Classification of News Dataset

Olga Fuks
Stanford University
ofuks@stanford.edu

1 Introduction and motivation

Nowadays on the Internet there are a lot of sources that generate immense amounts of daily news. In addition, the demand for information by users has been growing continuously, so it is crucial that the news is classified to allow users to access the information of interest quickly and effectively. This way, the machine learning model for automated news classification could be used to identify topics of untracked news and/or make individual suggestions based on the user's prior interests. Thus, our aim is to build models that take as input news headline and short description and output news category.

2 Data and features

2.1 Dataset

Our data source is a Kaggle dataset [1] that contains almost 125,000 news from the past 5 years obtained from HuffPost [2]. News in these dataset belong to 31 different topics (labels). Each news record consists of several attributes from which we are using only 'Category', 'Headline' and 'Short description' in our analysis. In addition, we combine data attributes 'Headline' and 'Short description' into the single attribute 'Text' as the input data for classification.

The data preprocessing consisted in combining some raw data categories that are very close (for example, "Arts" and "Arts and Culture", "Education" and "College" etc). The Fig. 1 show an analysis of the data statistics – number of samples per category and average number of words per combined news description. From the Fig. 1a it is obvious that we are dealing with imbalanced categories – first three most well represented categories, "Politics", "Entertainment" and "World News", if combined, make up around 44% of all data samples. However, from Fig. 1b we see that in terms of number of words per news description categories are much more homogeneous. Overall average is 25.6 ± 14.4 words. A sample description from "Entertainment" category is shown below:

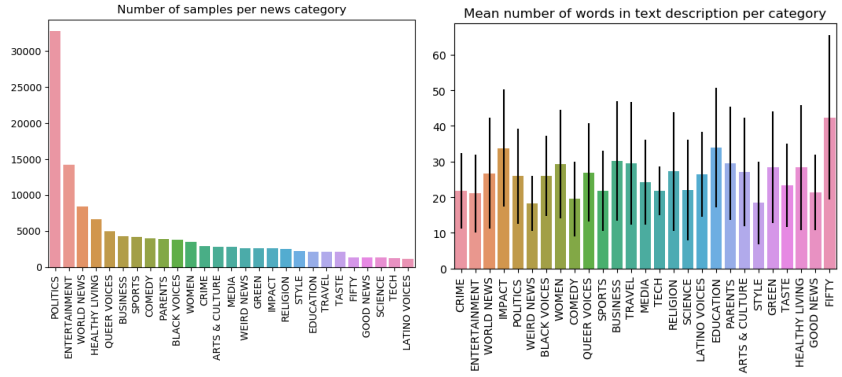
Hugh Grant Marries For The First Time At Age 57. The actor and his longtime girlfriend Anna Eberstein tied the knot in a civil ceremony

In the following work we decided to only consider samples with description's size greater than 7 words. Moreover, categories "Comedy" and "Weird news" were removed from the consideration. All this preprocessing left us with total number of samples 113,342 and 25 news labels. Last step of preprocessing included removal of stop words as well as punctuation and finally, stemming of each word.

2.2 Features

First, using the preprocessed news descriptions we created the dictionary of words. The total number of unique words is around 40,000. Then, we extracted the following word features for classification task:

- **Word binary and word count features:** For binary and count features we used first 5,000 most common words to define the dictionary and then, encoded the news descriptions as vectors - either as vectors of 0 and 1 for binary features or of word counts in the description.



(a) (b)

Figure 1: Statistical analysis of dataset: (a) Number of samples per category (b) Average number of words in the combined news description

- **Word level TF-IDF scores:** For TF-IDF method we decided to extend the dictionary to the first 10,250 most frequent words. Moreover, we combined the text from all the news belonging to that category and treated it as the one document. Thus, our corpus of documents consisted of 25 documents (one for each news category) from which we learn TF-IDF representation and then, we apply it both to train and dev set samples.

- **Word embeddings:** Word embeddings are a family of NLP techniques aiming at mapping the semantic meaning into a geometric space [3]. To learn the word embeddings from the data we applied an Embedding layer of Keras [4]. Also, we considered only 30,000 most common words in the dataset and we truncated each example to a maximum length of 50 words.

