Project Proposal - Machine Learning Project

Text-based speaker identification

Anonymous ACL submission

1 Introduction

Communication is a process in which information is exchanged between individuals through a common system of symbols, signs, or behavior. One of the most important ways of communication is spoken communication. There are differences between people in the ways they verbalize their thoughts through dialogues. In this paper we investigated the possibility of distinguishing between different speakers in dialogues. This can be used for, for example, identifying the writer of an anonymous hateful message or an anonymous message which contains communication about a terrorist attack. Being able to identify these anonymous messages can be of great importance, because only then specific actions can be taken. There are various experiments which showed significant differences in the linguistic patterns generated by different participants, which suggests the possibility to perform speaker identification with only text-based data. This paper focuses on the topic text-based speaker identification and creating and using models that can distinguish between different speakers from two different movie scripts. This can be helpful in for example movie question answering [4] or automatic identification of character relationships. Identifying speakers is a critical task for understanding the dialogue and storyline in movies. We used a dataset of the series South Park and a dataset of the series Game of Thrones to compare the results and see if there are any differences between the performances. Our research question states: "Are statistical models useful to distinguish between different speakers in the movie scripts of Game of Thrones and South Park?"

2 Relevant Literature

In Kaixin et al. [3] they performed text-based speaker identification of the six main characters in the dialogues occurring in the first 8 seasons of the TV show Friends. For implementation they used a Convolutional Neural Network (CNN). Contextual information is ignored when using a simple CNN, but since every utterance is highly related to its prior utterance they used a modified CNN were the previous two utterances and the subsequent utterance in the same scene are taken into account. In addition, they used utterance concatenation where all of the utterances for each speaker in one scene are concatenated in the original dialogue order. Their best model using multi-document convolutional neural network shows an accuracy of 31.06% and a macro average F1 score of 29.72. A Recurrent Neural Network (RNN) was also considered in their research, but got an accuracy of 16.05%.

In Serban et al. [2], they take a data-driven approach to learn turn changes and speaker identity from an open-domain corpus of movie scripts. The Movie-Scriptolog dataset (Serban et al., 2015) was used, consisting of 614 movie scripts. The dataset was developed by expanding and preprocessing another movie script corpus (Banchs, 2012), based on The Internet Movie Script Database1. They only took the 3 main characters per movie into account, all the others got the label 'Others'. Majority baseline ('Others') obtained an accuracy of 59.57%. They used a Recurrent Neural Network (RNN) as their own model, where the previous t-tokens before the classification label, and the next t-tokens after the classification label were taken into account, this model got an accuracy of 68.92% They also used a modified RNN with Word2Vec embeddings which were trained on the Google News dataset, containing about 100 billion words, this resulted in a slightly improved performance with an accuracy of 69.47%. Logistic Regression

was also used, but did not outperform the baseline on this task, in contrast to RNN models, which clearly outperforms the baseline.

In Azab et al. [1] they used a model for speaker naming in movies that leverages visual, textual, and acoustic modalities in an unified optimization framework. Their dataset consists of a mix of TV episodes and full movies. They addresed the task under a setup without a cast-list, without supervision from a script and dealing with the complicated conditions of real movies. The model includes textual, visual, and acoustic modalities, and incorporates several grammatical and acoustic constraints. Their experiments show that the multimodal modal significantly outperforms several competitive baselines on the average weighted F-score metric.

3 Dataset

We gathered two datasets on TV Shows, Game of Thrones and South Park. At first, we wanted to use a third dataset with the script of the series Friends, but this dataset turned out to be not pleasing to use, to which we decided not to use it anymore. The datasets for Game of Thrones and South Park already contained just the dialogue script. We downloaded these scripts from Kaggle. We used nltk's tokenizer to tokenize the dialogue from both datasets, to get a better sense of how they relate to each other in size.

A quick analysis of the gathered data gave us the following insights: South Park has 354480 lines of dialogue made up out of 1071788 tokens, the dialogue is spoken by 3950 different characters. Game of Thrones has 143466 lines of dialogue made up out of 356783 tokens, the dialogue is spoken by 564 different characters.

