Movie Genre Detection Based on Subtitle Analysis Methods

Enes Akdoğan, Ömer Akgül, Ali Burak Erdoğan,
Mesut Gürlek, Mustafa Yildiz

Group 4, CS 464 Introduction to Machine Learning course, Department of Computer Engineering, Bilkent University, Ankara
Email: a@b.com

In this project, we aim to categorise movies into genres by analysing subtitles with machine learning techniques. These techniques focus several aspects of the subtitles such as sound descriptions, dialogue frequency and raw subtitle scripts. Some of the applications offer novel solution for movie genre classification problem.

1. INTRODUCTION

A genre of a movie is a category based on similarities in the characteristics and elements of the movies. According to academics works on this area, recognising the genres are easier than to define. Therefore identification of genres in a rigid way is highly controversial.[1] This makes the genre classification a suitable area to apply machine learning techniques to categorise genres based on similar features that exist in the script of the movies. In this project we attempt to classify movies into a set of genres by analysing subtitles of the movies in distinct ways since subtitles are captions of the movies' transcripts.

We approach this problem from three main perspectives. (i) The most straight-forward method is to consider the subtitle texts as collection of words disregarding the grammar and word order. This produces bag of words that contains the multiplicity of the all words occur in input data. Since word order is not taken into account, time related information for the movie is also not considered. We have implemented methods that use this approach and this will probably have the most effect in our final model. (ii) The second approach is to use the descriptions that exist for people who has hearing impairment exist in certain subtitles. In order not to run into problems we primarily downloaded subtitles that have this kind of sound descriptions. There are a few amount of subtitles that do not have hearing impaired sound descriptions therefore we simply ignore these subtitles when training. We sift out the sound descriptions from the text and ran the same learning algorithm as the first method. The idea of the second approach is similar the the first one because both of them ignore the time factor and directly try to learn chunks of words. (iii) The third approach is a novel approach that analyses the speaking frequency as a whole and also in certain periods. This approach allows us to make a connection between time and text which differentiates the project from machine learning applications on plain text. The implementation details for above three perspectives will be discussed further in the section 4. It is ensured that during the training of models, none test data is used. To prevent overlapping, training and test data are split.

In this final report, the work done after the progress report for the points mentioned above is discussed and explained as follows. Section 2 mentions the problem description of our project. Section 3 describes the input data set and procedures applied on these raw sources until the progress milestone. Section 4 tells about the work done afterwards. In section 5, work performance of the explained methods are shown. Section 6 discusses the results. Afterwards, report is concluded.

2. PROBLEM DESCRIPTION

Our problem of focus is to classify movies solely on the subtitles of these movies. In this work we tried to find ways of classifying movies based on subtitles and combine these ways to increase accuracy. We wanted to find methods of generating models for each main feature of the subtitles and combining these models. The main features were: hearing impaired sound descriptions, the raw text as a whole, and activity frequencies such as dialog per minute and sound per minute. This report explains the details of our work based on the progression of the progress report.

3. METHODS

3.1. Data

For our project, the basic necessity is collecting proper and useful data. We needed to obtain regular and correct subtitle files, which should include sound descriptions for hearing impaired people as well. Since it was certain that our requirement is over hundreds of subtitles, we would not download it by hand. We used web crawling framework Scrapy[2] to search distinctive movies for each category on IMDb web

Category	Number of Subtitles
Action	1161
Comedy	1174
Crime	1172
Horror	1130
Musical	1124
Romance	1185
War	1102
Western	985
Total:	9033

TABLE 1: Number of subtitles for each category

site, and OpenSubtitles API[3] which serves more than 3 millions of subtitles to both normal users and developers. We built a script which automates the process of movie searching and downloading its hearing impaired supported subtitles for each category. However, OpenSubtitles provides developers with a limited download rights (200 subtitles per user in a day), and this slows the data collecting part. We managed to download more than 9000+ most popular films from different categories. (At least 6000 of them are with hearing impaired support.)

3.1.1. How data separated and processed?
First we divided whole data as 20% test data and 80% training data. We applied cross-validation to tune parameters before training actual model. To apply cross validation, we divided training data set as 80% percent training and 20% validation random sets in each iteration. There were several parameters for different models and each parameter values has several iterations and in each iteration we randomise whole training data set, divide it into new training and validation sets then predict validation set and find mean of the accuracies for corresponding parameter value. After this process, we pick the parameter with highest accuracy value. With optimal parameter value we train with whole training data set and predict the test data set.

