they are too specific, only the common concept between those words is useful. We replace all those similar words by the same general term it refers to. First, we scrap a list of meds provided by VIDAL [1] and use it to annotate all the med names with the word "médicament". We use a corpus of medical annotations from [6] to replace every word relative to the human body by "anatomie". The rest of the annotation corpus was too general to be used in our context. Finally, we replace every medical unit of measurement (mg, ml etc..) by "posologie", unit of time by "heure" and word concerning menstruation by "regle".

- **Lemmatisation** : Lemmatisation is the process of grouping together the inflected forms of a word, replaced by the word's canonical form (or lemma in lexicography). In practice, we use the library FrenchLefffLemmatizer.

2.2. Out of Vocabulary words

Embedding

The first layer of our models will be a embedding layer. This layer assigns to each token w of a sequence a unique vector of dimension d. Considering the low amount of data available, a common strategy is to load an embedding previously trained. We found two different trained embeddings

[1] https://www.vidal.fr/Sommaires/Medicaments-A.htm

in French : [1],[3]. We keep the second one [3] because it has a lexical size of around 150000 words compared to 100000. It is trained on the FrWac corpus with a skip-gram model. This embedding gives us a lexicon of reference that we call embedding dictionary. We also tried to use the fastText embedding (with 2000000 words) but we did not have to finish the OOV module treatment with this embedding.

OOV module

We want to minimize the number of Out Of Vocabulary words (OOV) : a word in our data not in the embedding dictionary. The number of OOV words obtained after the different preprocessing steps are shown in Table 1. As a comparison, the training corpus contains 8452 different words.

	number of OOVs
Initial sentence	3209
Annotate meds	2283
Annotate anatomy	2272

Table 1. Number of OOVs during preprocessing

Therefore almost one quarter of the sentences will not have a pretrained word embedding. We create an OOV module that is going to correct a maximum number of unknown tokens. We notice that a lot of OOVs correspond to spelling mistakes or accent mistakes. Our OOV algorithm is going to replace a word by another combining formal similarity and embedding similarity. A formal similarity between two words can be calculated thanks to the Damerau–Levenshtein distance (DL distance) [7] : the minimum number of operations (consisting of insertions, deletions or substitutions of a single character, or transposition of two adjacent characters) required to change one word into the other. The OOV module treats an input word as follows :

- If the word is on the embedding dictionary, don't treat the word.
- If the word, with accents removed, is in the embedding dictionary from which we removed all the accents. Return the corresponding word from the embedding dict with accent.
- List all the words in the embedding dictionary at maximum DL distance 2 from the input word
  - If no word at DL distance $\leq$ 2, don't treat the word.
  - Among these words, we return the word that appear the most in the training corpus. If it does not appear in our corpus, return the word at minimal DL distance (in case of equity, return the first one to appear in the embedding dictionary i-e the most frequent in the initial embedding training corpus).

2.3. Data preprocessing examples

- **Input sentences**

- "laroxyl à doses faibles pour le stress ?"
- "A partir de combien de temps la pilule optilova est-elle efficace ?"
- "mon psy me dit de prendre 50mg de sertraline le matin et 50 mg de sertraline le soir. peut on prendre 100mg soit le matin ou a midi?"

- **Tokenization**

- 'laroxyl', 'à', 'doses', 'faibles', 'pour', 'le', 'stress'
- 'a', 'partir', 'de', 'combien', 'de', 'temps', 'la', 'pilule', 'optilova', 'est', 'elle', 'efficace'
- 'mon', 'psy', 'me', 'dit', 'de', 'prendre', 'mg', 'de', 'sertraline', 'le', 'matin', 'et', 'mg', 'de', 'sertraline', 'le', 'soir', 'peut', 'on', 'prendre', 'mg', 'soit', 'le', 'matin', 'ou', 'a', 'midi'

