Comparison of Machine Learning Techniques for Multi-label Genre Classification

Mathijs Pieters and Marco Wiering

Institute of Artificial Intelligence and Cognitive Engineering
University of Groningen, The Netherlands
`{m.t.pieters,m.a.wiering}@rug.nl`

Abstract. We compare classic text classification techniques with more recent machine learning techniques and introduce a novel architecture that outperforms many state-of-the-art approaches. These techniques are evaluated on a new multi-label classification task, where the task is to predict the genre of a movie based on its subtitle. We show that pre-trained word embeddings contain 'universal' features by using the Semantic-Syntactic Word Relationship test. Furthermore, we explore the effectiveness of a convolutional neural network (CNN) that can extract local features, and a long short term memory network (LSTM) that can find time-dependent relationships. By combining a CNN with an LSTM we observe a strong performance improvement. The technique that performs best is a multi-layer perceptron, with as input the bag-of-words model.

Keywords: natural language processing, multi-label text classification, movie subtitles, CNN model, LSTM network, bag-of-words model

1 Introduction

Text classification is the task of assigning specific categories to documents, examples are spam detection and sentiment analysis. Naive Bayes, a technique based on applying Bayes' Theorem, is frequently used as a baseline method for text classification because it is relatively effective, fast, and easy to implement [11]. Numerous attempts have been made to tackle the poor assumptions of Naive Bayes [8, 17].

Various types of neural networks have been developed throughout the years, many of these techniques are used for natural language processing (NLP) applications. A traditional method is the multilayer perceptron (MLP), trained on the bag-of-words (BoW) model [1]. The BoW model is a sparse representation of texts, ignoring both word order and semantic and syntactic features, treating texts as unordered sets of words. In order to capture the subtleties of language, we seek a dense representation that does capture these features. Many state-of-the-art word embedding techniques [12, 15] are based on the distributional hypothesis [3], stating that *linguistic items with similar distributions have*

similar meanings. These dense representations capture multiple degrees of similarity [14], both semantic and syntactic, such that similar words have similar representations.

Convolutional neural networks (CNN) make use of the internal structure of the dense representation, both in the feature domain, and the temporal (word order) domain. CNN models have achieved remarkable results on various text classification tasks [21, 5]. Whereas CNN models make use of the word order for a specific region size, recurrent neural networks (RNN) have the ability to capture long-term dependencies for texts of any length. More specifically, the Long Short-Term Memory (LSTM) architecture [4] is well suited for longer texts because of its ability to remember information for long periods of time.

In this paper, we introduce a novel dataset which we will use for multi-label text classification. We compare several state-of-the-art techniques, such as the concatenation-CNN and the LSTM network, with more traditional techniques. Furthermore, we introduce a novel architecture that applies a histogram on word embeddings, followed by an MLP. Unlike most research, we trained our own word embeddings, making our setup stand-alone.

In section 2 we introduce our dataset, followed by section 3 where we explain the used methods. The experimental setup is described in section 4, and in section 5 we show and discuss the results. We conclude the paper in section 6 with a conclusion and a proposal for future work.

2   Dataset

The dataset used in the experiment is an intra-lingual movie subtitle corpus, collected by [9], and originates from OpenSubtitles[1]. We extracted the English corpus, and removed all tokens apart from the spoken text. Subsequently, we convert all words to lowercase and remove punctuation, see Figure 1. We did not apply stop word removal or stemming. The total dataset consists of 44,171 subtitles, with in total 135,862,112 words and 920,705 unique words. Every subtitle is linked to at least one, and often multiple genres. In total the dataset contains subtitles with 27 different genres, ranging from animation and comedy, to documentary. Because every subtitle can have multiple genres, the classification task is considered a multi-label classification task. This should not be confused with multi-class classification, where every document has exactly one label. Multi-label classification is considered to be significantly more difficult, due to the vast amount of possible label combinations.

Because of the limited availability of computer power we will narrow our focus to the classification of the following genres: "Romance", "Thriller", and "Action". This subset consists of 15,500 subtitles, with in total 48,998,774 words and 448,101 unique words. The distribution of the subtitle lengths is depicted in Figure 2.