3.2. Speech Density and Frequency Feature

Firstly, we calculated number of dialogues per minute of movies in order to find a pattern for classifying movie categories. We did not expect big differences between categories in terms of dialogue per minutes. However, as it be seen from below figure there are some important differences. As known, comedy movies mainly focus on talks of actors and there are many conversations in these movies. So there are approximately 16.71 dialogue per minutes in comedy movies, it is the highest number in DPM's. On the other hand, there are not many conversations in horror movies because usually silence is dominant in these movies. According to our data approximately there are 10.86 dialogue per minute in horror movies and it is the lowest number in DPM's. Refer to following figure 1 and table 2 for detailed numbers.

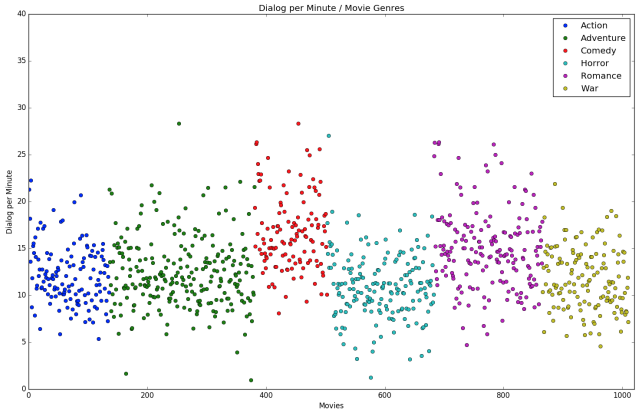

FIGURE 1. Dialogue per Minute for all movies. Same movie genres are denoted by same colour.

Category	Dialogue per Minute
Action	12.20
Adventure	12.39
Comedy	16.71
Horror	10.86
Romance	14.97
War	12.88

TABLE 2: DPM values for each category

Secondly we calculated number of words per minutes of movies, in order to improve our category prediction accuracy. This date is more distinctive than dialogue per minute. As it can be seen from below figure and table, there are differences between word per minutes of these categories. Action and Adventure categories are very similar but other categories have specific numbers about wpm. Similar to DPM data, Comedy movies have highest value for wpm with 69.27, Romance movies follows comedies with 61.96. On the other hand, Horror movies have lowest value for WPM with 43.29. Figure 2 and table 2 indicate these results. We applied kth

Category	Word per Minute
Action	50.47
Adventure	50.29
Comedy	69.27
Horror	43.29
Romance	61.96
War	54.15

TABLE 3: WPM values for each category

Nearest Neighbour Algorithms to DPM and WPM data and estimated accuracies with different k values. We tried k from 1 to 30 and when k is equal to 23 we reached the highest accuracy rate which is 32.22. It is not high

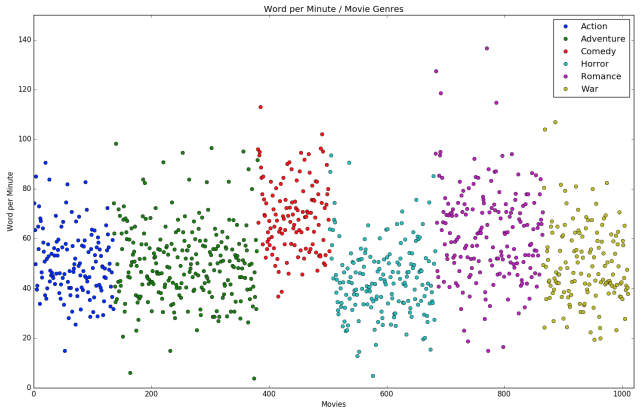

FIGURE 2. Words per Minute for all movies. Same movie genres are denoted by same colour.

accuracy rates alone but this will be additional feature for main algorithm and will increase its accuracy.

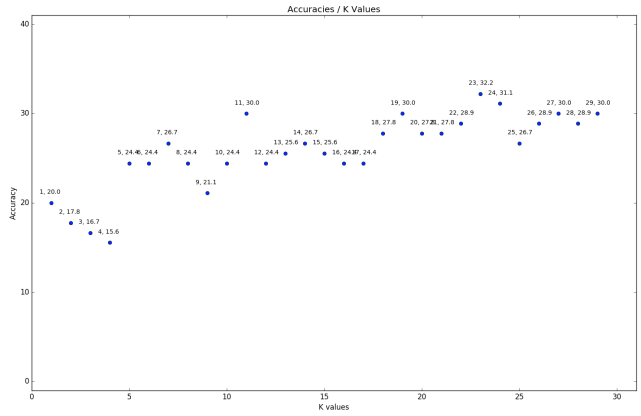

FIGURE 3. kNN Accuracies on DPM and WPM data for different K values.