3 Supervised Learning

3.1 Algorithms

In the first part of our work we experimented with traditional machine learning techniques: Naive Bayes, multinomial logistic regression, kernel SVM and Random Forest.

Naive Bayes With binary features we applied multivariate Bernoulli model and with count features - multinomial event model. For each example, we classify as $\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i|y)$, where we use MAP estimation for $P(y)$ and $P(x_i|y)$ while also applying Laplace smoothing [5].

Multinomial Logistic Regression We use the cross-entropy loss with L2 regularization [6]. The regularized cost function is $J(\theta) = -\sum_{i=1}^{m} \sum_{k=1}^{K} y_k^{(i)} \log \hat{y}_k^{(i)} + \lambda \sum_{l=1}^{n} ||\theta_l||_2^2$

Kernel SVM We use a multi-class SVM [7] with a "one-vs-rest" approach and an RBF kernel $K(x, z) = \exp\left(-\gamma||x - z||^2\right)$. Optimal parameter C and kernel parameter γ were optimized by 3-fold cross-validated grid-search over a parameter grid.

Random Forest We used the Gini measure $G(X_m) = \sum_k p_{mk}(1 - p_{mk})$, where p_{mk} is the proportion of class k samples in node m [8]. We regularized each tree in terms of maximum depth.

In the second part of our work, we focused on building the neural network models: with word embedding features provided by the Embedding layer of Keras we trained several neural network models with one or two convolutional layers (CNN) and/or recurrent (LSTM) layer (RNN [9]).

CNN This a class of deep, feed-forward artificial neural networks that excel at learning the spatial structure in the input data by learning the set of filters applied to the data.

RNN This is a class of artificial neural network where connections between nodes form a directed graph along a sequence. This allows it to exhibit temporal dynamic behavior for a time sequence.

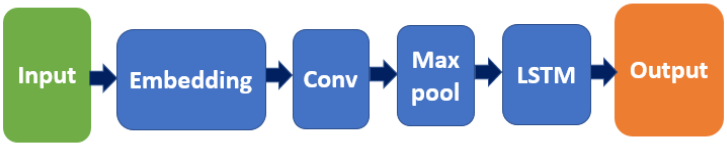


Figure 2: Typical neural network model architecture

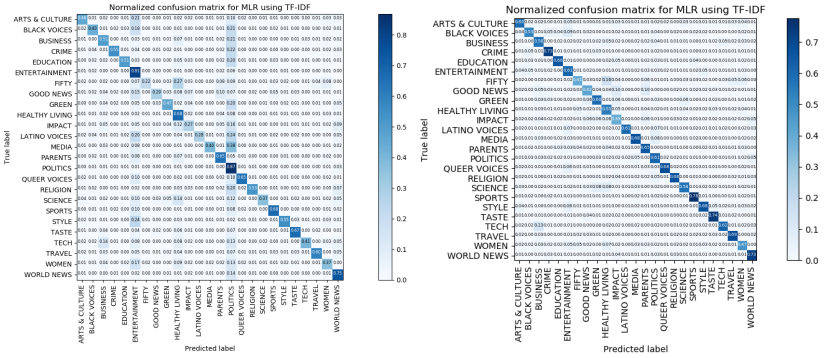
	Binary features		Count features		TF-IDF features	
	Train	**Dev**	**Train**	**Dev**	**Train**	**Dev**
Naive Bayes	0.666	0.611	0.678	0.619	0.601	0.560
Logistic Regression	0.742	**0.641**	0.747	**0.637**	0.777	**0.671**
Kernel SVM	0.996	0.609	0.975	0.611	N/A	N/A
Random Forest	0.999	0.587	0.999	0.584	N/A	N/A

Table 1: Model performance measured by classification accuracy

For our implementation, we experimented with several architectures (number of convolutional layers, number of filters in each layer, number of units in recurrent layer, dropout rate) as well as with different parameters such as an embedding dimension, maximum sequence length and maximum number of words (for words tokenization). Fig. 2 shows the typical model architecture. In addition, we tried applying pretrained GloVe embeddings [10] (with frozen Embedding layer) but the accuracy in this case was lower then when learning embeddings from the data.