- **Remove Stopwords**

- 'laroxyl', 'doses', 'faibles', 'stress'
- 'partir', 'combien', 'temps', 'pilule', 'optilova', 'efficace'
- 'psy','dit', 'prendre', 'mg', 'sertraline', 'matin', 'mg', 'sertraline', 'soir', 'peut', 'prendre', 'mg', 'matin', 'midi'

- **Annotations**

- 'médicament', 'doses', 'faibles', 'stress'
- partir', 'combien', 'temps', 'pilule', 'médicament', 'efficace'
- 'psy', 'dit', 'prendre', 'posologie', 'médicament', 'matin', 'posologie', 'médicament', 'soir', 'peut', 'prendre', 'posologie', 'matin', 'midi'

- **OOV correction**

- 'médicament', 'dose', 'faible', 'stress'
- 'partir', 'combien', 'temps', 'pilule', 'médicament', 'efficace'
- 'psy', 'dit', 'prendre', 'posologie', 'médicament', 'matin', 'posologie', 'médicament', 'soir', 'peut', 'prendre', 'posologie', 'matin', 'midi'

3. Data augmentation

The main challenge for accurate classification is the low amount of training data. We only have 8028 sentences. Moreover, there is strong class imbalance in the training set as shown in 2. The average number of elements per class is 157.4, the median is 72. Two of the classes have even less than 10 elements when there is one huge class with 1700 elements.

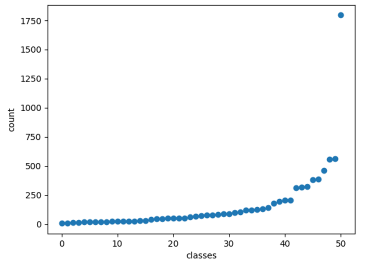


Figure 2. Number of training sentences in each class

Synonims replacement

A first straightforward strategy is to replace some words in the sentence with their synonym. To get a word synonym we use the Wolf corpus [2]. For an input word, we can access to a list of possible synonyms. From this list we select the synonym which is the most similar in the embedding space from the input word : the similarity is the cosinus similarity. ‘ For example, the word "douleur" gives the synonym list [’peine’, ’casse-pied’, ’emmerdeur’, ’inconvénient’, ’abattement’, ’affliction’, ’chagrin’, ’deuil’, ’regret’, ’inconfort’, ’souffrance’, ’nuisance’, ’tendresse’, ’malheur’, ’misère’]. The closest proposition in the embedding space is "souffrance".

For each sentence, we can create new sentences of same length with each word created like this : with probability $1 - p$, we keep the same word, and probability p, we take the replace it by the synonym.

we give some examples of this strategy on training sentences with $p = 0.5$:
- ’médicament’, ’nausée’, ’saignement’ $\longrightarrow$ ’médicament’, ’maladie’, ’hémorragie’
- ’laroxyl’, ’dose’, ’faible’, ’stress’ $\longrightarrow$ ’laroxyl’, ’dosage’, ’faible’, ’anxiété’
- ’quel’, ’médicament’, ’contre’, ’cancer’, ’anatomie’, ’anatomie’ $\longrightarrow$ ’quel’, ’médicament’, ’contre’, ’sein’, ’anatomie’, ’anatomie’