K value	Accuracy	K value	Accuracy
1	20.0	16	24.44
2	17.77	17	24.44
3	16.66	18	27.77
4	15.55	19	30.0
5	24.44	20	27.77
6	24.44	21	27.77
7	26.66	22	28.88
8	24.44	23	32.22
9	21.11	24	31.11
10	24.44	25	26.66
11	30.0	26	28.88
12	24.44	27	30.0
13	25.55	28	28.88
15	25.55		

TABLE 4: kNN accuracies for different K values

3.3. Hearing Impaired Sound Descriptions Feature

3.3.1. Preprocessing

We have processed hearing impaired subtitles which includes audio texts for non-speech sounds. We applied the following processes. Hearing Impaired audio texts have consistent formats in subtitles. We first parsed only these audio texts. After tokenization, stop words are extracted and stems of the words are processed. Therefore, we eliminated unimportant words. Overall, we parsed only these audio texts and observed following word frequencies in categories.

3.3.2. Training and Accuracy:

After preprocessing, we used scikit-learn to learn our data. CountVectorizer provided a matrix with token counts. Tfidf vectorizer made our matrix more meaningful by weighting words. Then we used multinomial Naive Bayes to train our data. Naive Bayes takes an alpha parameter which is additive (Laplace) smoothing parameter. We changed alpha parameter to improve accuracy. Also, to provide consistent train and data we applied following methodology. We used 150 test subtitle and 688 training subtitle in our accuracy calculations. First, we choose an alpha value and run Naive Bayes 50 times then calculate mean of the accuracies. For each learning we shuffle our dataset order and choose test and train data. Hence, we provide a consistent and reliable accuracy for each alpha value. Below figure shows accuracies for several alpha values. Since, we have 6 category and random chance give us (100/6) 15% accuracy, sample results in the following figures seem promising.

