Improving Topic Tracking in ADVISER through the implementation of a Multi-Class Perceptron

Kiflom Desta Abrha[1], *Pavlos Musenidis*[2]

[1]University of Stuttgart, Germany
[2]University of Stuttgart, Germany
st154345@stud.uni-stuttgart.de, st143387@stud.uni-stuttgart.de

Abstract

With our work we want to improve topic tracking in ADVISER, a multi-domain dialog system toolkit. We want to give a reliable and better alternative to the current keyword-matching algorithm by implementing a multi-class perceptron.

For our experiments we use a version of the MultiWOZ data set. We developed eight different baselines, all based on the keyword-matching system of ADVISER which differ in the amount of keywords they search, the use of WordNet to also consider synonyms of the keywords and the possibility of assigning one or multiple domains. Also we implemented a multi-class Perceptron using two different feature sets, one general which could also be used for different domains and one which is more fitted to the training data.

Our results show that the best keyword-matching configuration doesn't use WordNet for synonyms but uses multiple keywords and assigns multiple domains. The multi-class perceptron however beats this baseline with both feature sets by more than 20%.

Even though we had problems with the weight adjustment in the learning phase of the multi-class perceptron and did not find the best solution for that, we reached our goal of developing a reliable high-performing and generalizable classifier.

1. Introduction

As the world gets more digitalized and people want to receive information in a comfortable way and without having to search for information themselves, dialog systems become a more and more useful everyday tool. This can be seen in the success of Apple's Siri or Amazon's Alexa. In this paper we will try to improve a component of ADVISER, an open-source multi-domain dialog system toolkit which allows users to build conversational agents and do research on such dialog systems [1].

Topic tracking is just one of the many different tasks a dialog system must fulfill. To be able to track the dialog state and fill the right slots the system must know, about which topic the user wants information.

The current version of ADVISER uses keyword-matching to track the topic of the dialog. As this is a pretty basic approach to topic tracking, we aim to build a more complex topic tracking algorithm, which classifies topics using a variety of features which it learns from a data set. This way we hope to contribute to the ADVISER toolkit in that we provide users with a more reliable topic tracking mechanism, which can be trained with the data set we used for the domains 'Hotel', 'Attraction', 'Taxi', 'Train' and 'Restaurant' or with different data sets and for different domains.

The Baselines we chose are all based on the keyword-matching algorithm used in the ADVISER. However we created different configurations to see how good the keyword-matching system can get and how much impact assigning multiple domains, using multiple keywords and considering synonyms of keywords can be.

We look at topic tracking as a classification problem which is why we decided to use a multi-class perceptron to solve the task. We use two different configurations, one which uses keywords we picked from the data and is therefore less generalizable and one which doesn't use them and can be trained with different data for different domains.

2. Related work

There are some papers focusing on topic tracking, however we found that no paper uses the dataset which we are going to use for our experiments. For that reason we decided that we cannot compare our results with other papers.

[2] first released a paper exploring recurrent neural networks for topic tracking and showed the significant improvements they led to. The most recent and most comparable paper we found was [3]. In this paper he tries to solve the task by using different dynamic memory networks and compares different models for that. He showed that dynamic memory networks perform better than recurrent and normal convolutional networks and achieved an F-score of 0.705 in topic tracking. The data set that was used in both papers was the TourSG corpus [4] and included tourist and guide turns. For our classifier however, we will use a version of the MultiWOZ data set that only considers the tourist turns.

3. Methods

To perform the experiment we had to develop the baselines and implement the multi-class perceptron. We developed eight different baselines which alternate between using one keyword and using multiple keywords, between using WordNet to also look for synonyms of the keyword and not using it and between assigning just one domain and assigning multiple domains.

3.1. Developing the Baselines

For our first baseline we just used the domain tracking algorithm which is given in the ADVISER. This algorithm uses ontology files in which there are predefined keywords. However the domain tracker could only look for one keyword per domain and the predefined keywords for a domain always was the domain name. It also could assign only one domain to a sentence. To construct the seven remaining baselines, we changed the algorithm so it can:

- search for multiple keywords in a given utterance.

