Advanced Natural Language Processing Project: Keyword extraction from scientific documents

Williams Rizzi
Master in Computer Science
University of Trento
williams.rizzi@studenti.unitn.it

Abstract

The scientific production within the medical domain grown with a very fast pace in the last years by making the classification and retrieval effort of effective documents a difficult task. Document annotation is a task from Natural Language Processing (NLP) enabling the summarization of document content by means of a closed vocabulary of concepts. By exploiting effective annotations, researchers would be helped about discriminating among the available literature which documents are more related with their interests. This work will address this task by combining word embedding techniques and pattern recognition capabilities based on the Conditional Random Fields (CRF) strategy in order to propose an effective domain specific annotation algorithm.

1 Introduction

This work wants to tackle the problem of annotating documents in a domain specific environment. Such a task has to cope with the challenges of contextualization, summarization and abstraction of document content. The *contextualization* perspective is a matter of being able to spot spans of text talking about a single concept. Whereas, with *summarization* is meant the need of correctly expressing spans of words into concepts. Finally, *abstraction* is intended as properly keeping in mind that not all spans are informative or worse some of them might be misleading. To address this challenges the machinery underneath the proposed solution employees a Conditional Random Field (CRF) algorithm to select spans of words to then tag with a word embedding model and, then decides either or not to let the entire document inherit the tag.

In the following sections the challenges from this problem will be analyzed, the dataset we used will be presented, the approach to the problem, the evidences of the experimental results, and finally, the conclusions and the possible further developments.

2 Challenges

The challenge presented in this work is: given a set of coarse grained tagged documents, we want to learn which are the features in the documents that brought such tags. The tags come from a medical knowledge base (i.e. the Medical Subject Heading (MeSH) vocabulary) and, the documents are all from a specific domain. It is therefore necessary a machinery able to identify those features, i.e. a fine grained annotation and, in order to exploit them for tagging each document with the correct concepts. A human in reading the document is able to discriminate those artifacts and not be fooled by the language. This ability comes from the fact that humans make use of background knowledge and abstraction capabilities giving more and less weight to the semantics of some parts of the text according to their role in the document. In this context the proposed approach will work on the following three tasks:

- to provide an effective feature annotation on the documents used for training our model;
- to detect which are the features contained in each document leading to a common meaning in the text;
- to associate detected features with one or more concepts contained within the MeSH vocabulary. This task enables the fine grained

annotation of the document with the related MeSH entities.

3 Dataset

As already mentioned in Section 1, the used dataset comes from the medical domain. The tuples composing the dataset are in the form: $< document, keywords >$. Where the *document* is the text contained in a scientific paper, that has been split line by line, and the *keywords* are the associated entities from the MeSH vocabulary, in a one-to-many relation.

Medical Subject Heading (MeSH),[1] is a comprehensive controlled vocabulary for the purpose of indexing journal articles and books in the life sciences. It serves as a thesaurus that facilitates the summarization, indexing, and retrieval of the documents from the PubMed digital library. Created and updated by the United States National Library of Medicine (NLM), it is used by the MEDLINE/PubMed article database and by ClinicalTrials.gov registry to classify which diseases are studied by trials registered in ClinicalTrials.gov.

The dataset is composed by $302,023$ documents with $272,162$ unique different keywords associated with an un-even distribution, for a total of $4,587,686$ associations, with entities like $'humans'$ that occurs almost for each document, i.e., $206,857$ times and, other keywords like $'lyssavirus'$ that occurs only once. The average amount of different keywords per document is 15.23 with a standard variance of 6.89, a maximum amount of 62 different tags and, a minimum of 1.

In the next section will be discussed the approach adopted to address the challenges of the dataset.

4 Approach

The proposed approach copes with the mentioned challenges by exploiting various techniques in order to reach optimal results, by employing in turn the CRF++ command line toolkit[2], crfsuite[3] and, the DEAP python evolutionary computation framework[4]. Is possible to divide the algorithm in the following blocks: (i) keyword selection, (ii) preprocessing, (iii) IOB-tagging, (iv) CRF train and, (v) document tagging. Since, as depicted in Section 3, the dataset size is quite important, is necessary to avoid importing the entire dataset in bulk. Instead, it has been selected a set of target keywords through a set of constraints expressed as parameters of the machinery and only those are loaded by the system. Data are then converted in the format required by the following steps. Data were split in three sets, the train set, the test set and, the evaluation set. The train set and the test set are then labeled with the IOB-tags (described below). These tags are associated with the word span containing a target concept through the word embedding nearest neighbours method. As a baseline the train set is then used to train a CRF model with a randomly built feature template and, then validated on the evaluation set. There were investigated also the possibility of automatically engineering the feature template through a genetic algorithm and the capabilities of the approach to well fit the approach peculiarities through a random search over the set of parameters of the CRF training procedure. Each of the aforementioned blocks is covered by the next Subsections.