---

[1] http://www.opensubtitles.org/

```
<s id="1637">
    <time id="T514S" value="01:38:8,709" />
    <w id="1637.1">May</w>
    <w id="1637.2">the</w>
    <w id="1637.3">Force</w>
    <w id="1637.4">be</w>
    <w id="1637.5">with</w>
    <w id="1637.6">you</w>
    <w id="1637.7">.</w>
    <time id="T514E" value="01:38:11,018" />
</s>
```
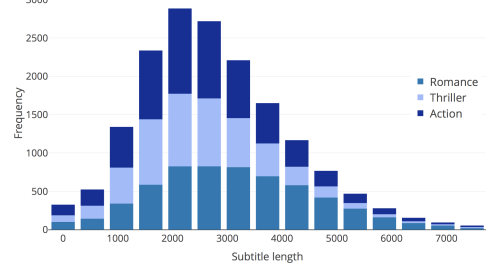
↓

may the force be with you

Fig. 1. Text preprocessing, single sentence example.

Fig. 2. Distribution of subtitle lengths

3 Methods

In this paper, we will differentiate between models that use the BoW model, and models that use word embeddings. For the first model we have two different methods, and for the latter we will discuss four methods.

3.1 Bag of Words

We will use the BoW model for both the Naive Bayes classifier and the multi-layer perceptron. Let $\boldsymbol{d} = \{d_1, ..., d_n\}$ be a collection of documents, where d_{ij} denotes the number of occurrences of word i in document j. Furthermore, let $\boldsymbol{l} = \{l_1, ..., l_n\}$ be the according labels, where l_i is itself a set of possibly multiple labels.

Multinomial Naive Bayes transformations We will focus on the Multinomial version of the Naive Bayes model (MNB), where each word position is assumed to be independent of every other word. We will use several improvements proposed by [17], e.g. normalising the weight vectors.

To train the MNB model, we apply the transformations described in equation 1. For a test document t, with word i occurring t_i times, the document is labelled according to equation 2 for some threshold θ.

$$d_{ij} \overset{(1.1)}{=} d_{ij} \log \frac{\sum_k 1}{\sum_k 1_{i \text{ occurs in } k}}$$

$$d_{ij} \overset{(1.2)}{=} \frac{d_{ij}}{\sqrt{\sum_k (d_{kj})^2}}$$

$$w_{gi} \overset{(1.3)}{=} \log \sum_{j:g \in l_j} d_{ij}$$

$$(1)$$

$$w_g \overset{(2.1)}{=} \sum_i t_i w_{gi}$$

$$w_g \overset{(2.2)}{=} \frac{w_g - \min w_g}{\max w_g - \min w_g}$$

$$l(t) \overset{(2.3)}{=} \{g : w_g > \theta\}$$

$$(2)$$

In equation 1.1, we down-weight common words, a heuristic known as "inverse document frequency". Common words have little influence on the class of a document, but small variations can cause spurious correlations. Note that in most literature a "term frequency" heuristic precedes equation 1.1, we however found that this did not improve the accuracy. Therefore as shown in equation 1.1 we just use the term frequency. In order to prevent that document length affects the classification, we normalize every document according to equation 1.2. Finally, in equation 1.3 we add the weights of all documents belonging to the same genre.

For classification we first multiply each word frequency with the weight, as illustrated in equation 2.1. In standard multi-class classification, we could now assign a label to the class with the highest score. However, since the task is multi-label classification, we have to be able to assign multiple labels to a single document. We do this by first normalizing the weights according to equation 2.2, and then assign each label for which the weight is greater than the predefined threshold θ. By increasing θ we can trade-off recall for precision (defined in section 5.1). We determine this threshold by means of the validation set.

Multi-layer Perceptron The multi-layer perceptron (MLP) has been shown to be effective on a wide variety of tasks, despite its simplicity. We use a fully connected network, with two hidden layers. We use the ReLU activation function, and in every layer we apply L_2-normalisation before activation. The input of the MLP is again the BoW model, with the n most frequent words. Every word frequency is rescaled according to $d_{ij} = \log(1 + d_{ij})$, reducing the influence of frequently occurring words.