Random walk in graph of words

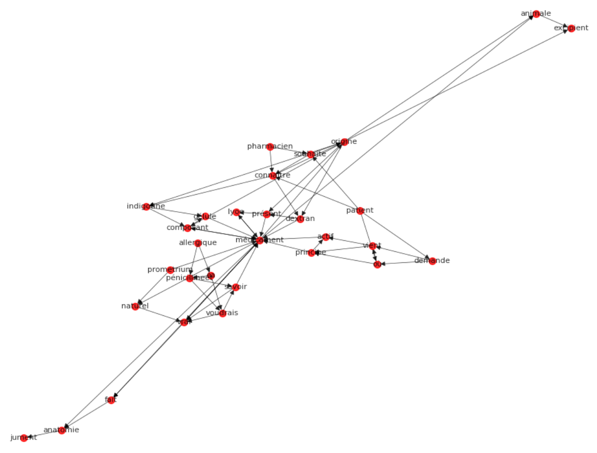
In [5] and [8], the authors use the graph-of-words (GoW) text representation. Textual document are represented as a graph whose vertices are unique terms in the document and whose edges capture term co-occurrence within a window of predetermined, fixed size, that is slided over the sentences. The authors applied this representation to the tasks of unsupervised keyword extraction (k-core) and extractive single document summarization. GoW enables to capture term dependencies, to encode the strength of the dependence as edge weights and to captures term order (via directed edges).

For each each of our 51 classes, we construct this weighted directed graph, with a sliding window $w = 3$. Our document is the concatenation of all the sentences in a given class.

Given a graph class representation, we generate new sentences with random walks on this graph. A random walk is initialized with a random node, at each step, we choose a new node among the directed neighbors of the current node. The probability to choose a node in this neighborhood is given by the normalized weight of the corresponding edge (representing term co-occurrence). If a node has no neighbor in the forward direction, the sentence stops. The created sentence has a maximum lenght $maxlen$.

We give en exemple with the most under-represented class : the class 40 with 7 sentences. We first give the input senteces pre-processed. - [’médicament’, ’origine’, ’composant’]
- [’pharmacien’, ’souhaite’, ’connaître’, ’origine’, ’indigotine’, ’composant’, ’gélule’, ’médicament’]
- [’allergique’, ’èa’, ’pénicillineet’, ’voudrais’, ’savoir’, ’vrai’, ’médicament’, ’fait’, ’anatomie’, ’anatomie’, ’jument’]
- [’patient’, ’demande’, ’où’, ’vient’, ’principe’, ’actif’, ’médicament’]
- [’patient’, ’souhaite’, ’connaître’, ’origine’, ’dextran’, ’présent’, ’médicament’, ’lyoc’, ’médicament’, ’lyoc’]
- [’médicament’, ’origine’, ’animale’, ’excipient’]
- [’prometrium’, ’médicament’, ’naturel’, ’vrai’]

The class seems to regard questions about med origins.



And here are examples of sequences generated by random walks :

- [’vient’, ’principe’, ’médicament’, ’origine’, ’excipient’]
- [’pénicillineet’, ’voudrais’, ’vrai’, ’médicament’, ’na-

turel', 'vrai', 'fait', 'anatomie', 'jument']
- ['naturel', 'vrai', 'fait', 'anatomie', 'jument']
- ['connaître', 'origine', 'animale', 'excipient']
- ['souhaite', 'connaître', 'origine', 'indigotine', 'gélule', 'médicament', 'origine', 'présent', 'lyoc', 'médicament', 'lyoc', 'médicament', 'composant', 'gélule', 'médicament', 'origine', 'excipient']

We observe that the sentences look relevant for the class has long as they are not to long. Moreover, we observe for example that the word "jument" appears in two of the created sentences when it appeared only once in the original sentences. It is a useless word for our class. To avoid this effect, an idea could be to compute the random walks only on the first K cores of the graph.

We create more data by over-sampling the under-represented classes. We want a minimum of nb_per_class elements per class. If a class has n_el elements and $n\_el \leq nb\_per\_class$, we create $nb\_per\_class - n\_el$ new sentences with random walks. All the created sentences are then modified with random word replacement.

4. Model

We study two different classification networks.

4.1. Convolunationnal Network

We first build the classical 1D-CNN sentence classification model, inspired from [4] and [10]. The model is illustrated in Figure 3. It is the model used for the benchmark of the challenge.

First all the sentences are padded and resized in order to be of same lenght s. We pad the sentences with a specific padding token. If the sentence is longer than s, it is resized by selecting randomly s words in the entire sentence.