We created a dataset for both scripts containing the names of the characters and their lines of dialogue. We split these sets into train- and test-sets of respectively 75% and 25% of the data. We tokenized our data using NLTKs word_tokenize function and used sklearn's CountVectorizers built in preprocessor, which strips accents and transforms all strings to lowercase. Afterwards we transformed the tokens into bigrams or trigrams. We then created feature vectors using CountVectorizer. We also made a custom preprocessor that transformed strings to lowercase, removed commas, normalised all end-of-sentence characters and removed certain other punctuation marks that cluttered the data. The results of both preprocessors were very similar. The results included in this paper are those where we used the built in preprocessor. Because both datasets contained a lot of classes with very little instances, we decided to keep an n number of characters that have the most text, and grouped the rest as one class named "Other". Before we started our project, our target point was to keep 10 characters, but during analyses we found out that it was better for our research to experiment with a different amount of characters. We experimented with n being 3, 5, 7 and 10. To see if the model performed differently, in addition to running it with the "Other" class, we also ran our experiments on the same subsets while excluding the "Other" class.

4 Methodology

In our research we used the majority class per dataset as the baseline, which our own model should outperform. For our own model, we experimented with Naive Bayes, Support Vector Machine (SVM) and K-nearest neighbor algorithm (KNN). For the Naive Bayes we implemented a Gaussian Naive Bayes using the scikit learn, we also tried the Multinominal Naive Bayes and the Bernoulli Naive Bayes, but the Gaussian seems to be the best one. It was not possible to put in the whole dataset and all the features, this results in a memory error. So we experimented with smaller parts of the dataset and/or less features, but different combinations showing only very little differences in results, the best results we obtained when using the whole dataset and 20000 features. The SVM was also implemented using scikit learn, we tried different kernels, the 'rgb' one, the 'linear' one and the 'polynominal' one, but we decided to use the linear kernel, since it gave the highest results and is the most efficient kernel. Memory issues also occured when running all the features and the whole dataset. Since SVM is a lot slower then the other models, the best results we obtained when using 10000 instances and 10000 features. For KNN we also used scikit learn, for the value of k we tried 35 different K's and it will use the K value which will returns the highest accuracy. For KNN we had the same memory error when using all the features and the whole dataset, we experimented with smaller datasets and less features, the best results are obtained when using the 20000 rows of the dataset and 20000 features.

5 Results

Tables 1 and 2 show the Naive Bayes results on both South Park and Game of Thrones data. These are the results without including the "other" class, since it does not score better than the baseline in any of the cases and in any of the models. As can be seen, for the South Park data, the model only scores better when including 3 characters, this is a small improvement of 0.03. With more than 3 characters, the model does not beat the baseline in any of the cases. For the Game of Thrones data, the model scores better than the baseline in all cases, it scores best with 3 and 5 characters, this is an improvement of 0.05. So in the case of Naive Bayes, the model scores better with the Game of Thrones data than with the South Park data, and with all numbers of characters the model scores best with the bigrams.

The tables 3 and 4 show SVM's results. As can be seen, the results are now better for the South Park data than for the Game of Thrones data. For both datasets and for any length of characters, the model beats the baseline. The maximum improvement here is 0.11 with the South Park data, when using 10 characters and unigrams. In both the South Park and Game of Thrones data, the model scores best for unigrams at all numbers of characters. For the Game of Thrones data, the best improvement is 0.08, at 5 characters.

Tables 5 and 6 show the results of the KNN model, table 5 is the South Park dataset and table 6 is the Game of Thrones dataset. When we tried with the 'Other' class the results were in all cases the same as the baseline, since it would simply pick the majority class, so these results are not included in the tables. As you can see the results are the highest for each number of characters when we use the unigrams as features, but this only improves the baseline just a little. For the Game of Thrones dataset the only model that outperforms the dataset is the one with the 5 characters and unigrams, but this is only a 0.01, the other ones are even worse than the baseline.