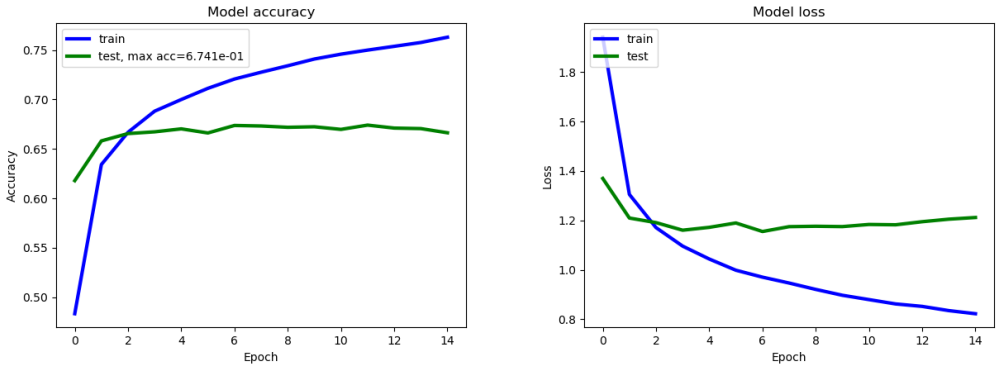
3.2 Results and Discussion

We divided the data into train/dev/test split according to 80/10/10. Table 1 shows the obtained classification accuracy across various models and features for traditional machine learning methods. For all set of features the highest accuracy is achieved by the logistic regression. Confusion matrix for logistic regression with TF-IDF features in Fig. 3a illustrates also our motivation for considering weighted logistic regression. It is obvious, that without weighting the model is biased towards predicting the more common classes, i.e. Politics, Entertainment and Healthy Living. By weighting each example by the inverse frequency of its class, we get a generally darker diagonal in confusion matrix (Fig. 3b). However, in this case the overall accuracy on dev set decreases from 0.671 to 0.622.



(a) Without example weighting (b) With example weighting

Figure 3: Confusion matrix for logistic regression with TF-IDF features



(a) Model accuracy (b) Model loss

Figure 4: Typical model accuracy and loss curves (train and dev) for neural network models

Next, we computed TF-IDF scores to select representative words for each category (these were the words with maximum TF-IDF score greater than a certain threshold). After manual inspection of the obtained words we found that they all corresponded semantically quite well to the news label.

Lastly, we trained several neural network models. For all models we observed that they quickly start to overfit the data - Fig. 4 shows the typical model accuracy and loss as functions of number of epochs. Obtained classification accuracy across various models is shown in Table 2. The architecture with additional RNN layer slightly outperforms the one with just convolutional layers. Also, the accuracy of the ensemble of four models is higher than accuracy of any individual model. However, surprisingly the accuracy on the dev dataset achieved by these models was about the same as that of the logistic regression classifier (see Tables 1 and 2). We examined the errors made by several top performing models and found that the model often confuses the true label with the label of one of the most frequent classes such as "Politics", "Entertainment", "Healthy living" and "World news" (these four categories make up 53% of the entire data set). Moreover, we realized that besides class imbalance there are at least two more factors that prevent our models from achieving higher accuracy:

- Often there is some combination of categories present in one news, though it has just one "true" label in the dataset. Example 1: *"Australian Senator Becomes First To Breastfeed On Parliament Floor "We need more women and parents in Parliament," said Larissa Waters."* - here the true category "Parents" was confused by the models with "World news", probably as it mentions Australia and senator but the news is also about parenthood. Example 2: *"Most U.S. Troops Kicked Out For Misconduct Had Mental Illness. The new report will likely add to scrutiny over whether the military is doing enough to care for troops with mental health issues"* - this news belongs to the "Healthy Living" category whereas the models identify it as "Politics" likely because the news mention US troops and military, however it is mainly about health issues.

- Overlap of different categories - we believe this may be due to the subjective assignment of the category upon news publication. Example: *"How Do Scientists Study Dreams? Dreams are a compelling area of research for scientists, in part because there's still so much to learn about how, and why, we dream. "* - this news belongs for some reason to "Healthy Living" category, though it mentions a lot about scientific research, so there is no surprise that all models identify it as category "Science".

Thus, often the model is able to understand some topic of the news but not may be the main one - sometimes the true topic is more subtle or even implicit, but there are some words in the news that are characteristic of other categories and as a result the model classifies it incorrectly. These observations motivated us to compare top three labels predicted by each model to the true label of the example (these results also shown in Table 2). In this case the maximum accuracy was 88.72% on the dev set and it is achieved by the ensemble of four NN models.

