Machine Learning - Final Project
Sentiment Analysis

DE BALTHASAR DE GACHEO Valentin

Mechanical Engineer
Graduate Student Dept. Computer Science and Information Engineering
National Taiwan University of Science and Technology
Taipei, Taiwan
valentindebalthasar@gmail.com

WESTHAUESSER, Fabian

Dept. of Electrical Engineering and Information Technology
University of Stuttgart and National Taiwan University of Science and Technology
Stuttgart, Germany
fwesthaeusser@gmail.com

I INTRODUCTION

NLP, short for Natural Language Processing is a field of computer science and artificial intelligence concerned with the interactions between computers and human languages. In particular how to program computers to process and analyze large amounts of natural language data. This paper presents an approach to solve a specific issue known in the field of Natural Language Processing: Sentiment Analysis

Choosing the right approach to improve our predictions' accuracy is a difficult task. Collecting more data, having a more diverse training set, training the algorithm longer, changing the size of the neural network, its parameters or even its architecture (number of hidden units, activation function, etc.) all seem to be possible options. Choosing well the directions will lead to better results, and choosing poorly might waste lot of time.

In this regard, this paper presents a simple method to tackle the Sentiment Analysis problem.

II SOLUTION

To approach this problem, we want to improve our predictions' accuracy. The method developed uses different preprocessing methods and also various models to provide predictions on unseen instances: two baseline models, and two deep learning models. Those models and preprocessing methods are then compared and analyzed.

III IMPLEMENTATION, METHOD

A. *Motivation and Scope of this Project*

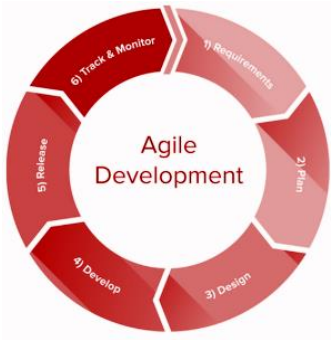

Fig. 1 Agile Development methodology

The methodology used to perform this project is based on the "Agile" methodology: collaborating to iteratively deliver improved results. Truth is, "Agile" is more of a set of principles for developing software. It emphasizes adaptability to changing situations, when there is potential for changing requirements. This continuous-cycle-based method allows to redefine short-term and long-term objectives on-the-fly, according to new ideas and requirements.

Based on literature reviews, and domain knowledge, very basic models were built in just few days to obtain some first results. Models' implementation was designed in order of complexity and relying on previous results, obtained through specific metrics analysis.

Besides, different preprocessing techniques were implemented to perform feature extraction.

Then, hyperparameter optimizations were performed to approach a better set of parameters based on a chosen reduction metric, against which optimization was performed to improve our results.

Finally, training on subsets to create a trendline gives information about whether or not the dataset yields a sufficient amount of training data.

Result analysis have been performed all along the project to assess improvements and redefine new requirements.

B. *Data*

a. *Dataset description*

The dataset contains sentences labelled with positive or negative sentiments, extracted from reviews of products, movies and restaurants form three different sources; imdb.com, amazon.com and yelp.com. For each website, there exist 500 positive and 500 negative sentences with a clear positive or negative connotation to avoid neutral sentences to be mixed with the former ones, and therefore avoid . Label is either "1" for positive reviews, or "0" for negative ones. This dataset was first created for the Paper From Group to Individual Labels using Deep Features from Kotzias et. al,. KDD 2015 [1]

b. *Data Structure*

The used data is split into three text-files (.txt) for the reviews from yelp, imdb and amazon. The sentences are separated from their corresponding binary label by a tab (\t)

Every pair of sentence and label is separated from the next pair by a line-break (\n). The data is therefore stored in the format:

"sentence" \t "label" \n

"sentence" \t "label" \n

"..."

c. *Data Split*

In our method, the three dataset sources are concatenated together and shuffled to form a bigger dataset:

- **Training set**: to run the learning algorithm.

- **Development(/validation) set**: to tune parameters, select features, and make other decisions regarding the learning algorithm.

- **Test set**: to evaluate the performance of the algorithm, but not to make any decisions regarding what learning algorithm or parameters to use.

The reason why the dataset is concatenated and shuffled in this method is because the models developed have to be able to generalize well, whatever the distribution used. Otherwise, the risk is to develop a model that provides good predictions only for one dataset source, but not for another one.

As Kaggle kernels have limitations[2], and as demonstrated later in this report, additional data are not added for this experiments.

C. Working Environment: Kaggle Kernels, a Friendly Environment for Programmers

According to Kaggle's website: "Kaggle Kernels is a cloud computational environment that enables reproducible and collaborative analysis." There are three different types of Kernels on Kaggle. The type of Kaggle Kernel used for this paper is a Jupyter notebooks which consist of a sequence of cells, where each cell is formatted in either Markdown or in a programming language (Python).