3.2 Skip-gram Model

Many state-of-the-art techniques require dense word vectors as input. It is hypothesised that the techniques developed by e.g. [12] create dense word vectors that contain 'universal' features that can be used for various tasks. We will focus on the Skip-gram model [13]. In this model, each current word is used as an input, and the target is to predict the words that occur within a certain context c before and after the center word, as illustrated in Figure 3. Furthermore, we use Negative sampling (NGE) as objective, where the task is to distinguish the target word from k negative samples drawn from a noise distribution. Since frequent words generally provide less information, we apply subsampling to all words as described in [13]. We train the model using all subtitles in our dataset, in section 4.2 we denote the used hyperparameters. In order to explore the quality of the word vectors we use the Semantic-Syntactic Word Relationship test set, defined in [12]. This test set consists of five types of semantic questions and nine types of syntactic questions. The task is to predict a word, based on the relationship between three given words. An example for the semantic test is: "What word is similar to *Oslo* in the same way as *France* is similar to *Paris*?", the answer would be *Norway*. This test is performed by computing the vector

$\boldsymbol{x}$ = vector("france") - vector("paris") + vector("oslo"), and finding the word that has the smallest cosine distance to this vector $\boldsymbol{x}$ (different from the three question words). An answer is considered correct only if the closest word is identical to the word in the question. Table 1 shows the results on the word analogy task, indicating the effectiveness of the technique as well as generalizability of the used dataset. For the accuracy we denote both the percentage correct, and the number of correct classified pairs combined with the total number of pairs. Note that we used a subsection of the original test set, because some of the test words do not occur in our dataset. We evaluated 6,067 out of the original 8,869 semantic relations, for the syntactic relations we evaluated 10,300 out of 10,675 pairs. The results show that for the semantic relations the categories Common capital city and Man-Woman are learned very accurately, whereas Currency scores poorly. We expect that this is a result of the nature of movie subtitles, relationships (Man-Woman) and famous locations (Common capital city) play an important part in many movies, in contrast to currencies. The syntactic relations show a more balanced result, probably because all nine syntactic categories occur in spoken language.

Table 1. Results of Semantic-Syntactic Word Relationship test set.

| Category	Accuracy
Semantic:	43.9 % (2665/6067)
Common capital city	86.6 % (433/506)
All capital cities	43.1 % (996/2310)
Currency	7.40 % (37/502)
City-in-state	35.4 % (824/2328)
Man-Woman	89.3 % (375/420)
Syntactic:	61.8 % (6362/10300)
Adjective to adverb	31.1 % (271/870)
Opposite	25.6 % (180/702)
Comparative	81.6 % (1087/1332)
Superlative	64.4 % (723/1122)
Present participle	62.7 % (622/992)
Nationality adjective	68.6 % (1044/1521)
Past tense	61.3 % (957/1560)
Plural nouns	82.1 % (1093/1332)
Plural verbs	44.3 % (385/869)
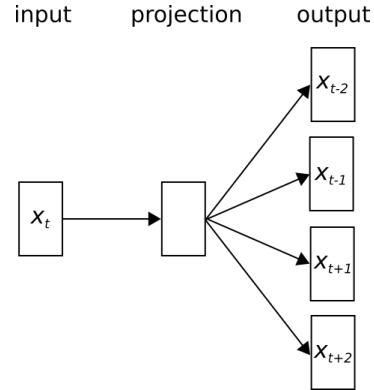
Fig. 3. The Skip-gram architecture, with a context size c of 2.

3.3 MLP on Histogram of Word Embeddings

Previous research has shown that first training a part of the model on an unsupervised task can reduce the training time and increase the accuracy on the supervised task [16]. Because the pre-trained word embeddings contain various features, we expect that a basic model can find relationships between several features in order to learn a supervised task. Preliminary experiments have shown

that taking scalar indicators (such as min, max, or mean) of a single feature over all words in combination with an MLP does not lead to satisfying results. Both the min and max operators can be affected by single, meaningless outliers, whereas the mean operator can potentially reduce significant positive and negative weights to a meaningless average. In order to capture more information we propose to use a histogram, where each word-embedding feature is described by a certain number of bins. Every bin denotes the relative frequency of a range of values for that specific feature. We will now describe the method used to convert a document to a word-embedding histogram, that subsequently can be used as an input for an MLP. Let every subtitle be consisting of n words, such that