	Train	Dev		Test	
		top1	top3	top1	top3
CNN 2	71.38	64.41	84.1	64.83	84.28
CNN1-RNN 100	81.62	66.72	87.3	66.18	86.89
CNN1-RNN 200	**84.63**	66.81	87.4	66.34	86.78
CNN2-RNN 200	79.67	66.28	86.2	66.18	86.09
Ensemble of four models	83.59	**68.85**	**88.72**	**68.38**	**88.44**

Table 2: Model performance of different neural networks measured by classification accuracy (number after "CNN" in the model's name denotes the number of convolutional layers in the model, the number following "RNN" denotes the number of units in LSTM layer of the network). "Top1" column denotes the results if considering the top one label predicted by the models, "top3" - if considering top three labels.

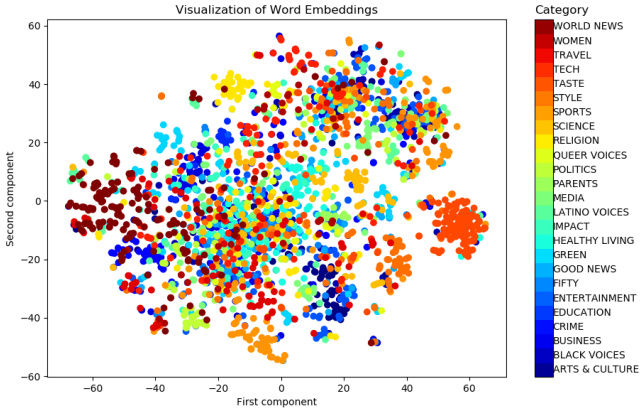


Figure 5: Visualization of word embeddings for different news categories

4 Visualization of Word Embeddings

Application of TF-IDF method allowed us to select for each news category the words that are characteristic of this category. Then, we extracted pre-trained GloVe embeddings [10] of the selected words (we used vectors of dimension 100) and applied a dimension reduction mehtod (t-SNE [11]) to visualize the word vectors in 2-D space. Fig. 5 shows the result of this procedure. Some clusters do emerge - for example, for category "Taste" (orange cluster on the right), "Sports" (light orange in the bottom), "World News" (big dark red on the left), "Religion" (small yellow at the top). In the future, this may also be employed for classification (for example, applying kNN method).

5 Conclusion

We have built a number of models to predict the category of news from its headline and short description - using methods both from traditional ML and deep learning. Our best model (ensemble of four NN models) achieves on the dev set 68.85% accuracy, if considering top 1 label, and 88.72%, if considering top 3 labels predicted by the model. It is interesting how this news dataset is extremely hard to classify for even the most complex models. We attribute this to the subjectivity in category assignment in the data. However, in the future work we may also try to apply character-level language models based on multi-layer LSTM or learn embeddings for the whole news descriptions (as in doc2vec).

References

[1] Kaggle News Category Dataset. https://www.kaggle.com/rmisra/news-category-dataset. Accessed: 2018-10-05.

[2] The Huffington Post. https://www.huffingtonpost.com/.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[4] Keras: The Python Deep Learning library. https://keras.io/.

[5] Scikit-learn 0.20.0 documentation. "1.9 Naive Bayes". https://scikit-learn.org/stable/modules/naive_bayes.html.

[6] Scikit-learn 0.20.0 documentation. "1.1.11 Logistic Regression". https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression.

[7] Scikit-learn 0.20.0 documentation. "1.4 Support Vector Machines". https://scikit-learn.org/stable/modules/svm.html.

[8] Scikit-learn 0.20.0 documentation. "1.10 Decision Trees". https://scikit-learn.org/stable/modules/tree.html.

[9] Jürgen Schmidhuber's page on recurrent neural networks. http://people.idsia.ch/~juergen/rnn.html.

[10] J. Pennington, R. Socher, and C. D. Manning. Glove: Global Vectors for Word Representation. *EMNLP*, 14:1532–1543, 2014.

[11] Scikit-learn 0.20.1 documentation, t-distributed Stochastic Neighbor Embedding. https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html.