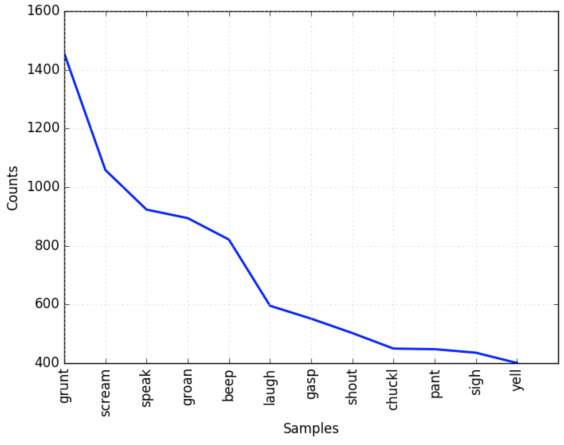

FIGURE 4. Top occurrence of words in Action Category

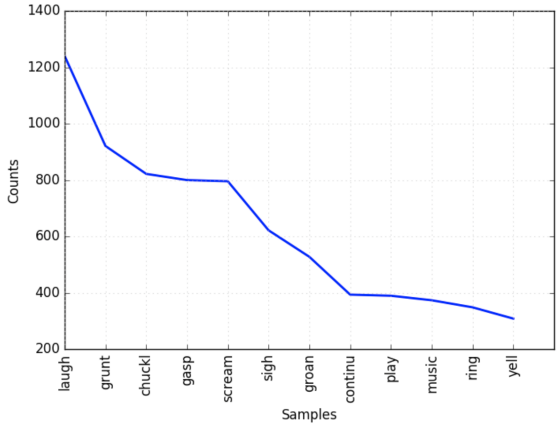

FIGURE 5. Top occurrence of words in Comedy Category

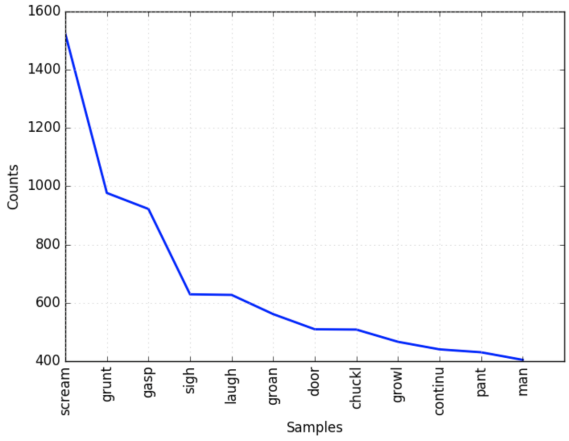

FIGURE 6. Top occurrence of words in Horror Category

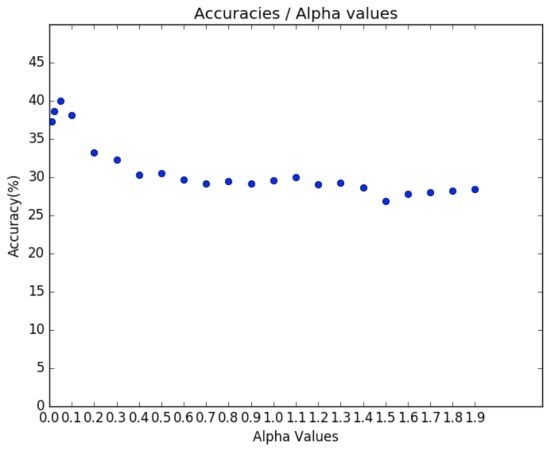

FIGURE 7. Accuracy for different alpha values

3.4. Whole text as a feature

3.4.1. Preprocessing

The preprocessing of all dialogues which is the bulk of the whole subtitle was done in a similar manner to the preprocessing of impaired subtitles. To get the raw text without the impaired parts we first used a regular expression filter to get rid of them. This was done with the same filter that we used in the separation of impaired dialogues. Tokenization, stop word removal, punctuation removal, and stemming was done using the Natural Language Toolkit (NLTK) [4] as was in the sound description part. When we took a look at word frequencies we saw that a lot of meaningless words existed for a bag of words implementation such as yeah, oh etc. To prevent these words from having an affect on the model we removed them.

3.4.2. Training and Accuracy:

Almost the exact same code with the hearing impaired part was used to do the learning. Details are discussed in the hearing impaired sound description section. The accuracies are slightly better being around 45%.

4. WORK DONE AFTER PROGRESS

4.1. Dialogue per Minute and Word per Minute

Previously, we used only 900 subtitles in total to train our WPM and DPM learning algorithm. Up to now, we increased the number of subtitles to more than 9000 (nearly 1100 from each category). We expected a high increase in our F1 scores by having 9 times greater number of data, however, we could not obtain such an improvement. Still, we have remarkable f1 scores for some categories such as Crime, Comedy, Romance. The main reason behind is, we think, those categories have significant difference from others in terms of dialogues per minute. Since horror is the lowest in WPM and DPM, it is easily predictable for the learning algorithm.

Action	50.92884278822343
Adventure	50.29885300837908
Comedy	70.20427639111396
Crime	59.13997907503586
Horror	46.04569823281735
Musical	74.99845382251573
Romance	62.0360795765368
War	55.79225354108949
Western	55.404143056896224

TABLE 5: New WPM values for each category

Action	11.752364947756044
Adventure	12.398248315034923
Comedy	15.543227618801078
Crime	12.997386458500177
Horror	11.029250406923195
Musical	14.046774517883888
Romance	13.973299009645306
War	11.637057205037284
Western	11.394383759610237

TABLE 6: New DPM values for each category

	precision	recall	f1-score	support
Action	0.11	0.14	0.12	66
Comedy	0.31	0.24	0.27	80
Crime	0.39	0.37	0.38	97
Horror	0.24	0.27	0.25	83
Musical	0.30	0.31	0.30	75
Romance	0.47	0.54	0.50	114
War	0.29	0.24	0.27	103
Western	0.15	0.16	0.16	68
avg / total	0.30	0.30	0.30	686

TABLE 7: Precision, recall, f1-score and support values for DPM and WPM classification for each category.accuracy: 24.12177985948478

4.2. Hearing Impaired and Model Combination

4.2.1. Classification with Hearing Impaired

We have 6000+ hearing impaired subtitles in our data set. These subtitles preprocessed in 2 phase. In the first phase, non-impaired and non-srt subtitles are filtered out. Then, subtitles are parsed to get only hearing impaired audio texts.