It begins with a preprocessed tokenized sentence which we then convert to a sentence matrix, the rows of which are word vector representations of each token. These might be outputs from pretrained embeddings. we denote the dimension of the word vectors by d. The dimension can be tuned using PCA. If the length of a given sentence is l, then the dimension of the sentence matrix is $l \times d$. We then apply 1D-Convolutions with $n\_filters$ filters of 3 multiple sizes $filter\_sizes$. It allows to learn complementary features from the same regions. The dimension of the feature map generated by each filter will vary as a function of the sentence length and the filter region size. After a max-pooling layer, each feature map is then concatenated into a vector, itself connected to a last fully-connected layer.

4.2. Bi-LSTM Network with and without Attention

We also coded the standart LSTM in a bi-directional form. The standard LSTM cannot detect which is the important part of the sentence for aspect-level classification.

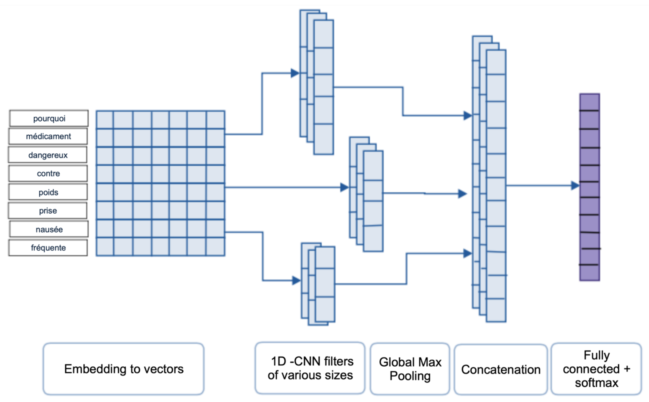


Figure 3. CNN model

In order to address this issue, [9] propose to deign an attention mechanism that can capture the key part of sentence in response to a given aspect. We tried both networks but they did not obtained better results than the previous CNN network. We did not have time to study them further.

5. Tests and Results

The networks are fitted with the Adam optimizer and batches of size 64. One technique for handling class imbalance, is to use a weighted loss function. In order to boost performance on the minority class, the penalty for misclassifying minority class can be increased. We use a weighted categorical cross-entropy loss :

$$L(y, x) = -\sum_{j \in C} w_j log(s_j)$$

where s_j is the model output score for each class (result from a softmax function).

We tried 2 different king of weights :

- method 1 : $w_j = \frac{N}{N_j * nb\_class}$ with N the total number of inputs and N_j the number of elements in class j.
- method 2 : $w_j = \frac{2*N_{max}}{N_j + s*N_{max}}$ where N_{max} is the number of elements of the bigger class and s is a smoothing factor.

We plot in 4 the evolution of the weights for the classes sorted from smallest to biggest. The corresponding evolution of class counts were displayed in 2.

5.1. Data augmentation

We work with the CNN model with standart values for parameters, we try data augmentation with the parameters :

- p : the probability of word replacement
- $maxlen$: the maximum lenght of sentences created by random walks

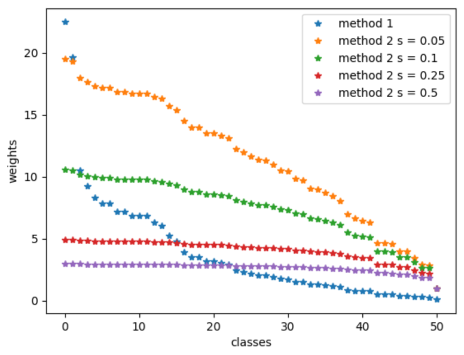


Figure 4. Weights for the weighted categorical cross-entropy.

- nb_per_class : the minimum number of sentences per class after data augmentation.