D. Preprocessing Methods and Explanations

Preprocessing is a crucial part in every language processing project, as computations can only be performed on numeric values and not on raw text. In the following, different approaches of transforming the given text-data into numerical data are shown:

a. TF-IDF

The "Term Frequency - Inverse Document Frequency (TF-IDF)" [3] -Algorithm calculates the importance of a word in the context of the containing document as well as the whole dataset. Formula … shows how the importance-value tfidf(t,D) of a single word t in a dataset D is calculated. tf(t,d) refers hereby to the frequency of the word t in the containing document d, whereas idf(t,D) gives a logarithmic coefficient of the total number of documents N divided by the number of documents in the dataset, which contain the given word t.

$$tfidf(t,D) = tf(t,d) * idf(t,D)$$

$$tf(t,d) = f_{t|d}$$

$$idf(t,D) = log(\frac{N}{|\{d \epsilon D : t \epsilon d\}|})$$

The multiplication of the idf-term ensures that common words like "and" or "the" are weighted with low a importance-value despite their high occurrence.

The TF-IDF-Algorithm therefore returns a scalar value for every word in the dataset.

b. Glove

Another approach is the vectorization of words using word-embeddings [4]. Fig. 2 visualizes the single steps of this process.

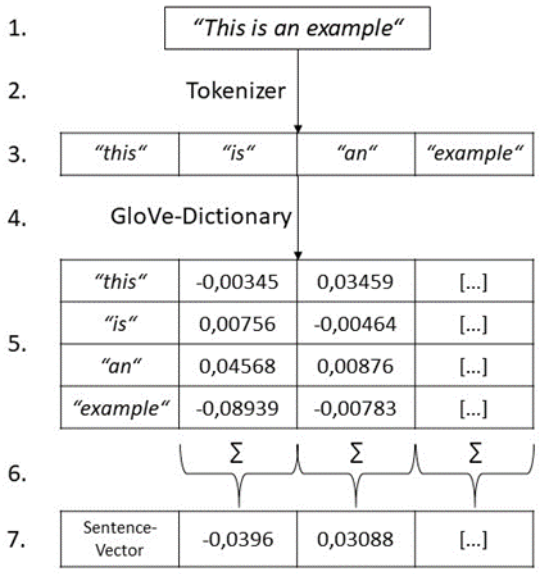

Fig. 2 Preprocessing Using Glove Dictionary and NLTK Tokenizer

A sentence or document (1) is divided into single logical units (tokens) using a tokenizer-algorithm (2) like "Natural Language Toolkit (NLTK)" [5]. To save computation time and memory, stopwords like "and" or "the", which do not provide contextual information, are filtered out. The tokenized vector (3) is then matched with a dictionary of word-embeddings (4), which consists of numerical vectors. These vectors refer to a single word and are computed by decomposition of a co-occurrence-matrix. A co-occurrence-matrix stores information about the amount of usages of a certain word in the context of another word in a dataset. To simplify calculations, in this work the pre-trained decomposition-matrix GloVe [6] is used. After extracting the GloVe-vectors (5) for the single words of a sentiment in the sentiment-analysis-dataset, element-wise summarization (6) and normalization by the absolute value is performed to provide a vectorized representation of the full sentiment (7).

E. Models Choices and Explanations

a. Decision tree

Discussed during the lectures.

b. Random forest

Random Forest classifier is a ensemble model of multiple Decision Tree classifiers with the same hyperparameters. The single trees in the forest are trained on subsets of the training-dataset which results in varying shapes of the trees. For the final classification, a majority vote of the prediction of every tree of the data is performed, which generally results in superior results compared to the prediction of a single tree.

In our work, the hyperparameters of the best performing single Decision Tree were used to train a forest with 1000 trees.

c. Deep Neural Network (fully connected)

A deep fully connected neural network is used as a baseline model for deep learning.

d. Gated Recurrent Units (GRU)

GRU models [7] introduced in 2014 by Cho, et al. are improved versions of standard Recurrent Neural Network [8][9].

The main drawback of standard recurrent neural network is that at each time step during training, the same weights are used to calculate the predicted word at a given step t. That multiplication is also performed during back-propagation. The further we move backwards, the bigger (or smaller) the error signal becomes. In other words, a standard recurrent neural network experiences difficulties in memorising words that are far away in the sequence and therefore, is making predictions on only the most recent ones. This phenomenon is also called the "vanishing gradient problem".

GRU as well as LSTM models were developed to solve the above issue and became a usual way to implement Recurrent Neural Networks. In short, the idea of the GRU model, is that to solve the "vanishing gradient problem", it uses an "update gate" and a "reset gate", two vectors which decide what information should be passed to the output. They can be trained to keep information from long ago, without washing it through time or remove information which is irrelevant to the prediction. GRU model has 2 gates instead of 3 as in the case of LSTM models. The "reset gate" determines how to combine the previous memory with the current input, and the "update gate" decies how much of the previous memory to keep around.

