AHLT: Drug-Drug Interaction and Drug Named Entity Recognition

Carlo Alessi, Doria Saric

July 5, 2018

1 Introduction

Drug-drug interaction (DDI) is obtained when the effect of a drug changes as a result of the interaction with another drug. The study of DDI is an important research area in the medical domain, which aims to ensure the safety of patients that need to take a variety of drugs at the same time. The automation of DDI, using Natural Language Processing techniques, is rapidly gaining momentum in order to aid physicians and reduce healthcare costs.

This paper reports the methods and experiments carried out to solve the DDI Extraction challenge of SemEval-2013, which aims to discover patterns in two medical datasets (MedLine and DrugBank), and is divided in two subtasks:

- Task 9.1: Recognition and classification of pharmacological substances.
- Task 9.2: Extraction of drug-drug interactions.

The tasks will be performed using Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs). Their performance will be evaluated and compared, while operating with different sets of features, with the aim of preventing overfitting and obtaining the best possible results on the test set. The outline of this report is as follows. In section 2 we will describe the methods and approaches taken to address the challenge. The experiments and results of Task9.1 and Task9.2 are reported respectively in section 3 and section 4. Finally section 5 concludes with a discussion on the main points and gives some insights for future work.

2 Methods

Python was used as a base language for the high variety of libraries it offers. First we parsed the XML files using ElementTree library and stored the data in several pandas dataframes in order to decrease redundancy. In one dataframe we stored the sentences along with their corresponding IDs. In another dataframe we have put only entities, their IDs, their names, and their position in a sentence. Finally, for Task 1 we created the dataframe in which we stored the pairs of drugs, ID of the sentence this is mentioned in, and their interaction type.

Scikit-learn was used for the feature transformations and the SVM classifier, whereas Keras was used for the definition of the ANN. Custom callbacks were defined to monitor different metrics during the fitting process. Numpy was used to perform basic mathematic operations and to create a dataset in the format required by the classifiers. The word embeddings were created with Word2Vec from the Gensim library. The continuous bag-of-words model, which predicts a word based on its context, was used to create embeddings from the DrugBank and MedLine datasets. In order to avoid overfitting and select the best hyperparameters and features, we retained 10% of the training data as validation set.

In the preprocessing pipeline we used NLTK library to tokenize sentences. Additionally, we experimented with NLTK's Part-of-Speech tagger, appending POS tags on words and obtaining the embeddings that way. Furthermore, we experimented with the stemmer and lemmatizer.

3 Task 9.1

3.1 Experiments using ANN

In this section are first reported the experiments regarding different word-vector setting. Afterwards, we analyze the learning process of our network. The experiments settings are described in Table 1.

Table 1: ANN architecture and training setting.

Architecture	$[vector\_size, 512, 256, 3]$
Activations	ReLU, Softmax
Dropout	0.5
Objective	Cross-entropy
Solver	Adam
Epochs	30

Word-vector size The first experiment aimed to find a word-vector size that was big enough, in order to be able to capture the variability of the data, and of contained size, to do not fall in the curse of dimensionality problem. As reported in Table 2 the performance sweet spot was found with vectors of size 20. It is also clear to see that the performance dropped significantly with vectors of size 200.

Table 2: F1-score for different word-vector sizes.

Vector size	Micro F1	Macro F1
10	0.9481	0.5041
20	**0.9517**	**0.5435**
30	0.9517	0.5364
50	0.9495	0.5215
100	0.9499	0.5232
200	0.9479	0.4855

Word-vector type The second experiment investigated the individual and joint contribution of stemming, lemmatizing and adding the POS in creating the word-vectors. Table 3 summarizes the results obtained, and shows that best results were achieved creating the word-vectors from stemmed words.

Table 3: F1-score for different word-vector types.

Type	Micro f1	Macro f1
original	0.9499	0.5098
stem	**0.9599**	**0.6144**
lemma	0.9526	0.5499
original + PoS	0.9481	0.5105
stem + PoS	0.9517	0.5435

Word-vector preprocessing This experiment compared different preprocessing techniques applied to the word-vector. The results summarized in Table 4 tell that it was better to operate with the non-transformed word-vectors.

Table 4: F1-score for different word-vector preprocessing.