Table 1: *feature set 1 (upper features) and feature set 2 (all features) (WORD, DOMAIN and KEYWORD are placeholders)*

Name	**Explanation**
BIAS	feature which always holds
WORD in sentence	for every WORD in the sentence
? in sentence	true if there is a ? in the sentence
dialogue-start	true if it is the first sentence in a dialog
dialogue-end	true if it is the last sentence in a dialog
DOMAIN-label in sentence	true if DOMAIN is in the sentence
DOMAIN-label in sentence-1	true if DOMAIN is in the previous sentence
DOMAIN-label in sentence-2	true if DOMAIN is in the penultimate sentence
KEYWORD-keyword in sentence	true if KEYWORD is in the sentence
KEYWORD-keyword in sentence-1	true if KEYWORD is in the previous sentence
KEYWORD-keyword in sentence-2	true if KEYWORD is in the penultimate sentence

Table 2: *Distribution of the domains*

Domain	**Occurences**
'Train'	10,951
'Hotel'	10,756
'Restaurant'	10,246
'Attraction'	7,195
'Taxi'	2,850

- also search for synonyms of the predefined keywords using synsets from WordNet.
- assign multiple domains

To find keywords, we looked through the training set and searched for co-occurences between specific words and domains. We selected between four and eleven keywords for each domain.

Baseline 1, 3, 5 and 7 search for one keyword in the sentence while baseline 2, 4, 6 and 8 search for all our keywords inside a domain. Baselines 3 and 4 assign multiple domains, Baselines 5 and 6 use WordNet to find synonyms for all keywords and baselines 7 and 8 do both.

3.2. Implementing the multi-class perceptron

Our topic tracker uses a multi-class perceptron to predict if there is a transition to another topic and to which other topic or if there is no transition. It classifies utterances by learning which features co-occur with which domain tag, and derives its prediction off of these observations.

Table 3 shows the features and feature templates for different sentences. The features of a sentence are mainly the words it is built off and special words which appear in it or in its preceding context (two sentences to the left). Keeping in mind that we want the topic tracker to be able to classify domains of sentences in real-time we couldn't use features which considered the context to the right of the sentence as this would not be possible in a real-time dialog. We used two feature sets FS1 and FS2. FS1 is a subset of FS2 and only uses features which can be easily obtained for any given data. FS2 on the other hand also looks at the keywords we selected by manually looking through the training data and judging what keywords occur often with specific domains. FS1 is therefore much more generalizable and can be trained with different data sets to predict different domains, while FS2 shall achieve better results on the specific data set we used. The features 'DOMAIN-label in sentence' and 'KEYWORD-keyword in sentence' could seem counter-intuitive as each word in the sentence already exists as a feature, however by having two features for the same word, we are hoping to achieve better results by putting extra emphasis on these important words.

We decided not to classify every sentence with its domain but instead only classify the sentences in which a transition from one domain to another occurs. This doesn't make sense for tasks like part-of-speech-tagging because a part-of-speech is no entity which spans over multiple tokens however a topic or domain is very likely to span over multiple sentences in a dialog. This way the algorithm can not only learn what sentences are likely to appear in specific domains, but also what sentences are likely to stay in the same domain.

The model generates a unique feature set for each sentence, which represents the sentence's feature vector in a sparse way as it only shows the features that are activated in the sentence. This corresponds to a vector showing '1' in the positions of the activated features and '0' in all other positions. It then creates a perceptron for each domain tag, representing the transition to this tag. A perceptron for 'no transition' is also created which shall be used when the current sentence's domain is equal to the previous'. The weight vector of each perceptron is initialized by assigning random weights between '-1' and '1' for each feature.

