Omar López & Antonio Pariente

MUD LABORATORIES

SESSION 2

Introduction

Through this part of the project, we aimed to enhance the sequence tagging process. These improvements relayed in two essential parts: feature extraction and classifier. Therefore, we added features to the pool of the feature extractor and tried some models to achieve better classification rates.

Feature Extraction

Before adding any features, the macro average percentage was of 55% as it was indicated in the project statement.

The first kind of features that we decided to add to the pool were some basic ones: presence or absence of upper cases, numbers, abbreviations, etc. Despite of resulting in an improvement of the macro average of success (both in CRF and NB), this advance was too mild since it oscillated in between 0.5% and 5% of success increasement, leading to a 60% of success.

The second kind of features resulted in more significant changes. These were inferred from the Annotation Guidelines for DDI Corpus, through this document, the usual nomenclatures of drugs are explained. The abbreviations, numbers and words from usual chemistry formulas are exposed together with some examples, for most of the drugs included in this project. By using this information, we added some features, for example: numbers followed by commas and some dashes are used in drugs (organic chemistry) like *1-methyl-4-phenyl-1,2,3,4-tetrahydropyridine*.

The greatest improvement was achieved by including vocabularies of words provided in the resources folder. This includes the *DrugBank.txt* and the *hsdb.txt* files, with already known drug names. We also added a custom drug list of 50 names. With these changes the macro average percentage scaled up to 73% with the CRF.

By looking at the confusion matrix, it became clear that the *drug_n* entities were the ones resisting the classifier, this was due to their low presence in the original dataset. Once again, we used the *Annotation Guidelines for DDI Corpus* to better understand these kinds of drugs. The annotation defined *drug_n* as "*active substances altering drugs such as drugs, food, drinks or environmental chemicals*". We also included some *drug_n* vocabularies, but it didn´t impact classifier success. The last remaining option is to include all these entities present in the original dataset as a feature vocabulary, however we considered this out of the scope of this project.

Models & Results

The first classifier was the CRF (Conditional Random Field), was the one providing better results. And the one used for feature testing of the previous section. Some initial proves lead us to use this first classifier as the reference one.

Other used classifiers were the NB (Naïve Bayes), which obtained a final macro average of success of 58% over the testing set. Other considered model was the SGD (Stochastic Gradient Descent) providing a 58.7% of accuracy. The SGD classifier was considered as an alternative since, by doing some research, it revealed itself useful in tagging procedures. Another reason behind, was the simplicity and smoothness of the classifier, being widely used in many optimization techniques.

As a conclusion we can confirm that, after all performed trials, the CRF classifier outperforms the two previous models: NB and SGD. As it has already been said, considering more complex models could raise the accuracy to significant levels. However, we have detected a failure point: *drug_n* drugs. If fixed, the accuracy levels would raise to 80%, which is enough for many real-world applications in this field. Therefore, we can say that CRF is able to capture the underlying complexity of drug tagging. In some tasks, other accuracy percentages may be required, but we should remember that we´re dealing with human language tasks, which are by nature irregular and difficult to solve.

SESSION 4

Introduction

This second part of the project took care of the Relation Extraction process. The aim in this section is to bind words through labels called relations. Just as the section before, it is performed in two main steps: feature extraction and model training. Here, we will focus on adding possible labels to the words, called features, to improve the accuracy of the later trained model.

Feature Extraction

The base option for the model is a Naive Bayes classifier. By using this structure, the initial macro average (F1) is of 33%, which leaves much space for improvement. The main and relevant features added to the pool were labels for tokens between the desired words *E1* and *E2*.

As a first measure, three properties were added: the tag, the lemma and relationship type of each word. All this information is extracted from CoreNLP, including the DependencyGraph. All of these labels raised the success percentage to 36%, but it wasn't considered a satisfactory rate. Also entity types from the tree were inserted, resulting in good guessing impact.

Other features were also considered, for example, the siblings label between tokens close to entities E1 and E2, for each respective pair. Labeling lemmas as negations such as *not*, *no* or *n't* or the presence of modal verbs like *could*, *may*, *might*, *etc,* were some of the modifications considered. However, these mentioned changes did not result in relevant changes over the F1 measure and therefore, were discarded from the feature pool.

The most relevant change, which raised the macro average of success to 40%, was to include adverse, beneficial and neutral verbs. By doing this, all verbs that were filed in each of the pools, were labeled with three ranks from good to bad in the scale of the human language intention. It may be because of the considerable size of the number of total verbs considered (nearly 60), that this change became the most relevant in the success improvement.

Models & Results

As it was already mentioned before, the starting F1 was of 33% for Naïve Bayes classifier. After all the feature enhancement, this success percentage was raised to 40%. Apart from this model, three more classifiers were considered: the SVM (Support Vector Machine), SGD (Stochastic Gradient Descent) and RF (Random Forest).

The worst results were achieved with the SDG model, which after applying all the feature changes, provided a F1 percentage of 33%, similar to the one given by the original NB without the extra labels. The Random Forest obtained a 38% of success, which puts it in the second place when ranking the new models. Finally, the SVC outperformed the previous two models by achieving the best result, 46% of macro average. Despite of the significant improvement with respect the original model, this percentage is does not meet our goals, and therefore we suggest training other models. The NN are the natural step when adding complexity to these kinds of classifiers but were forbidden for this project.

When looking more in detail at the result tables provided by the models, there are two kind of relations presenting problems. The first one, *advise*, gave low prediction percentages for the SGD, and the original model, NB. The other problematic relation was *mechanism*, giving values of success around 10% for RF and SVC.

Despite some kind of relations presenting problems for our models, it is not problem of one kind of input, like happened in the Session 2 with *drug_n* drugs. In this case, we cannot see the presence of a single failure point. Instead, we perceive a poor overall performance, since no kind of input accuracy raises above 40% for any model, with exception of the *int*, presenting nearly 70% of F1 for some models. We guess, that is outstanding performance with integers, happens because the kind of relations that can be inferred from numbers, are more specific than for general words like verbs or nouns.

As it has already been said, we consider that it is not a matter of adding more features to the label pool. In this case, it is necessary to use other kind of models since the ones here used are not able to capture the underlying complexity of relations.

Code & Results

Every resource used during the project, together with the written code and generated results, is available in the corresponding GitHub repository. We highly recommend checking everything in the repository, since it is better structured and documented.