$$X = \boldsymbol{x}_1 \oplus \boldsymbol{x}_2 \oplus ... \oplus \boldsymbol{x}_n \tag{3}$$

where $\boldsymbol{x}_t \in \mathbb{R}^k$ is the k-dimensional word embedding and $\oplus$ is the concatenation operator. Note that in most literature word embeddings are referred to by "words", we will use "concepts" because the word embeddings are actually the representation of the concept of a word, and not the word itself. The concatenation of the word embeddings results in a matrix $X \in \mathbb{R}^{n \times k}$, where n and k denote the number of words in the subtitle and dimension of the word embedding respectively. In order to use this matrix in combination with a histogram, we first need to scale the values such that we can use bins with a prefixed size and range. We normalize the matrix X according to

$$X_{ij} = \frac{X_{ij} - \min_i X_{ij}}{\max_i X_{ij} - \min_i X_{ij}} \tag{4}$$

We will now make a histogram along every word dimension, using s bins, where every bin has a width of size $1/s$. The range of the bins are denoted by $\{b_1 = [0, \frac{1}{s}), b_2 = [\frac{1}{s}, \frac{2}{s}), ..., b_s = [\frac{s-1}{s}, 1]\}$. For every bin b_l and every word dimension k we now calculate

$$H_{lk} = \text{card}(\{X_{jk} : X_{jk} \in b_l\}) \tag{5}$$

Subsequently, we calculate the L_1 norm

$$H_{lk} = \frac{H_{lk}}{\sum_{i=1}^{s} H_{ik}} \tag{6}$$

and calculate the z-score

$$Z_{lk} = \frac{H_{lk} - \mu}{\sigma} \tag{7}$$

where μ, and σ are the mean and standard deviation of all values in H respectively. The resulting matrix $Z \in \mathbb{R}^{s \times k}$ is then used as an input for an MLP.

3.4 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a feedforward neural network, originally used for image classification [7]. CNN models have shown to be effective

on various NLP tasks, by utilising local features of the word embeddings [6]. We will now describe the CNN architecture. Let every document of length n be described by a sequence as defined in equation 3 (padded if necessary). Let $x_{i:i+j}$ denote the concatenation of concept x_i up to x_{i+j}. The convolutional filter $\boldsymbol{w} \in \mathbb{R}^{h \times k}$ is applied to a window of h concepts, which produces a new feature c_i. Note that k denotes again the word embedding size. We could in theory slide the convolution along the word-features too, there is however no reason to assume that any specific local relationships exist between concepts. The window of concepts $\boldsymbol{x}_{i:i+h-1}$ generates a new feature by

$$c_i = f(\boldsymbol{w} \circ \boldsymbol{x}_{i:i+h-1} + b) \tag{8}$$

where $\circ$ is the element-wise multiplication, $b \in \mathbb{R}$ is a bias term, and f is a non-linear function such as the sigmoid, hyperbolic tangent, etc. By applying this filter to all possible windows, we obtain a feature map $\boldsymbol{c} = [c_1, c_2, ..., c_{n-h+1}]$. Note that we can use multiple filters, with possibly different filter widths.

In order to capture the more significant events, we subsequently apply a max pooling operation on the feature map $\boldsymbol{c}$. Throughout the paper we will differentiate between two types of max pooling, namely max-over-time pooling and 1-D max pooling.

Max-over-time pooling The first technique extracts a single (maximum) scalar from each feature map. By using multiple convolutional filters, with varying filter widths, we obtain several features which are then passed on to a fully connected layer. This architecture was introduced by [6], and is referred to as concatenation-CNN (C-CNN). Whereas the architecture introduced by [6] uses a final softmax layer, we adapt the network for a multi-label problem by using a sigmoid activation output layer.