4.1 Keyword Selection

The Keyword selection procedure allows the algorithm to cleverly handle the size of the dataset by loading and processing only what is strictly necessary in order to keep the footprint of the machinery on the memory limited. This is achieved by letting the user: (i) to specify the desired minimum and maximum amount of documents per keyword, (ii) whether or not to use sets of document disjoint from the keywords perspective, and (iii) the type of keywords. Concerning the last point, the MeSH vocabulary is in fact composed by three types of keywords. The single-word keyword, i.e., $'Coenzymes'$, the multi-word keyword, i.e., $'Heart\ Failure'$ and, the keywords in slash notation, i.e., $'Heart\ Failure/pathology'$.

4.2 Pre-processing

As already mentioned in Section 3, the dataset is composed by a set of documents associated with their keyword. Those documents come in a natural language format stripped from the relative documents, with no distinction between sentences. Therefore, the first step of the machinery is to split the documents in sentences through the sentence

[1] https://en.wikipedia.org/wiki/Medical_Subject_Headings

[2] https://github.com/taku910/crfpp

[3] http://www.chokkan.org/software/crfsuite/

[4] https://github.com/DEAP/deap

tokenizer of the Stanford NLP Framework (Manning et al., 2014). The sentences are then split to achieve a per-word granularity for each sentence.

4.3 IOB-tagging

Each sentence is then tagged with the IOB-tag notation. IOB-tags follow the CoNLL IOB notation (Tjong Kim Sang and De Meulder, 2003) in which each concept is classified as the part of the multi-word span it belongs to, as begin (B), in (I) or out (O) of the concept span. For readability purposes the words *concept* and IOB_tag will be used from now on interchangeably. In particular, the sentence is scanned in search of a token which point in the word embedding space is close, within a given distance, to one of the concepts selected by the keyword selection process 4.1. When a token is within this given range the token is labeled with an I-tag and, the surrounding tokens within the concept span window parameter, inherit the label too; with a peculiarity that that the first token in the sequence is marked with a B-tag; defining in this way a concept span as shown in Snippet 1. In particular, in the snippet it is used a window parameter of 1, the matching values are $'X - rays'$ and $'bone'$ for the $'Orthopedic'$ MeSH entity and, $'Imaging'$ for the $'Imaging'$ MeSH entity.

The current support of the approach is given for the single-word keyword, as aforementioned in Section 4.1, as the other two need to provide a custom solution for the pre-training of the word embedding model. In particular the word embedding model is pre-trained on a domain specific dataset, as in literature there is evidence that outperforms the multi-domain models (Dragoni and Petrucci, 2017). The pre-trained model is generated through the Intel optimized word2vec pipeline (Ji et al., 2016).

Snippet 1 Here it is represented an example of the span annotation over a sentence from the medical domain. The first column represents the sentence itself, whereas the second column is the IOB-tag output from the designed machinery.

```
X-rays          B-Orthopedic
and             I-Orthopedic
bone            I-Orthopedic
scan            I-Orthopedic
are             O
the             O
standard        B-Imaging
imaging         I-Imaging
techniques      I-Imaging
.               O
```

4.4 CRF training

The train and the test set are now in the form of a matrix with on the left most column the tokens from the sentences split per line and on the right most column the IOB-tags. When training, the CRF algorithm takes care of constructing the features using a customizable template. This template is then used as a regular expression to construct the feature set that is then used to train the weights of the CRF model. The feature construction algorithm is then evaluated for each token of the sentence; meaning that for an average document with 140 sentences and about 10 to 14 tokens per sentence, depending on the feature template used, the size of the feature set can easily explode. This requires the expert a particular attention when designing the feature template. Moreover, in making use of CRF algorithms is necessary to carefully tune parameters or, adopt strategies, to avoid the overfitting of the algorithm. This two challenges are addressed by exploiting a genetic algorithm to automatically develop an optimal feature template, by carefully tuning the parameters of the CRF training procedure through a random search optimization algorithm. And, by dividing the dataset in three subsets, i.e., train, test and evaluate set, using the evaluation set solely to evaluate the models.