Important point of the input data organisation as following: We divided processed subtitles into 80% training data and 20% test data. Also all models in the combined model uses same training and test data.

We used Multinomial Naive Bayes to train hearing impaired texts. We applied cross-validation to tune alpha parameter before training actual model. To apply cross validation, we divided training data set as 80% percent training and 20% validation random sets in each iteration. There were 6 different alpha values and each alpha values has 50 iteration and in each iteration we randomise whole training data set, divide it into new training and validation sets then find mean of the accuracies for corresponding alpha value. After this process, we pick the alpha value with highest accuracy value. With optimal alpha value we train with whole training data set and predict the test data set. Below classification report gives details about precision, recall and f1 scores

	precision	recall	f1-score	support
Action	0.36	0.24	0.29	75
Comedy	0.29	0.30	0.30	69
Crime	0.34	0.35	0.35	68
Horror	0.51	0.57	0.54	58
Musical	0.62	0.54	0.58	80
Romance	0.26	0.31	0.28	65
War	0.64	0.66	0.65	65
Western	0.69	0.80	0.74	60
avg / total	0.46	0.46	0.46	540

TABLE 8: Precision, recall, f1-score and support values for Hearing Impaired Model classification for each category.Avg accuracy score: 0.462962962963

Although accuracy score seems very low, if we analyse each category by its f1 score; Horror, Musical, War, Western gives promising f1 scores for that particular categories. So predictions for these categories seems reliable.

4.2.2. Model Combination Logic

Since we get promising f1 scores for some categories in the previous model, we tried to combine our models using their f1 scores for each categories. We have 3 different models: MultinomialNB model using Hearing Impaired data set, Knn model using dpm(dialogue per minute), wpm(word per minute) and Multiclass SVM model using normal subtitles. Combining different models needs to generate a common format for model training, prediction and parameter tuning. Combining process can be discussed in 2 different title.

4.2.3. Providing API's for each Model

We have decide to provide a format to create APIs for each model. Each model API will be in python class format and these classes will be initialised with data sets for preprocesses. They have a 'tune_and_train' function to apply cross validation to get optimal parameters and then to generate model with optimal parameters. Another function is 'get_f1_scores' which returns f1 scores for each categories that are calculated within training data set with optimal parameters. Last function is 'predict' which takes single file path as parameter and returns label by using previously trained model.

4.2.4. Model Combination

3 models are generated kept in models list. Then , their 'tune and train' functions are called to tune parameters, apply cross validation. Next step is to get individual category f1 scores from each model and keep them in a list. In the next process we predict each single test data in an iteration and choose prediction of 1 model according to their f1 results that we keep in a list. This process is divided into two different prediction calculation. In the first method we predict a label for each model and look at the f1 score of corresponding

models corresponding category. We select label with best f1 score so that we rely on the models f1 score. In the second method, we sum f1 scores with same predicted labels the pick the label with highest f1 score. Below you can see the flow of the combination process.

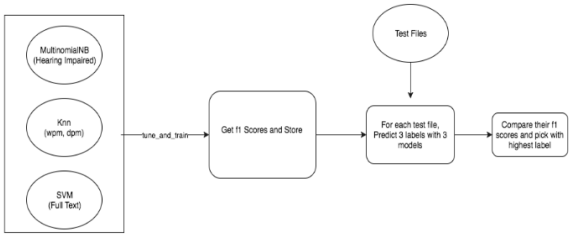

FIGURE 8. Model Combination Mechanism

4.3. Full Text Classification

This part is the continuation of the progress report section 'whole text as a feature'.

The implementation of the model has changed dramatically since the progress report. This was due to some experimentation with different methods that was advised in the feedback. Our accuracy rate has almost doubled with the following improvements and experiences.