1-D max pooling Max-over-time pooling reduces a feature map to a single feature, we can also reduce the feature map to several features, for different windows. In order to determine the maximum value for a window of size m we define

$$p_i = \max(c_{i:i+m-1}) \tag{9}$$

with $i = (1, 1+s, 1+2s, ...)$, where s denotes the size of the stride. In both the convolutional layer and the 1-D max pooling layer we can vary the stride, meaning that instead of moving the filter one step at the time, we move the filter several places per step. We use multiple filters for the same region, making it possible to learn complementary features from the same regions. With l filters, the generated l feature maps are combined to create a matrix $X \in \mathbb{R}^{l \times \lfloor (n-h-m+2)/s \rfloor}$. These feature maps are then used in combination with an LSTM network, as explained in section 3.6.

3.5 Long Short-Term Memory Network

A Recurrent Neural Network (RNN) has the ability to capture time-dependent relations between words. It does this dynamically, without the use of fixed-size context windows. In particular, the Long Short-Term Memory Network (LSTM) [4] excels at tasks where long term dependencies are important. This network has received a lot of attention because of its capability of capturing important events throughout time series, and being relatively unsusceptible of gaps between important events. Given a sequence as described by equation 3, at time step t the LSTM network updates c_t and h_t with input x_t as follows

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \text{tanh} \end{bmatrix} W \cdot \begin{bmatrix} h_{t-1}, x_t \end{bmatrix} \tag{10}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \hat{c}_t \tag{11}$$

$$h_t = o_t \circ \tanh(c_t) \tag{12}$$

where c_t and h_t are the memory and hidden state respectively, i_t, f_t, o_t, and $\hat{c}_t$ are the input gate vector, forget gate vector, output gate vector, and current cell state vector respectively. Note that in equations 10 and 12 the functions sigm and tanh are applied element-wise. In order to map the output of the LSTM network to the output layer, we apply mean-over-time pooling on the output gate vectors o_t, meaning that we calculate the mean of all h_t values over all time steps t. Finally, the mean-over-time pooling is followed by a fully-connected layer with a sigmoid activation function.

The traditional LSTM network may have problems when the change of the parameters of one layer has an effect on the distribution of the input to all subsequent layers, also known as internal covariance shift. A solution proposed by [2], called Batch Normalized LSTM (BN-LSTM), normalizes both the input-to-hidden and hidden-to-hidden transformations by empirically estimating their means and standard deviations.

3.6 CNN-BN-LSTM

We discussed that CNN leverages the local features of words, whereas LSTM dominates in tasks where long term relations play a part. By combining the two techniques, we hope to get the best of both worlds. We start with applying a CNN layer, followed by a 1-D max pooling layer, as discussed in section 3.4. The resulting matrix is then used as input for the LSTM network, such that there are $\lfloor (n - h - m + 2)/s \rfloor$ time steps, each with dimension l. Similar to the procedure discussed in section 3.5, we subsequently apply mean-over-time pooling on the output gate vectors, together with a fully-connected layer with sigmoid activation. An example of this network is shown in figure 4.
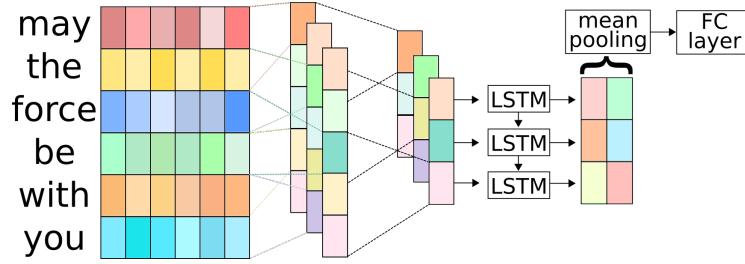
Fig. 4. Graphical representation of the CNN-BN-LSTM network. The hyperparameters of this example are as follows. The word-embedding size is 6. The convolutional layer uses a window size of 2, with a stride of 1, this is followed by 1-D max-pooling with a window size of 3, and a stride of 1.

4 Experimental Setup

4.1 Dataset

The proposed models are tested on the dataset introduced in section 2. We split the dataset into a validation set and a train-test set of respectively 1500 and 14,000 subtitles, so that we tune the hyperparameters on the validation set and use cross validation on the train-test set. We use 7-fold cross validation in order to test the methods, the train set consists each time of 12,000 movies, the test set of 2,000 movies.