Parameter	Parameter space	Optimal value
p	[0.,0.1,0.2,0.3]	0
$maxlen$	[0,10,20,50,100]	20
nb_per_class	[0,50,100,200]	50

Table 2. Data augmentation parameters tuning

We give in tables 4 the results of data augmentation. Word replacement does not help. We explain this by the fact that, given the small number of elements is some classes, the network focuses on very specific words. For example, in the class that with introduced previously in part 3, the word "origin" seems very important. If we modify this words, we may add variance to the data. Moreover, the dictionary of synonyms should be learned on a specific medical corpus. The augmentation with random walks helps only when treating only under-represented classes with less than 50 elements. The reason is that our sentence have a minority of nodes useful for classification. Therefore with bigger classes, the graph gets more nodes, but it does not get much nodes relevant for classification.

5.2. Embedding layer

We study the importance of using a pretrained embedding for learning the word representation. We can choose to adjust the input pretrained embedding along training or to fix this representation and not to learn it. In table 3, we give the accuracy results with this different methods.

We observe that we obtain better result with no pretrained embedding than with the pretrained embedding described before. We explain this by the fact that the word representation learned is not optimal for the problem. We need a fine-grained separation between very specific words in order to distinguish classes. However, the pretrained embedding was learned on a FrWac general corpus and not on

	Accuracy
No pretrained embedding	0.657
pretrained embedding trainable	0.638
pretrained embedding not trainable	0.435

Table 3. Embedding layer training

a medical specific one. In 5 we display the 100 most frequent words of the dataset in the embedding spaces. We observe that the pretrained embedding assigns each word related to the medical domain closely in the space : e.g "risque", "douleur", "saignement", "traitement" ... ; It does not learn a strong difference between these terms. When the algorithm learns a representation by itself, it focuses on these important terms for classification. The solution should be to learn our own embedding on a big medical texts corpus. However, such a learning takes a long time to process.

This result disappointed us as the majority of our work was realized with this embedding and for this embedding. The data augmentation with word replacement by synonyms becomes useless with no pretrained embedding. Moreover, the OOV module was created with the embedding dictionary as reference.

5.3. Model parameters

We did not have time to tune all the parameters of the CNN. We use standart parameters given by the challenge provider :

- $d = 128$ the output dimension of the embedding layer.
- The filter sizes $filter\_sizes = [3, 4, 5]$.
- $hidden = 128$ The number of neurons in the hidden layer of the CNN.

We tune by cross-correlation the following parameters :

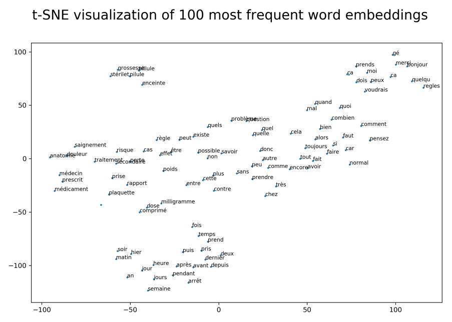
- l the lenght of the sentences in the input.
- s the weights of the loss function.

Parameter	Parameter space	Optimal value
l	$\{50, 100, 150, 200\}$	150
loss function	{normal, method 1, method 2}	method 2
s	$\{0.05, 0.1, 0.25\}$	0.25

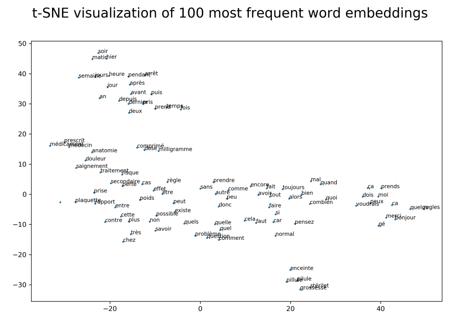
Table 4. MModel hyper-parameters tuning by cross-validation

