Report for Coreference Resolution

Brief Introduction

In this project, we used the syntactically annotated German corpus -TüBa in CONLL12 format as our data resource. In part 1, We extracted mentions from train.tueba.coref.txt and then created gold chains and gold standard, and based on that, we extracted features. In part 2, we generated markables dataset as our test dataset and training models by learning the features we extracted. Finally, we evaluated the performance of models on our training set and test the models with extracted markables and true labels.
Due to a different understanding of the features that we were supposed to extract, we followed a parallel approach and used 2 methods to generate the features.

Workflow

1.Mention and gold chain extraction

1) Method1

Using the 'train.tueba.coref.txt' file, we matched and extracted all the mentions by the last column of the file. Storing all the mentions in a form of [coref_id,seg_id,sent,starttoken,endtoken] in a stack. Then generated the gold chain based on the seg_id (The id for the segment) and coref_id(The id for the correfence). The form of gold chain is goldchains= {'segment_id' :[[(coref_id, sent_id, start_token_id, end_token_id)]]}.

2) Method2

Follow a similar step in Method1, the mentions extracted in Method2 are in a form of [coref_id, segment, sentid1, tokenid1, (sentid2, tokenid2)], and then work out the tuple ordered by doc_id, sentence_id and start_pos in a form like: goldchains=(doc_id, sent_id, start_position, end_position, coref_id).

2. Gold standard extraction

1) Method1

Using 'train.tueba.sync.txt', extracted features like 'head', 'PoS', 'gramFct' and so on to the matching mentions. Then for every segment, we pairs the markables in gold chain and add all features we extracted as positive samples. For the negative sample, we compare the every neighbor pair between mention i and j. In this part, we finally generate 79637 positive samples (label=1) and 323997 negative samples (label=0) with single features include [distance, POS1, gramFct1, Case1, Gender1, Number1, POS2, gramFct2, Case2, Gender2, Number2] and matching features including [sameHead, samePOS, sameGramFct, matchOrNot, sameCase, sameGender, sameNumber].

2) Method2

Storing the extracted single features in a nested dict: feature_dict[doc_id][sent_id][token_id] = [pos_tag, grammarfunct, head, (person, number, gender)]. Matching single features between start_ and end_token. Then followed a similar process of negative samples generation, exclude the positive mentions pairs and find the negative mentions in all mention pairs.

3.Markables extraction

1) Used the information from the 11th column in file ' test.coref.txt' to extract all the named entity as markables.
2) Extracted all the noun phrase , noun and pronoun as markables based on our experience about what kind combinations STTS tag can form a noun phrase.
3) Combined the above two results and sorted the result based on the sentence id and segment id.
4) Used the information from the last column in file ' test.coref.txt' to extract true mentions.
5) Compared the result from step3 with the result from step4 and using window-based strategy to generate positive and negative samples.

4.Model training and evaluation

1) Data preprocessing

Split the gold standard features dataset to train an evaluate set with a ratio of 7:3. And read the extracted markable dataset as test set. Observe the distribution of training and testing set and find the correlation

between matching features. Use label encoder to transfer all the features into numeric one. And then encode every features in one hot encoding.

2) Model training, evaluating and testing

Using different classifiers to do the classification of positive and negative samples. Evaluate the model by calculating the accuracy of cross validation. Tuning the parameters of the models(MLP and SVM) by GridsearchCV methods and applying test set on the optimized models to compare the performance of different models by the precision score (TP/(TP+FP), recall score (TP/(TP+FN)) and f1 score (2*(Recall * Precision) / (Recall + Precision)).

Results (best performance)

features	models	Tuned parameters	Precision score	Recall score	F1 score
All	SGD	Default	0.27	0.03	0.06
All	SVM	Default	-	-	-
All	Decision Tree	Default	0.40	0.36	0.38
All	Naïve Bayes	Default	**0.5**	**0.43**	**0.46**
All	Logistic Regression	Default	0.15	0.035	0.06
All	Perceptron	Default	0.33	0.32	0.33
All	Multi-Layer-Perceptron	solver='adam', alpha=1e-5, activation='relu', hidden_layer_sizes=(200,200), random_state=1, max_iter=200, batch_size = 512	0.48	0.31	0.38
All	KNN	Default	0.44	0.25	0.32

The results for Method2 were much lower and are therefore omitted.

Conclusion

The best performance model for this task is Naïve Bayes with all features, it has the best f1-score=0.46. Comparing with our evaluation accuracy by cross validation in SGD, which is as high as 0.87, we assume the low scores maybe due to the features we chose were not suitable for a Binary classification problem. We can try to extract more features from the raw data and improve our markable extraction results in the future research.