4.2 Hyperparameters and Training

The following hyperparameters are all determined by performing a grid search on the validation set. For the MNB model we only take into account words that occur more than 3 times. We use a classification threshold θ of 0.7 for the MNB model. For all other models we use a threshold value of 0.5.

For the BoW-MLP model we use the 50,000 most frequent words. The first hidden layer contains 512 nodes, the second layer 256. In both layers we apply the ReLU activation function, followed by dropout [19] with a dropout rate of 0.5.

Throughout all experiments we use a word embedding size of 300. We use static word embeddings, we thus apply no back propagation on the word embeddings in any of the experiments. We trained the word-embeddings on all subtitles, thus not only on the used subset for the multi-label classification task. The training was performed for 12 epochs, using a learning rate of 0.1, a mini-batch size of 16, a subsample threshold of 10^{-3}, a context size c of 5, and with 15 negative samples.

For the MLP-Histogram model we use 25 bins, followed by 128 hidden nodes in the first layer of the MLP, and 64 nodes in the second layer. Furthermore, in order to prevent overfitting we add Gaussian noise to the input with a mean

of 0, and a standard deviation of 0.02. Additionally, after each layer dropout is applied with a rate of 0.5. Finally, in each layer the ReLU activation function is applied.

The BN-LSTM model uses 300 hidden units, on both the input and output connections we use dropout with a rate of 0.2. We constrain the L_2-norm of the gradient to not exceed 10, this is known as gradient clipping.

For the CNN-BN-LSTM network we use similar LSTM hyperparameters, proceeded by a CNN. The CNN consists of 200 feature maps, with a window size of 8, a filter stride of 2, followed by a 1-D max pool filter of size 4, with a stride of 2. The activation function used in the CNN is the ReLU. Again we constrain the L_2-norm of the gradient to a maximum of 10.

In the C-CNN model we use filters of width 3,4 and 5, all with 128 feature maps. We apply dropout with a rate of 0.5, and constrain the L_2 norm again to 10.

For the CNN-BN-LSTM and the C-CNN model we pad the documents to a maximum length of 4000 words. In all models we use a mini-batch size of 20. We train the MLP-BoW and C-CNN for 6 epochs, all other models are trained for 10 epochs. We used the Adadelta update rule [20] for training, while shuffling the mini-batches.

Throughout all experiments (apart from training the word-embeddings) we anneal the learning rate α using exponential decay, defined by $\alpha = \alpha_0 r^{t/k}$, where α_0 is the initial learning rate, r is the decay rate, t is the iteration step, and k indicates the decay step, such that every k steps the learning rate is decayed. In all experiments we use a decay rate r of 0.97. For the MLP-Histogram, BN-LSTM, and CNN-BN-LSTM we used an initial learning rate of 0.1, for the MLP-BoW and C-CNN we use an initial learning rate of 0.005.

5   Results and Discussion

5.1   Metrics

In order to compare our models we will use the F_1 score, which takes into account both the recall and precision. Recall, precision, and the F_1 score for one label are respectively defined as:

$$\text{recall} = \frac{|\{\text{relevant labels}\} \cap \{\text{retrieved labels}\}|}{|\{\text{relevant labels}\}|} \qquad (13)$$

$$\text{precision} = \frac{|\{\text{relevant labels}\} \cap \{\text{retrieved labels}\}|}{|\{\text{retrieved labels}\}|} \qquad (14)$$

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \qquad (15)$$

In order to calculate the final recall, precision, and F_1-score of the models, we calculate the mean scores over all three genres.

5.2 Results

The results of our models are listed in Table 2. Our baseline method (MNB) does not perform well. The model that performs best is the MLP-BoW model, with an average F_1-score of 0.77 ± 0.02. This is a significant higher result ($P < 0.005$) compared to the other models. The novel MLP-Histogram model achieves the second highest F_1-score. The BN-LSTM does not perform well on it own, however, in combination with a CNN layer (CNN-BN-LSTM) the model obtains the third best results. Finally, the C-CNN model is outperformed by all but two models.