6 Conclusion

Returning to our research question of whether statistical models are useful to distinguish between different speakers in the movie scripts of Game of Thrones and South Park, we conclude from our results that statistical models might not be the most suitable for this task. The results showed that sometimes our models cannot even beat the baseline, and when they do, it is a very small improvement. We can conclude that SVM is the best model to use here, as it shows the best results. It always gives an improvement of at least 0.05 and maximum 0.12, compared to naive bayes with a maximum improvement of 0.05, and KNN with a maximum improvement of 0.03. We suggest that in future research neural networks should be used in order to see if those are more suitable for this task than statistical models.

References

[1] Mahmoud Azab, Mingzhe Wang, Max Smith, Noriyuki Kojima, Jia Deng, and Rada Mihalcea. Speaker naming in movies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2206–2216, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[2] J Pineau IV Serban. Text-based speaker identification for multi-participant open-domain dialogue systems. 2015.

[3] Catherina Xiao Kaixin Ma and Jinho D. Choi. Text-based speaker identification on multiparty dialogues using multi-document convolutional neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics-Student Research Workshop*, pages 49–55, 2017.

[4] Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Movieqa: Understanding stories in movies through question-answering. 06 2016.

Naive Bayes	Baseline	Unigram	Bigram	Trigram	Improvement
Characters: 3	0.39	0.40	0.42	0.42	0.03
Characters: 5	0.34	0.25	0.32	0.29	x
Characters: 7	0.32	0.20	0.28	0.25	x
Characters: 10	0.29	0.19	0.27	0.25	x

Table 1: Naive Bayes, South Park

Naive Bayes	Baseline	Unigram	Bigram	Trigram	Improvement
Characters: 3	0.46	0.49	0.51	0.45	0.05
Characters: 5	0.30	0.33	0.35	0.26	0.05
Characters: 7	0.24	0.25	0.26	0.22	0.02
Characters: 10	0.21	0.20	0.23	0.16	0.02

Table 2: Naive Bayes, Game of Thrones

SVM	Baseline	Unigram	Bigram	Trigram	Improvement
Characters: 3	0.42	0.53	0.49	0.45	0.11
Characters: 5	0.34	0.44	0.40	0.35	0.10
Characters: 7	0.34	0.45	0.40	0.35	0.11
Characters: 10	0.32	0.44	0.39	0.34	0.12

Table 3: SVM, South Park

SVM	Baseline	Unigram	Bigram	Trigram	Improvement
Characters: 3	0.46	0.52	0.50	0.46	0.06
Characters: 5	0.30	0.38	0.34	0.31	0.08
Characters: 7	0.24	0.30	0.27	0.24	0.06
Characters: 10	0.21	0.26	0.23	0.22	0.05

Table 4: SVM, Game of Thrones

KNN	Baseline	Unigram	Bigram	Trigram	Improvement
Characters: 3	0.44	0.47 (K=39)	0.42 (K=3)	0.37 (K=5)	0.03
Characters: 5	0.37	0.39 (K=23)	0.30 (K=11)	0.37 (K=1)	0.02
Characters: 7	0.34	0.36 (K=63)	0.32 (K=1)	0.35 (K=1)	0.02
Characters: 10	0.34	0.36 (K=61)	0.26 (K=5)	0.33 (K=21)	0.02

Table 5: KNN, South Park

KNN	Baseline	Unigram	Bigram	Trigram	Improvement
Characters: 3	0.48	0.46 (K=61)	0.45 (K=61)	0.43 (K=15)	x
Characters: 5	0.33	0.34 (K=59)	0.29 (K=69)	0.33 (K=1)	0.01
Characters: 7	0.27	0.24 (K=55)	0.22 (K=27)	0.24 (K=57)	x
Characters: 10	0.23	0.19 (K=55)	0.18 (K=63)	0.02 (K=67)	x

Table 6: KNN, Game of Thrones