Previous findings [10] demonstrates that GRU models are indeed better than more traditional recurrent units such as tanh units, but also that GRU models are comparable to LSTM on polyphonic music modeling and speech signal modeling.

F. Grid Search Explanation and Choices

a. Scikit learn

Scikit-learn [11] is an extensive custom library for machine learning applications in python. Decision Tree and Random Forest are implemented using scikit-learn, which enables the usage of the Grid-Search-package of the library for the tuning of hyperparameters. The Grid-Search determines the setting of highest accuracy in a predefined parameter-space, using 3-fold cross-validation to avoid overfitting.

b. Talos

Talos [12] is a custom package used for hyperparameter optimization with Keras models. As Keras is utilized in our experiments, Talos is used to find a set of best parameters.

The package incorporates grid, random, and probabilistic hyperparameter optimization strategies, with focus on maximizing the flexibility, efficiency and result of random strategy.

G. Case study: GRU, Influence of Hyperparameters' Tuning on accuracy

a. Changing the Number of Hidden Neurons

b. Introducing Dropout

As GRU models are the last ones implemented, a case study showing the influence of hyperparameters on the predictions' accuracy is presented in this paper.

IV EXPERIMENTS, RESULTS

A. Models' accuracies using Grid Search

a. Models' Best Parameters

For each selected model, a description of the best parameters found is provided below, based on the best validation accuracy on the Training set during grid search.

For classification with few classes, min_samples_leaf=1 is often the best choice. [13]

DECISION TREE

Parameter	***Value for TF-IDF***	***Value for GloVe***
Criterion	[**Gini**, Entropy]	[**Gini**, Entropy]
Max. Depth	[2,3,...,**9**,...,14,15]	[2,3,...,**7**,...,14,15]
Min. Samples Split	[3,4,**5**,6,7,]	[**3**,4,5,6,7,]
Max. Leaf Nodes	[10,20,...,**80**,...,90,100]	[10,**20**,30,...,90,100]
Min. Samples Leaf	[**1**]	[**1**]

Fig. 3 Decision Tree Grid Search Parameters Using Different Preprocessing Methods

ANN & GRU

Parameter	***Value for Fully connected Neural Network***	***Value for Gated Recurrent Unit***
Optimizer	Adam	Adam
Loss-Function	Categorical Crossentropy	Categorical Crossentropy
Epochs	100	100
Batch-Size	2000	2000
Dropout	0.3	0.8
Neurons in 1st hidden Layer/ units in GRU	500	100
Neurons in 2nd hidden Layer/ units in GRU	1200	100
Neurons in 3rd hidden Layer/ units in GRU	300	100

Fig. 4 Fully Connected Neural Network and Gated Recurrent Unit Grid Search Parameters Using Talos

b. Case study: GRU, the Influence of Hyperparameters' Tuning on Predictions' accuracy

GRU ACCURACIES

Number of gated units per hidden layer	***Test Accuracy***
5	84.00 %
100	87.83 %
5, 5	85.33 %
100, 100	88.33 %
800, 800	88.08 %
100, 100, 100	**89.49 %**

Fig. 5 Results of the GRU Case Study

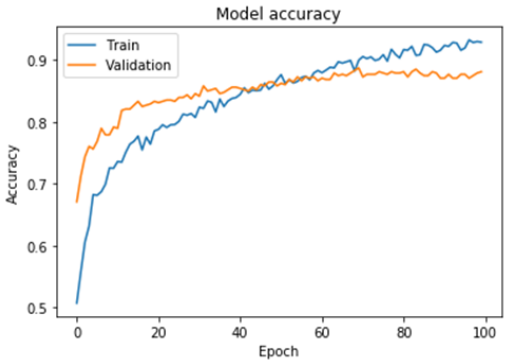

Fig. 6 Evolution of Accuracy on the Validation set and the Training Set During Training of the GRU Model

B. Best Accuracies per Model

MODELS' ACCURACIES

Model	***Train Accuracy***	***Validation Accuracy***	***Test Accuracy***
Decision Tree [TF-IDF]	78.59 %	66.35 %	63.67 %
Random Forest [TF-IDF]	70.52 %	-	69.67 %
Decision Tree [GloVe]	78.88 %	71.81 %	68.50 %
Random Forest [GloVe]	82.96 %	-	77.83 %
Sequential Neural Network	84.55 %	85.94 %	84.00 %
Gated Recurrent Unit	**89.73%**	**89.76 %**	**89.49 %**