Model	Romance			Thriller		
	Recall	Precision	F_1-score	Recall	Precision	F_1-score
MNB	0.93 ± 0.08	0.49 ± 0.14	0.64 ± 0.10	0.02 ± 0.01	0.67 ± 0.14	0.04 ± 0.03
MLP-BoW	0.75 ± 0.13	0.77 ± 0.04	0.76 ± 0.09	0.72 ± 0.01	0.77 ± 0.01	0.74 ± 0.08
MLP-Histogram	0.61 ± 0.03	0.72 ± 0.04	0.66 ± 0.02	0.78 ± 0.04	0.77 ± 0.02	0.77 ± 0.02
BN-LSTM	0.17 ± 0.13	0.72 ± 0.10	0.25 ± 0.14	0.76 ± 0.08	0.74 ± 0.05	0.75 ± 0.02
C-CNN	0.52 ± 0.17	0.69 ± 0.09	0.56 ± 0.10	0.77 ± 0.05	0.76 ± 0.09	0.76 ± 0.05
CNN-BN-LSTM	0.54 ± 0.05	0.74 ± 0.03	0.62 ± 0.04	0.76 ± 0.04	0.79 ± 0.02	0.77 ± 0.03

Model	Action			Mean		
	Recall	Precision	F_1-score	Recall	Precision	F_1-score
MNB	0.83 ± 0.11	0.71 ± 0.15	0.73 ± 0.08	0.59 ± 0.02	0.62 ± 0.05	0.47 ± 0.04
MLP-BoW	0.81 ± 0.06	0.82 ± 0.06	0.81 ± 0.05	0.76 ± 0.03	0.79 ± 0.03	$\mathbf{0.77 \pm 0.02}$
MLP-Histogram	0.80 ± 0.03	0.79 ± 0.06	0.79 ± 0.02	0.73 ± 0.01	0.76 ± 0.01	0.74 ± 0.01
BN-LSTM	0.75 ± 0.05	0.80 ± 0.06	0.77 ± 0.01	0.56 ± 0.04	0.75 ± 0.04	0.59 ± 0.04
C-CNN	0.75 ± 0.07	0.80 ± 0.08	0.77 ± 0.03	0.68 ± 0.06	0.75 ± 0.02	0.70 ± 0.04
CNN-BN-LSTM	0.78 ± 0.01	0.83 ± 0.05	0.80 ± 0.03	0.69 ± 0.03	0.78 ± 0.03	0.73 ± 0.03

Table 2. The results on the test set, after the specified number of epochs. Both the mean and standard deviation of the cross validation are displayed. We denote the recall, precision, and F_1-score for the three genres, together with the mean of the recall, precision, and F_1-score of the three genres.

5.3 Discussion

Our baseline model (MNB) does not perform well, the genre thriller has a very low recall and therefore a low F_1-score. The other two genres have however a very high recall (higher than all other models). We expect that the poor results on the genre thriller are caused by a combination of how the threshold is determined and the poor assumptions of the MNB model. We also experimented with n-grams, with n ranging from 1 to 3, but the performance decreased for n higher than 1.

The MLP-BoW model outperformed all other (more complex) models. This was in contrast with our expectations, because the model is relatively simple compared to the other machine learning models. Not only is the F_1-score high, the training time was also relatively short. The fact that this model achieves the highest F_1-score could suggest that there exist some combinations of important 'indicator words' that are strong predictors for certain genres. We experimented with adding more layers to the network, but this had no significant positive effect on the results. Removing one layer had a negative effect on the accuracy.

Considering that the MLP-Histogram model only takes into account the relative frequency of word embedding feature values the model performs remarkably well. This is another illustration of the 'universal' features of word embeddings. Similar to the MLP-BoW model, the training time is relatively short. Furthermore, this newly proposed model performed best with using the word-embeddings.