4.4.1 Genetic feature discovery

A common problem in employing CRF algorithms is that the feature engineering process is quite complicated and needs to be repeated for each different configuration of keywords. To make the approach fully dynamic have been used a genetic optimization technique. Therefore, the feature template selection has been treated as a black-box optimization problem. The CRF single template is treated as an individual, the single feature function as a gene and, a set of individuals is intended as a population. Genetic algorithms work relying on the definition of: (i) the selection function, (ii) the reproduction function, (iii) the gene generator function and, (iv) the mutation function. Those functions allow the genetic algorithm to simulate the evolution through out the lifetime of a generation of individuals and the natural selection of the best individuals making the following generations better and better.

Initialization and Selection. The population is initially composed by some randomly generated individuals. After the first epoch, the population performances are tested using as reference score the F-Measure and the best individuals are selected. Following, an individual can either mutate, reproduce itself or, both.

Reproduction. The reproduction generates a new individual randomly choosing genes between the parents' genes. Parents are randomly selected among the population.

Gene Generator. The gene generator, at least for the random case, is quite straightforward. First of all it is necessary to define the amount of tokens are meant to be inserted in this gene, then is necessary to define a separator for the tokens of the gene and, finally tokens are initialized as particular words through random offsets over the input.[5]

Mutation. If an individual mutates it will either, delete, insert or, substitute some of his genes. Where the latter two mutations will take place by randomly generating new genes, with the aforementioned generation function.

The resulting new population is made by new individuals generated through the recombination operator and the best ones of the previous generation selected by the selection operator. At each epoch the genetic algorithm will keep a rank of the best individuals in the population. This way, the best individual is always preserved.

4.5 CRF hyperparameter optimization

The selected feature model is then used to train a CRF model exploiting a random search optimization algorithm. The random search optimization algorithms rely on the definition of: (i) the fixed parameters, (ii) the parameter space and, (iii) the scorer function. Fixed parameters were: the training algorithm (i.e. the L2 algorithm in our case) and a flag enabling which transitions are allowed within the CRF network. In our case we set it to true. The parameter space has been defined over the variations of the c parameter that quite often lead the model to overfit and therefore must be set with particular sensibility.

4.6 Document tagging

At this point the trained CRF model is able, given a textual input, to find out the tokens in which the text is mentioning a specific concept. The evaluation set is then annotated with the CRF model. The resulting annotated document is parsed and, the document inherits the labels of the concept spans found in the text.

In the next section will be presented the experimental setup and results from the proposed technique.

5 Evaluation

The experimental setup of the proposed algorithm was the following: (i) the keyword selection specification, (ii) the data splitting percentages, (iii) the word embedding pretrained model, (iv) the word embedding labeling strategy, (v) the genetic algorithm hyperparameters and, (vi) the random search optimization hyperameters. For what regards the keyword selection specification the minimum amount of documents per keyword have been set to 1000, and the maximum to 10000 in order to have a various, informative and, well balanced amount of training examples. The type of MeSH ontology keywords accepted have been set to the single-keyword and the intersection of the document keywords have been left to be either populated or not. For what regards the data splitting percentages have been set empirically to 60% training, 20% testing and, 20% evaluation. The pretrained model used is created with the Intel optimized pipeline as already mentioned in Section 4.3 and; has been trained in 15 epochs with a layer size of 512 nodes. For what regards the labeling of the train and test set the threshold used to identify whether a given token projection is close enough to be treated as a word related to a concept is set to 4.5 and the window size of the resulting concept span is set to 2. Which results in spans containing the two previous words, the current and, the following two words.

CRF baseline Two different approaches have been used as baselines; an instance of the StanfordNER (Finkel et al., 2005) CRFClassifier and a simple CRF model trained without any optimizations. As paramters of the CRF model have been

[5] The idea of mutating the genes by randomly rewriting the feature template and, therefore, making the evolutionary algorithm easy to implement is courtesy of Davide Pedranz, https://github.com/davidepedranz/lus_project_2

selected the L2 as training algorithm, with a c parameter of 1.5 and a maximum amount of 200 iterations.