To predict the transition-tag of a sentence, a likelihood score is calculated for each perceptron by taking the dot product of the sentence's feature vector and the perceptron's weight vector. The perceptron with the highest score is then chosen as the predicted domain tag. After predicting a transition tag for each sentence, the algorithm runs a relabel function which iterates through every dialog and substitutes each 'no transition'-tag with the domain-tag of the previous sentence.

Learning in our case is more complex than in a standard classification setting because we are predicting transitions and not domains. he multi-class perceptron learns by adjusting the weights of the perceptrons depending on the correctness of its prediction. Normally if the prediction is wrong it adds the feature vector (consisting of binary positions) to the weights of the perceptron with the predicted tag and subtracts it from the weights of the preceptron with the gold tag. As we faced certain problems which will be discussed more in detail in the error analysis, we changed the value which gets added or subtracted to a value which is proportional to the epoch count, so that after each epoch the adjustment gets smaller.

In a standard classification setting the algorithm would check if the predicted tag is equal to the gold tag and adjust the weights of the two perceptrons if they are not the same. How-

Table 3: *Performance of the different classifiers and configurations*

Classifier	**Macro-averaged F-Score**
BASELINE 5: one keyword + WordNet synsets	37.0%
BASELINE 1: one keyword	37.0%
BASELINE 6: multiple keywords + WordNet synsets	39.5%
BASELINE 3: one keyword, multiple assignments	43.1%
BASELINE 7: one keyword + WordNet synsets, multiple assignments	43.1%
BASELINE 2: multiple keywords	54.4%
BASELINE 8: multiple keywords + WordNet synsets, multiple assignments	56.7%
BASELINE 4: multiple keywords, multiple assignments	61.3%
multi-class perceptron with FS1	85.4%
multi-class perceptron with FS2	86.4%

ever in our case we can not just compare the predicted tag with the gold domain as there are no 'no transition'-tags in the gold domains. To test if a predicted 'no transition'-tag corresponds to the gold domain, the algorithm compares the gold domain of the current sentence with the gold domain of the previous sentence. If they are the same and the predicted tag is not 'no transition' the weights get adjusted. If they are not the same and the predicted tag is different from the gold domain the weights also get adjusted. In every other case the weights remain the same.

4. Experiments

4.1. Experimental Setting

The multi-class perceptron was trained and tested on a pre-processed version of the MultiWOZ 2.1 dataset [5]. The data we got provided with was pre-processed in a way that for every sentence only the domain was specified as this was the only relevant information for our task, whereas in the original data many different slots were specified. The data included sentences for seven different domains: 'Hotel', 'Attraction', 'Taxi', 'Train', 'Restaurant', 'Police' and 'Hospital'. As only nine sentences were labeled with 'Police' and only five were labeled with 'Hospital', we decided to remove the dialogues the sentences appeared in. This decision was motivated by our aim to construct a reliable topic tracker, which would be difficult for domains which appear that infrequently in the data. Also 42 instances were not labeled and there were five dialogues with no content, so we manually labeled the 42 instances and deleted the five empty dialogues. After applying those changes to the data we ended up 6,304 dialogues. We decided to do a 80/20-split to construct the training and the test set. As sentences in the same dialog should not be split in such a task, we split the data on dialog level. This resulted in 5,044 training dialogues with 40,074 sentences and 1,260 test dialogues with 10,069 sentences. Because sentences are nearly equally distributed across the dialogues the relation of sentences is also 80/20. Only 1799 sentences are labeled with more than one domain. You can see in Table 2 that the distribution of domains is very similar for 'Train', 'Hotel' and 'Restaurant', but smaller for 'Attraction' and even smaller for 'Taxi'. This uneven distribution along with the fact that we want the classifier to be reliable in all given domains, motivated us to use the macro-averaged F-score for evaluation. The macro-averaged F-score shows the balance of precision and recall averaged against all five domains and in contrast to the micro-averaged F-score it treats all classes equally – e.g the score of a class with 5 instances is weighted equally as the score of a class with 50,000 instances. We will first compare the different baselines to find the best performing one and will then compare our multi-class perceptron with the two feature sets to it.

