INTRODUCTION

**Introduction**

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

Objective

**Data and features**

**1) DATA**

DATASET are contains almost 125,000 news from the past 5 years obtained from HuffPost .

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).

**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.

* + - **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.

**Context :**

This dataset contains around 150k news headlines from the year 2016 to 2021. The model trained on this dataset could be used to identify tags for untracked news articles or to identify the type of language used in different news articles.

**Content :**

Each news headline have been a corresponding category. Categories and corresponding article around counts are as follows:

1. **Politics :30000**
2. **Welness : 17200**
3. **Entertainment : 16300**
4. **Travels : 98000**
5. **Style and Beauti : 9600**

**…etc**

BAKGROUND

* **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 yˆ = arg maxy P(y) Qn i=1 P(xi |y), where we use MAP estimation for P(y) and P(xi |y) while also applying Laplace smoothing .
  + Multinomial Logistic Regression We use the cross-entropy loss with L2 regularization . The regularized cost function is J(θ) = − Pm i=1 PK k=1 y (i) k log ˆy (i) k + λ Pn l=1 ||θl ||2 2
  + Kernel SVM We use a multi-class SVM with a "one-vs-rest" approach and an RBF kernel K(x, z) = exp −γ||x − z||2 . 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(Xm) = P k pmk(1 − pmk), where pmk is the proportion of class k samples in node m . 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 featur es 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 ).
  + 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

**Modules uses:**

mpl\_toolkits

sklearn

nltk

matplotlib

numpy

os

pandas

**HARDWARE AND SOFTWARE REQUIREMENTS**

* **HARDWARE :**

LAPTOP OR DESKTOP

* **SOFTWARE REQUIREMENTS :**

SYSTEMESOFTWARE(ANY ONE) :

WINDOW 10

LINUX

MAC

APPLYCATON SOFTWARE :

PYTHON (DATA SCIENCE AND ML)

VS CODE OR PYCHARM

JUPYTER NOTEBOOK

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 AND BIBLIOGRAPHY**

* [1] The Huffington Post. <https://www.huffingtonpost.com/>.
* [2] Keras: The Python Deep Learning library. <https://keras.io/>.
* [3] Scikit-learn 0.20.0 documentation. ”1.9 Naive Bayes”. **https://scikit-learn.org/ stable/modules/naive\_bayes.html.**
* [4] Scikit-learn 0.20.0 documentation. ”1.1.11 Logistic Regression”. **https://scikit-learn. org/stable/modules/linear\_model.html#logistic-regression**.
* [5] Scikit-learn 0.20.0 documentation. ”1.4 Support Vector Machines”. **https://scikit-learn. org/stable/modules/svm.html.**
* [6] Scikit-learn 0.20.0 documentation. ”1.10 Decision Trees”. **https://scikit-learn.org/ stable/modules/tree.html.**
* [7] Jürgen Schmidhuber’s page on recurrent neural networks. **http://people.idsia.ch/ ~juergen/rnn.html**.
* [8] Scikit-learn 0.20.1 documentation, t-distributed Stochastic Neighbor Embedding. **https: //scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html.**
* **[9]** T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013
* **[10]** J. Pennington, R. Socher, and C. D. Manning. Glove: Global Vectors for Word Representation. EMNLP, 14:1532–1543, 2014.

……….etc.