The BN-LSTM model performs rather poorly. We expect that this is due to the length of the documents. A careful observation of Figure 2 shows that the genre romance has relatively long subtitles. This could explain the poor results on this genre for models that are susceptible for document length. Although the BN-LSTM network does suffer less from vanishing gradients compared to other RNN networks, the network still has problems with documents of substantial length. Another explanation for the inadequacy of the BN-LSTM model could be that for this task word order is irrelevant and only the occurrence of certain words is important. Preliminary experiments have shown that stacking multiple BN-LSTM layers on top of each other had no effect on the final accuracy. The accuracy increases drastically with the use of batch normalization. Adding batch normalization also causes faster, more stable convergence. Furthermore, the model often diverged without the use of gradient clipping.

Contrary to the MNB model, we saw that for the C-CNN model the use of n-grams (by means of the filter widths) did increase the performance. Although the similar model introduced in [6] achieves state-of-the-art results on various tasks with a similar model, we find only moderate results on our task. One main difference is that the documents in the datasets used in [6] are significantly shorter compared to our dataset, making them less susceptible for outliers that can affect the max-over-time pooling.

By combining a CNN model with a BN-LSTM model (into the CNN-BN-LSTM model) we see a performance improvement compared to a separate C-CNN or BN-LSTM model. By combining the two methods we get the powerful feature extractor of the CNN model, and the capability of detecting long term dependencies of the LSTM model. The downside of this method is that even more hyperparameters have to be tuned. Exploratory research indicated that adding a CNN layer after the BN-LSTM or CNN-BN-LSTM model did not improve the accuracy.

6 Conclusion and Future Work

In this paper we described various techniques that can be used for multi-label classification of movie genres based on subtitles. First, we established a baseline using a multinomial naive Bayes (MNB) classifier combined with several heuristics that "tackle the poor assumptions of MNB" [17]. We trained word embeddings on an unsupervised task, and showed that these embeddings contain indicative features for genre classification. We developed a novel architecture that combines a histogram of the word embeddings with an MLP. Despite the simple nature of this model it outperforms several more complex models. Both the C-CNN network and the BN-LSTM perform poorly on their own. However, by combining both techniques we observe a drastic increase in performance. The model that performs best is the MLP-BoW model, a surprising result given that many papers consider this network to be a baseline method.

We observed that simple models sometimes outperform more complex, state-of-the-art networks. The best network thus completely depends on the problem at hand. Therefore we would like to stress that exploring simpler text-classification methods is of great importance when a new dataset is studied. This directly relates to the principle of Occam's razor, stating that of all possible hypotheses, the one with the fewest assumptions should be used. When we decide to use a specific technique, we make certain assumptions about the data. A simple technique is less prone to overfit the data compared to a more complex technique, because it makes less assumptions about the data. With more assumptions, it is easier to choose parameters such that they only fit the observed data, and do not generalise well.

In follow-up work we would like to consider non-static word embeddings. In [6] it is shown that for certain tasks the performance improves when either non-static word embeddings, or a combination of both static and non-static word embeddings are used. Moreover, we would like to explore the use of random word embeddings and word embeddings trained by others, e.g. [13]. The final F_1-scores could be improved by using more advanced threshold techniques, and in future research the number of genres should be extended (up to 27). Finally, experiments on more datasets can be conducted, e.g. the Movie Review Sentiment dataset [10] and the Stanford Sentiment Treebank [18].

References

1. James Clark, Irena Koprinska, and Josiah Poon. A neural network based approach to automated e-mail classification. In *Web Intelligence, 2003. Proceedings. IEEE/WIC International Conference on*, pages 702–705, 2003.
2. Tim Cooijmans, Nicolas Ballas, César Laurent, and Aaron C. Courville. Recurrent batch normalization. *CoRR*, abs/1603.09025, 2016.
3. Zellig Harris. Distributional structure. *Word*, 10(23):146–162, 1954.
4. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

5. Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*, 2014.

6. Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

7. Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

8. David D Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *European conference on machine learning*, pages 4–15. Springer, 1998.

9. Pierre Lison and Jörg Tiedemann. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*, 2016.

10. Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, 2011.

11. Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48, 1998.

12. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

13. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

14. Tomas Mikolov, Scott Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013.

15. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

16. Alec Radford, Rafal Józefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. *CoRR*, abs/1704.01444, 2017.

17. Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, and David R. Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 616–623, 2003.

18. Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

19. Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

20. Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

21. Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.