4.2. Results

We expected that baseline 1 would perform the worst as it was the baseline that all other baselines built on. However baseline 5 performed equally bad even though it used WordnNet to look for synonyms. The baselines using multiple keywords performed better than the ones using only one keyword with the exception of baseline 6 which looked at multiple keywords and their synonyms. Given that applying the same settings but allowing multiple assignments (baseline 8) results in the second best score, our explanation for the bad performance of baseline 6 is that the amount of words it searches for often triggers multiple domains some of which could be right, but then it just picks one at random. The best configuration by far only uses multiple keywords and assigns multiple domains. We think that the keywords we picked were specific enough to capture most domains and using WordNet created synonyms which were to general.

The multi-class perceptron was trained for 40 epochs before predicting. It outperformed baseline 4 with both feature sets by more than 20 percent. FS2 performed slightly better than FS1 which could be attributed to the fact that the algorithm paid more attention to important keywords with FS1 than with FS2.

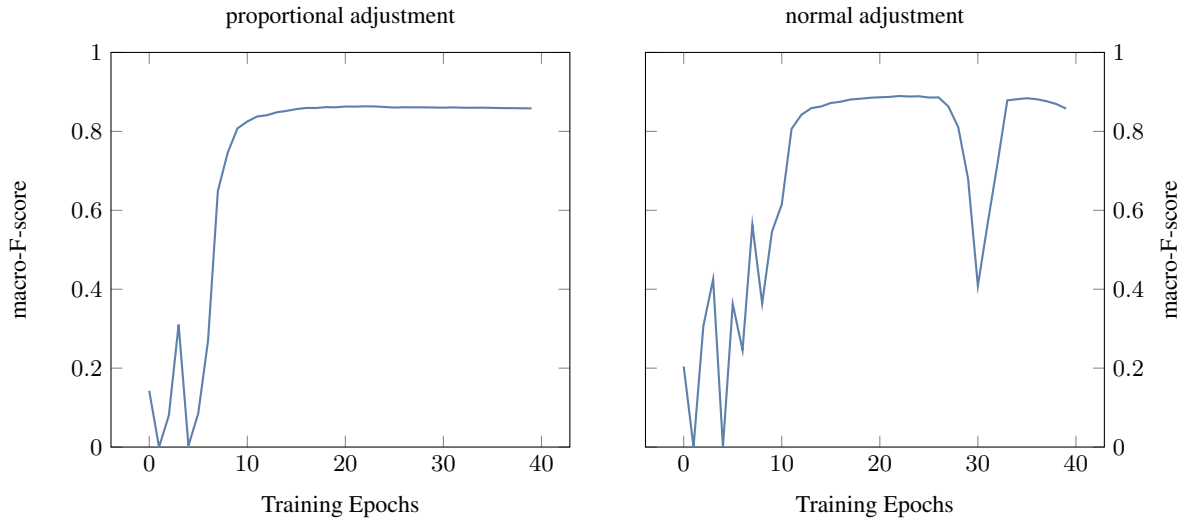
4.3. Error Analysis

When implementing the multi-class perceptron we faced a problem with the development of the scores over epochs. The learning curve was highly unstable and it sometimes ended up with good training scores which resulted in good predictions but also sometimes with bad training scores and bad predictions. We therefore decided to change the weight adjustment so that the weights get adjusted less and less the higher the number of epochs is.

In figure 1 the right plot shows the learning curve with normal adjustment and the left plot shows the learning curve with proportional adjustment. As the learning curve of the latter was different every time we ran the algorithm, this is just one of many examples. In the first five epochs both curves go up and down however in the next 5 epochs the left curve constantly goes up at a very high rate until about 0.8. After that the growth decreases until the scores reach an F-score of 0.859 and in the remaining epochs the scores only vary between 0.858 and 0.860 and it ends up with a score of 0.858.