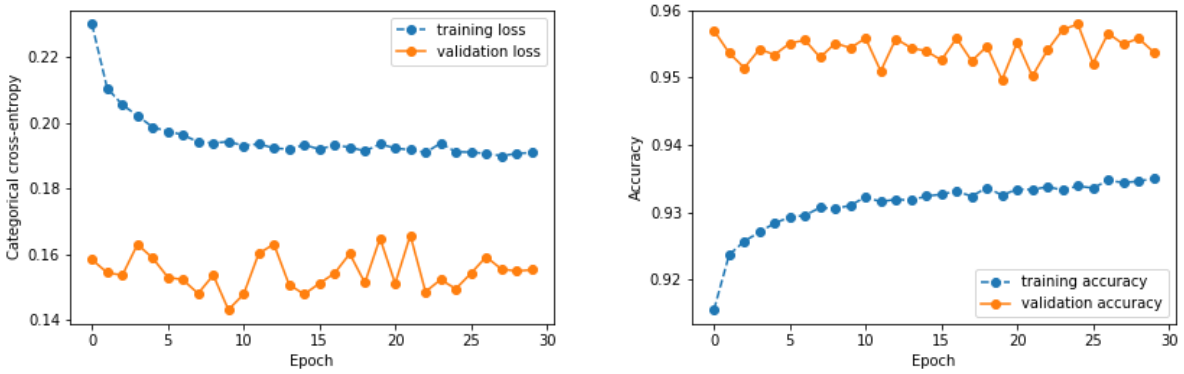
Preprocessing	Micro f1	Macro f1
None	**0.9599**	**0.6144**
MinMax(0,1)	0.9515	0.5207
Standardize	0.9581	0.5897

Learning process The learning process of the network was monitored by looking at how different metrics (loss, accuracy, precision, recall, f1-score) changed as the training proceeded.

With the objective of monitoring the problem of overfitting, loss and accuracy where monitored in both training and validation set. The learning curves are shown in Figure 1. Figure 2 shows the curves for the macro- and weighted-averaged precision, recall and f1-score.

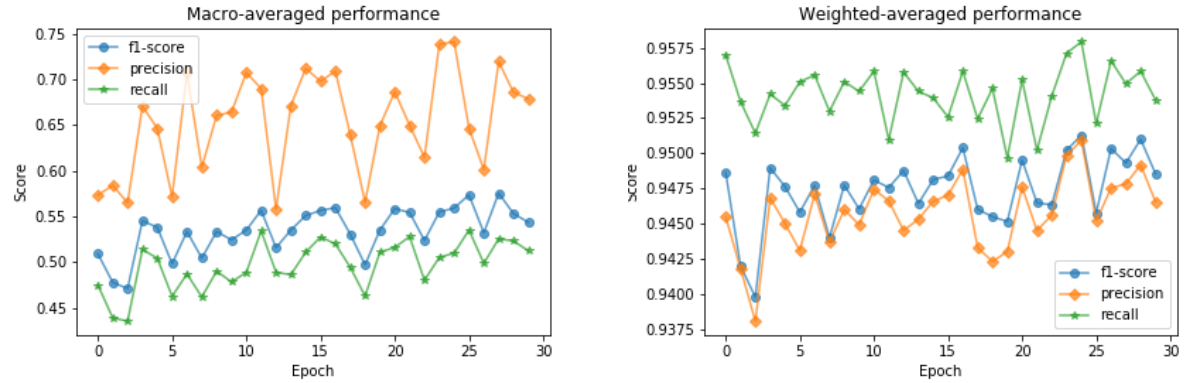
3.2 Experiments using SVM

This section reports the results of the experiments performed with the SVM. After briefly tuning the hyperparameters via grid-search, it was decided to use radial basis function kernels, $C = 1$, $\gamma = auto$ and tolerance $\alpha = 0.001$. It was experimented with different combinations of hand-crafted features, sometimes combined with the embeddings. Table 5 shows



(a) loss (b) accuracy

Figure 1: Learning curves



(a) macro (b) weighted

Figure 2: Macro and Weighted scores.

the performance obtained training using embeddings and hand-crafted features.

Chosen feature list Table 6 reports the hand-crafted features used by the SVM, which are justified as follows. Named entities (drugs) are usually capitalized or uppercased, and are nouns. Moreover, most of the technical terms in medicine have Latin or Greek roots ($ph, th, etc.$), and can have more consonants than vowels. Furthermore, drugs are generally long words, and may contain hyphens or even numbers. A final drug feature is that they are usually surrounded by trigger words. In Table 7 are reported the results obtained with the SVM on the validation set with different set of features.

Table 5: The comparison of SVM performance when trained with features and only on embeddings.

Metrics	embedding + features	embeddings
Acc	0.2505	0.9487
Precision	0.0627	0.9396
Recall	0.2505	0.9487
F1	0.1004	0.9355

Table 6: Features used by SVM in Task 9.1.

Feature	Description
has_POS_NN	word is a noun
has_numbers	word has numbers or hyphens
is_capitalized	word is capitalized
has_more_consonants	word has more consonants
is_a_long_word	word has more than 7 letters
has_trigger	sentence contains

Table 7: SVM validation results.