5.4. OOV module

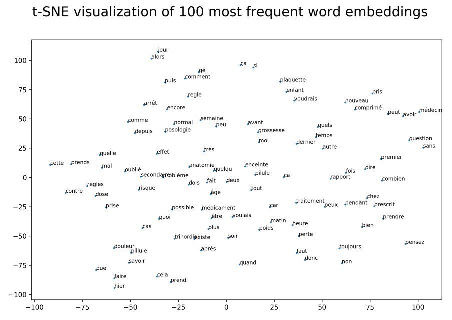
In 5, we show the improvement in performances when we had the correction of the OOV words. The performances are boosted even when not using the pretrained embedding. Given the fact that the pretrained embedding layer does not



(a) pre-trained embedding



(b) tuned pre-trained embedding



(c) Trained embedding without pretrained

Figure 5. T-SNE representation if the different embeddings considered

help, we could have performed an OOV correction with the "training corpus" dictionary as reference. We did not have time to do it.

	Accuracy
Without correction	0.649
With correction	0.656

Table 5. Performance with OOV correction

5.5. Bagging methods

To improve the classification performance of model, we propose to use the bagging ensemble method. We train N different model independently using randomly chosen training samples of 90% of the training data. Then, the model are aggregated into to make a collective decision with a voting scheme : we choose the class that obtained the maximum of votes from each of the N models.

We display in 6 the evolution of the accuracy with N. We choose to keep $N = 10$.

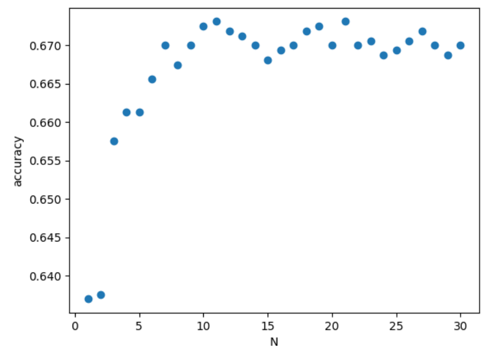


Figure 6. Evolution of the accuracy with N, with voting aggregation

6. Conclusion

In this work, we developed an algorithm for medical question classification. The main focus of our work was on data preprocessing. Due to the limited time we had for this project, we mainly tried data cleaning and data augmentations methods we already studied in class. We obtain an accuracy of 0.67 on the test set. We are disappointed of the classification performances we obtained with our methods. We did not succeed in handling the strong unbalance of the data set and overfitting correctly. However, we have appreciated trying to solve this difficult problem applying interesting methods studied in class. If we had more time, our next contribution would have been to train a new embedding on a specific medical corpus. We could also gain performance using methods based on the One-Against-Asll division idea to handle unbalance data sets.

References

[1] R. Al-Rfou, B. Perozzi, and S. Skiena. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[2] S. B. et Fišer Darja. Building a free french wordnet from multilingual resources. In *Ontole*, 2008.

[3] J.-P. Fauconnier. French word embeddings, 2015.

[4] Y. Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[5] R. Mihalcea and P. Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 2004.

[6] A. Névéol, C. Grouin, J. Leixa, S. Rosset, and P. Zweigenbaum. The QUAERO French medical corpus: A ressource for medical entity recognition and normalization. In *Proc of BioTextMining Work*, pages 24–30, 2014.

[7] I. Setiadi. Damerau-levenshtein algorithm and bayes theorem for spell checker optimization. 12 2013.

[8] A. Tixier, K. Skianis, and M. Vazirgiannis. Gowvis: A web application for graph-of-words-based text visualization and summarization. In *Proceedings of ACL-2016 System Demonstrations*, pages 151–156. Association for Computational Linguistics, 2016.

[9] Z. Yang, D. Yang, C. Dyer, X. He, A. J. Smola, and E. H. Hovy. Hierarchical attention networks for document classification. In *HLT-NAACL*, 2016.

[10] Y. Zhang and B. C. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR*, abs/1510.03820, 2015.