The right curve is oscillating much more even after the first five epochs and the growth only stabilizes after the eighth

Figure 1: *Development of* ***macro-averaged F-score*** *while training the classifiers with FS2.*



epoch. After that it goes up at a high rate until it reaches epoch 12 then grows at a small rate up to epoch 26 until it reaches a score of 0.889. This late into the training process the curve rapidly goes down to a score of 0.409 and goes up again to a score of 0.881. The last remaining epochs, the curve goes down again at a small rate and ends up at a score of 0.858.

The learning curve for the classifier with normal weight adjustment reaches a maximum of 0.890 and ends up with a score of 0.858 while the classifier we used reaches a maximum of 0.863 and also ends up with a score of 0.858. However the maximum of the right curve shows that the final score could have ended up better than our model. While using proportional adjustment leads to really similar results in every run, using normal adjustment leads to very different results in every run and sometimes even leads to better results on the test set than the ones we got.

Because our aim was to build a reliable topic tracker and we couldn't find a better solution, we decided to use proportional weight adjustment for our final model. Considering this and the potential the right curve shows, this would be a good starting point for doing research on how to improve the classifier.

5. Discussion

In this paper we showed that topic tracking in ADVISER can be improved by implementing a multi-class perceptron, which classifies sentences into domains based on different features. Even without the multi-class perceptron, we were able to improve the basic keyword-matching-model by extending it with multiple keywords per domain, using WordNet for synonyms and allowing multiple assignments for sentences. The results showed that the inclusion of WordNet does not necessarily lead to better results, as there is a higher possibility that a keyword appears in more than one domain. Adding multiple keywords to the domains and allowing the domain tracker to assign more than one domain to a sentence however improved the performance a lot.

Considering that the multi-class perceptron is a machine learning algorithm, it makes sense that it performed much better than keyword-matching. Putting more emphasis on important keywords did not have the effect we expected as the scores for FS1 and FS2 were really similar. It seems that the algorithm already acquires enough information when considering the words in a sentence and only searching for domain names the context the current and preceding context. This is good considering that FS2 is far more specific and fitted to the data than FS1 is. Using FS1 leads to similar results and it is easier to train with different data sets and for different domains.

We achieved our goal of improving the topic tracker in ADVISER and ended up with a reliable topic tracking algorithm which is easily extendable to more domains.

6. Future Work

In the future we could try to improve the classifier. Specifically fixing the weight adjustment problem in a way that the classifier reliably achieves its best or close to best performance when training is done, could improve it by at least two if not more percent as can be seen in the error analysis section. Also the features sets still are very basic and we could try to extend them to better capture unseen data.

Another idea would be to try out our classifier on the TourSG data set and compare the Multi-class Perceptron to the dynamic memory networks of [3].

7. References

[1] C.-Y. Li, D. Ortega, D. Väth, F. Lux, L. Vanderlyn, M. Schmidt, M. Neumann, M. Völkel, P. Denisov, S. Jenne, Z. Kacarevic, and N. T. Vu, "Adviser: A toolkit for developing multi-modal, multi-domain and socially-engaged conversational agents," 2020.

[2] S. Kim, R. Banchs, and H. Li, "Exploring convolutional and recurrent neural networks in sequential labelling for dialogue topic tracking," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 963–973. [Online]. Available: https://www.aclweb.org/anthology/P16-1091

[3] S. Kim, "Dynamic memory networks for dialogue topic tracking," 2019.

[4] S. Kim, L. F. D'Haro, R. E. Banchs, J. D. Williams, and M. Henderson, "The fourth dialog state tracking challenge," in *Dialogues with Social Robots*. Springer, 2017, pp. 435–449.

[5] M. Eric, R. Goel, S. Paul, A. Kumar, A. Sethi, P. Ku, A. K. Goyal, S. Agarwal, S. Gao, and D. Hakkani-Tur, "Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines," 2019.