Randomized search over CRF hyperparameters For what regards the random search has been set a 3-fold cross validation, the F1 as scorer function and, the range for the searching space for the c parameter to be drawn from a uniform distribution between 0.8 and 2.5 for every configuration.

Genetic feature engineering Finally, for what regards the genetic algorithm the amount of generations or epochs used is 5, the single tournament is composed by 4 individuals, 2 is the amount of selected individuals for the next population from the tournament and, 1 individual is selected from the hall of fame as champion model to output.

Table 1: The table shows the comparison of the results with a dataset of $96,578$ features from the StanfordNER CRFClassifier (NER), the baseline CRF model (BS), the random search optimized CRF model (RS) and, the genetic feature engineered CRF model (GEN); the Tag column represents the amount of target tags in the feature set.

Tag	F1-NER	F1-BS	F1-RS	F1-GEN
3	0.00	0.57	0.61	0.85
4	0.00	0.57	0.69	0.86
5	0.27	0.57	0.56	0.90
6	0.14	0.57	0.55	0.90
7	0.09	0.57	0.74	0.86
8	0.09	0.57	0.65	0.85
9	0.36	0.57	0.69	0.85
10	0.09	0.57	0.71	0.86

5.1 Discussion

The StanfordNER show very poor performances on low amount of tags; in particular on 3 and 4 tags where it has not been able to give any correct label to the evaluation set. There is high variability in the results and in many cases its performances are close to a randomic classifier. This can be due to the fact that the features used to train the CRF inside the StanfordNER are not able to adapt properly to varying environment. Therefore, justifying the spike in the performances in the case of the 9 tag classification.

We can notice how the CRF baseline algorithm, when the c parameter as been fixed, were not able to discriminate among the correct concepts and this lead to the prediction of the most frequent concept every time. This is the reason for which the F1 value is always the same.

By observing the results contained in the third and fourth column we can observe the importance of varying the value of the c parameter. The results show important difference w.r.t. the baseline algorithm and, better learning abilities also with more tags.

Then the effectiveness of the genetic optimization strategy is definitely demonstrated by observing the results reported in the fourth column. We noticed that when the number of tags increased, some of them resulted more difficult to be learned than others. However, overall the genetic feature template discovery strategy seems to effectively answer this challenge.

The results from the baseline with a simple feature template show interesting results and, has been used as an initial experimentation for a feasibility test. The following two tests, i.e., the randomized search and the genetic approach, show the importance of a proper design of the feature template and, the importance of an optimized set of hyperparameters. The feature template remains one of the most important factors for a well performing CRF model and the hyperparameters show to have substantial importance in avoiding overfitting and granting stable performances. Moreover, the randomized search with its embedded cross validation of the result is a strong enforcer to avoid extreme cases of overfitting in which is easy to incur due to the CRF training algorithm nature.

6 Conclusions

The results obtained in our experiments are overall very interesting and, the approach seems to well answer the presented challenges. Nonetheless, might be very interesting enhancing the support to the types of entities of the MeSH vocabulary of the pre-trained word embedding model, mentioned in Section 4.3 and, better exploiting the Stanford NLP Toolkit by not only using its sentence splitter function but also the Part-Of-Speech (Martinez, 2012) and Lemma (Gross, 1998) tagger since the CRF exploited library supports a user defined amount of input features. Moreover, the technique used to abstract from the fine grained IOB-tags to the document level keywords is naïve, a possible further development might involve enhancing this feature abstraction step exploiting a pattern based filter machinery.

References

Mauro Dragoni and Giulio Petrucci. 2017. A neural word embeddings approach for multi-domain sentiment analysis. *IEEE Trans. Affective Computing* 8(4):457–470. https://doi.org/10.1109/TAFFC.2017.2717879.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*. pages 363–370. http://aclweb.org/anthology/P/P05/P05-1045.pdf.

Maurice Gross. 1998. Lemmatization of compound tenses in english. *Lingvisticae Investigationes* 22(1):71–122.

Shihao Ji, Nadathur Satish, Sheng Li, and Pradeep Dubey. 2016. Parallelizing word2vec in shared and distributed memory. *CoRR* abs/1604.04661. http://arxiv.org/abs/1604.04661.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. pages 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

Angel R Martinez. 2012. Part-of-speech tagging. *Wiley Interdisciplinary Reviews: Computational Statistics* 4(1):107–113.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics, pages 142–147.