	Metrics	STEM	STEM+POS
Micro	Precision	0.9487	0.9368
	Recall	0.9487	0.9368
	F1	0.9487	0.9368
Macro	Precision	0.6994	**0.7229**
	Recall	**0.4644**	0.398
	F1	**0.5195**	0.4348
Weighted	Precision	**0.9396**	0.9269
	Recall	**0.9487**	0.9368
	F1	**0.9355**	0.914
Accuracy		**0.9487**	0.9368

3.3 Task 9.1 results

In this section are reported the results of Task 9.1 of the challenge. Table 8 summarizes the performance obtained in the DrugBank and MedLine datasets, using the best models selected in subsection 3.1 and subsection 3.2. It can be seen that, on one hand the SVM achieved higher precision than the ANN, and on the other hand, the ANN obtained higher recall. As a result of the harmonic average between the two scores, the ANN obtained a higher F1-score.

Table 8: Results Task1 on gold test dataset. In bold are shown the best results, whereas the bad results are underlined.

		Exact			Partial		
Model	Dataset	Precision	Recall	F1	Precision	Recall	F1
ANN	DrugBank	**0.61**	0.43	0.5	**0.61**	0.5	0.55
	MedLine	0.51	<u>0.29</u>	0.37	0.51	<u>0.35</u>	0.41
	Both	0.56	0.35	0.43	0.56	0.41	0.48
SVM	DrugBank	**0.78**	0.34	0.47	**0.78**	0.37	0.51
	MedLine	0.55	<u>0.12</u>	0.2	0.55	<u>0.14</u>	0.23
	Both	0.69	0.22	0.33	0.69	0.25	0.36

4 Task 9.2

For the second task we used a hierarchical classification approach composed of two phases. The objective of the first step was only to determine whether there was an interaction between a pair of drugs (regardless of the type). The second part aimed to distinguish the type of drug-drug interaction (mechanism, advise, effect, int.). The two phases were performed respectively by a binary and a multi-class classifier, which operated in two different set of features. Table 9 summarizes the features used in the first phase. The second step consisted in classifying only the examples that went through the first step of the pipeline (i.e. those with prediction equal 1). The features used for discriminating the type of interaction were created from a list of trigger words divided into 6 groups. It was thus created a binary vector of length 6, where the i^{th} entry was either 1 or 0 depending on whether a trigger word belonging to the i^{th} group was present in the sentence. Both phases of the pipeline use an SVM.

4.1 Experiments using SVM

Table 10 reports the results achieved by the SVM on the validation set, at both stages of the pipeline.

5 Conclusion

In this report we used ANNs and SVMs with different features to solve task 9.1 and task 9.2 of the SemEval-2013 DDI Extraction challenge. From the

Table 9: Features set used for the first phase of Task 9.2.

Feature	Description
are_the_triggers	if word is surrounded by trigger words within a centred window of size 4
is_there_negation	1 if there are negation words or verbs in a negated form, 0 otherwise
token_distance	number of tokens between two entities
is_there_punctuation	1 if there is punctuation between the two drug entities, 0 otherwise
are_there_conjunctions	1 if there are some conjunction words in the whole sentence, 0 otherwise

Table 10: The performance in both phases for Task 9.2.

Metrics	Phase 1	Phase 2
Acc	0.8546	0.4
Precision	0.7435	0.16
Recall	0.8546	0.4
F1	0.7908	0.2286

experiments performed in section 3 and section 4 the following observations were drawn:

a. Word-vectors with small dimensionality were not able to capture all the semantic patterns in the data. However, word-vectors of higher dimensionality (such as 200) caused a performance drop due to the limited availability of data (see Table 2).

b. Creating word-vectors from the stem, from the lemma, or from the word-PoS concatenation improved the performance with respect to creating the word-vectors from the original words. However, combining different approaches (e.g. creating vectors from stem-Pos) resulted in worse results than when the stem and Pos were used separately (see Table 3).

d. It was more challenging to obtain good results on the MedLine dataset. The reason could be related to the much smaller dataset size compared to the size of DrugBank.

e. The results obtained in the two stages of the pipeline, for task 9.2, suggested that the first binary classifier was able to predict whether two drugs interacted, achieving 85% accuracy. Unfortunately, due to poor feature engineering in the second stage, the performance drastically dropped to 40%.

Future work would focus more on creating and selecting more representative features. Better features would allow the exploration of more complex models, such as Long-Short Term Memory, which are able to capture long term dependencies that frequently occur in natural languages. It would also be interesting to combine publicly available embeddings, such as the Google-News embeddings, with embeddings created from the two datasets used in this report.

References

[1] B. Bokharaeian, A. Diaz, M. Neves, V. Francisco. Exploring Negation Annotations in the DrugDDI Corpus

[2] J. Bjorne, S. Kaewphan and T. Salakoski. UTurku: Drug Named Entity Recognition and Drug-Drug Interaction Extraction Using SVM Classification and Domain Knowledge