We first tried to have a word2vec approach. The idea was to use a pretrained word2vec network in order to extract features from our subtitles. The extracted features would then be fed to a linear kernel support vector machine. We implemented this with the gensim and scikit learn libraries.[5] We expected much better results however, the results were worse than our simple bag of words and naive bayes implementation in the progress report. One possible explanation for this behaviour is that our data does not contain much meaning. When we take a quick look at some of the filtered and preprocessed subtitles we can see that there simply are too many slang and meaningless words. This is visible especially in the hearing impaired part of our model. The data does not carry meaning since it is mostly depictions and imitations of sound. After this experimentation with word2vec we decided to continue with the bag of words approach we had in the first place. This decision was also made under the influence of a blog post.[6]

Our second approach was to have a multilabel classification. This was unfortunately not so easy to implement due to our already collected data. However we drew inspiration from the assignments given in class and came up with a solution. The idea was to train 'one vs rest' models for all categories and when classifying a subtitle we would run the data through all of the models. The top three results are our labels for the data. When the maximum assignment of labels is changed we observe the following:

Number of labels	Accuracy for the number
1	80%
2	90%
3	95%

TABLE 9: Accuracy of getting at least one right with corresponding number of labels

When we analyse the results we can see that the first label is already on its own quite effective while additional labels fill in half of the remaining accuracy. This shows that the model also strong in second guesses. This can be a very desirable feature if the model knows that the first classification is not very likely. The reason why the final model is not multilabel is due to compatibility problems with other models. The full text classification can be implemented as multilabel with minimal change but the other parts of the final model are not easily compatible. More detail can be found in the model combination logic section.

5. RESULT

Our most accurate model was the full text classification model. We have applied multiclass SVM classification with and without multilabel support. Single label model also combined with other models to analyse multimodel classification. Multilabel results are evaluated in next sections. Below you can observe the single-label SVM classification on full subtitle texts.

	precision	recall	f1-score	support
Action	0.72	0.64	0.68	89
Comedy	0.70	0.70	0.70	89
Crime	0.74	0.73	0.73	89
Horror	0.87	0.85	0.86	89
Musical	0.85	0.90	0.87	89
Romance	0.77	0.75	0.76	89
War	0.86	0.93	0.89	89
Western	0.93	0.97	0.95	88
avg / total	0.81	0.81	0.81	711

TABLE 10: Full text classification with single-label. Accuracy: 0.79

There are two main results which combined different models in order to increase accuracy.There are not significant differences between 2 model because applying bag of words on full text of subtitles is dominating on other models so final results are close to result of bag of words on full text.

5.1. Sum of f1-scores

In first combining method, for each model we find it's predicted label and update our f1's sums with f1 score of that model on that category. Finally we find the label which has the maximum f1 sum of all models and

return that label as final predicted label for particular subtitle.

	precision	recall	f1-score	support
Action	0.66	0.52	0.58	102
Comedy	0.72	0.57	0.64	08
Crime	0.61	0.67	0.64	93
Horror	0.76	0.91	0.83	109
Musical	0.78	0.92	0.84	87
Romance	0.74	0.62	0.67	115
War	0.84	0.86	0.85	109
Western	0.83	0.98	0.90	85
avg / total	0.74	0.75	0.74	808

TABLE 11: Combined Model (Sum of F1 scores. accuracy: 0.747524752475

5.2. Max f1-scores

In the second combining model, for each model we predict the label and take f1 scores on these label categories. We compare all method's f1 scores on their predicted label categories and we select maximum f1 score of predictions and return its label.

	precision	recall	f1-score	support
Action	0.67	0.58	0.62	102
Comedy	0.73	0.62	0.67	108
Crime	0.63	0.69	0.66	93
Horror	0.80	0.90	0.84	109
Musical	0.80	0.90	0.84	87
Romance	0.72	0.66	0.69	115
War	0.85	0.86	0.85	109
Western	0.92	0.96	0.94	85
avg / total	0.76	0.76	0.76	808

TABLE 12: Combined Model (max f1 score). accuracy: 0.764851485149

6. DISCUSSION

We combined 3 models, which are kNN on dpm and wpm values, bag of words on Impaired Data and finally the model of bag of words on full text of subtitle, in order to improve our predictions. While combining these models we used 2 different method, one of them is choosing prediction with maximum f1 score. Other one is choosing label which has maximum sum of s1 scores from models. However these two method do not improve our prediction results effectively because one model, which is the model of applying bag of words on full text of subtitles, is dominating result. The reason of that, this method has maximum f1 scores on every category so while we are choosing the label with maximum f1 score, we are choosing that method's f1 score and accuracy do not changed well. We try to use other models as supporting models, it means that improve the accuracy of third model, which is using bag of words on full text. We try to do that with using second first combining model which is looking sum of f1 scores however, results of that method is not well to improve third model.

Overall, we can see that the combination of model did not contribute to the accuracy of our system. In fact, when we look at the results section there is a drop in performance. This drop, however, is not significant thus we did not add go back to the single model system. It is acceptable to revert to a single model system since the accuracy is better. But for the sake of this report and our efforts we decided not to. Also, we tried to combine models not the features into a single model. Since normal procedure is to combine features with a weight into a model, we tried more intuitional and experimental method. As conclusion, we have experimented that combining multiple methods by their f1 score does not give better results.

One of the problems we faced was with bag of words implementation. At first we were not aware that the test and train should not be mixed for the word vocabulary construction and the implementation was this way. The mistake was recognised during a bug fix. Constructing the vocabulary of the bag was made with the train set and the test only used the bag to vectorise without adding any words. This fix resulted in a significant performance drop at first but was compensated with some fine tuning of model parameters especially for the svm models in the full text classification part.

We realised that training models on large data sets can be very slow and annoying. Trying to debug the train data format or tune parameters becomes very challenging when working this way. One approach we took was to work on smaller data sets to achieve these goals and then scale to the full data set.

Model Combination Concerns We also applied multilabel for SVM with normal text data sets but could not implement for all models. Therefore we did not add multilabel support for combined model. We kept multilabel result as a separate observation for the sake of simplicity.

7. CONCLUSION

As a conclusion, we used different methods in order to analyse categories of movies from their subtitles. We have 3 main ways of analysing the subtitles (frequency, hearing impaired descriptions, whole text) and machine learning model associated with them. In some of them, we take remarkable results and continue to improve that method. On the other hand, in some methods results are not significant and not available to improvement so we do not continue developing in these kinds of methods. knn, SVM, Naive Bayes and Bag of Words are some of the methods which we are used in the project.

APPENDIX A. WORK DIVISION

- Enes Akdoğan: Worked mostly on analysing the density of speech of the whole movie. The work described in Dialogue per Minute and Word per Minute were done by him, worked on kNN Dpm Wpm model. He also contributed to the downloading of the data. Worked with Ali Burak since they were both working on similar areas. He also worked on combining models.
- Ömer Akgül: Worked with Mesut on isolating, filtering and training sound descriptions for hearing impaired people. With slight modifications to the code written for these steps he also applied the same steps to the whole text without the sound descriptions. This showed slightly better results but we believe machine learning on the whole text will have a much better chance of correct classification. Preprocessing of the whole text was done by him. After the progress report he worked on the improvement of the whole text model. He experimented with word2vec with Mesut Gürlek and implemented the multilabel version of the whole text model. One vs rest SVM was implemented as a part of the multilabel model. Some fine tuning was done by adjusting parameters. The model accuracy increased significantly with the increased data and new methods.
- Ali Burak Erdoğan: Worked on collecting useful subtitle data. Wrote the main script which automates the process of collecting most popular and distinguishable movie information of several categories from IMDb web site, then downloading their subtitles from OpenSubtitles API. Also, worked with Enes on dialogue tension feature of subtitles which indicates the distribution of dialogue counts throughout a single movie.
- Mesut Gürlek: Was the main contributor to the parsing hearing impaired texts, implementing filtering and training to sound descriptions. He visualized the data by creating plots of word counts in each categories. Experimented with the learning parameters in order to get the best results. Also, carried out further classification analysis for MultinomialNB with Hearing descriptions. Most importantly, implemented core logic of model combination and integration of hearing description model into combined model.
- Mustafa Yıldız: Contributed for bag of words and worked on progress and final report documentation and latex typesetting.

REFERENCES

[1] Bordwell, D., Kristin, T., and Smith, J. (2016) *Film Art: An Introduction*. McGraw-Hill Higher Education, Boston.

[2] Scrapy (2016). Scrapy — a fast and powerful scraping and web crawling framework. https://scrapy.org/. [Online; accessed 26-December-2016].

[3] OpenSubtitles.org (2016). Xmlrpc - opensubtitles.org. trac.opensubtitles.org/projects/opensubtitles/wiki/XMLRPC. [Online; accessed 26-December-2016].

[4] NLTK (2016). Natural language toolkit - nltk 3.0 documentation. www.nltk.org/. [Online; accessed 26-December-2016].

[5] nadbor (2016). Text classification with word2vec. nadbordrozd.github.io/blog/2016/05/20/text-classification-with-word2vec/. [Online; accessed 26-December-2016].

[6] NLTK (2016). models.word2vec – deep learning with word2vec. radimrehurek.com/gensim/models/word2vec.html. [Online; accessed 26-December-2016].