Fig.7 Accuracies of the Implemented Models

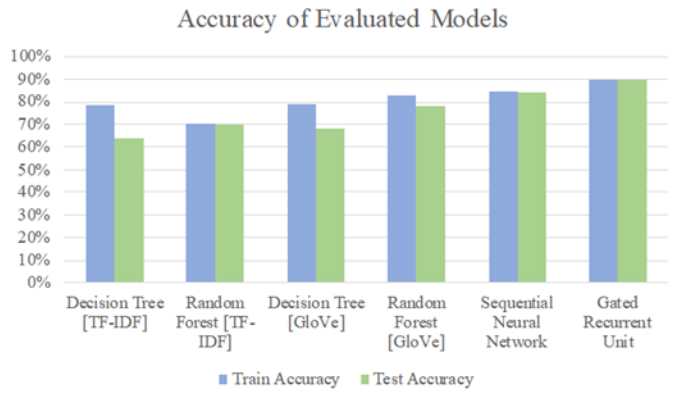

Fig.8 Diagram of Accuracy of Evaluated Models

C. Influence of dataset size on test accuracy

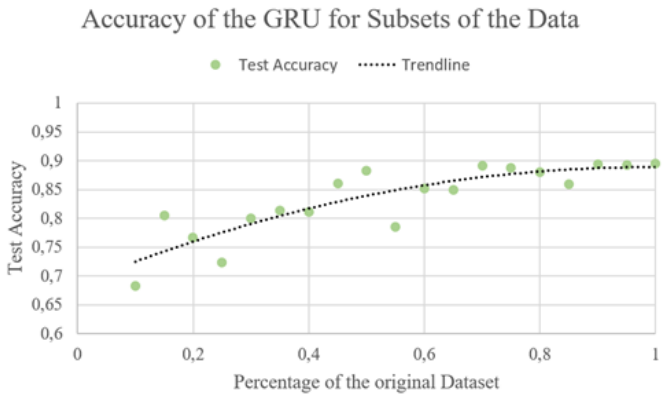

Fig. 9 Accuracy of the GRU Model for Subsets of the Data

V RESULTS OF PREVIOUS WORK ON ^ THE DATASET

	Accuracy		
	Amazon	IMDb	Yelp
Logistic w/ BOW on Documents	85.8%	86.20%	**91.25%**
Logistic w/ BOW on Sentences	88.3%	81.81%	78.16%
Logistic w/ Embeddings on Documents	67.82%	58.23%	81.00%
GICF w/ Embeddings on Sentences	**92.8%**	**88.56%**	88.73 %

Fig. 10 Results Obtained in Previous Research [1]

VI DISCUSSION

As the method is based on Agile methodology, it can only be described by looking at the results at the same time.

1. As explained in the methodology, a baseline model was selected to obtain first results. After a thorough hyperparameters tuning, the first implemented model, Decision Tree, using TF-IDF preprocessing method, provided 63.67 % accuracy on the test set. Besides, the best model overfits the data on the training instances when we compare the accuracy on the training set (78.59 %) and the test set (63.67 %). It is possible that the model is not adapted to the problem, and/or that the data preprocessing does not extract enough informations from the text data.

2. To verify those two possibilities, a second model was implemented: Random Forest, which is an ensemble of decision trees, using similar preprocessing (TF-IDF). After a

thorough hyperparameters tuning, it provided 69.67% accuracy on the test set. The training and test accuracies, 70.52% and 69.67% respectively, are very close, therefore the model does not overfit the data during training and Random Forest model generalizes better and is therefore superior to a single tree.

3. After having changed the model the next step consisted in changing the preprocessing:

After new hyperparameters tuning using a Tokenizer and a Glove dictionary, the accuracy of the predictions on the test set of the Decision Tree and Random Forest models improved from 63.00% to 68.67% and 70.0% to 77.83% respectively. As a result, this preprocessing method was used for further analysis as it outperformed the results obtained with TF-IDF.

4. As many more complex models are available, strategies were developed to improve the results.

Further experiments were performed with a deep fully connected neural network. Using the same preprocessing method and after performing hyperparameters tuning using Talos custom package, the best prediction accuracy achieved on the test set was 84.00%. A great improvement considered that the best model (parameters selected based on the best accuracy on the validation set) did not overfit the data on the training instances (84.55%).

5. However, there is another class of artificial neural networks that allows to exhibit temporal dynamic behavior for a time sequence, unlike feedforward neural networks, where information moves in only one direction. Recurrent Neural Networks use their internal memory to process sequences of inputs. However, as standard Recurrent Neural Networks undergo the "vanishing gradient problem" previously explained in Models Choices and Explanations, a more complex model called GRU, which relies on a gated mechanism in recurrent neural network, was created as a final model. The prediction achieved 89.49% accuracy on the test set, 89.76% on the validation set, and 89.73% on the instances used for training, after a thorough hyperparameters tunings. The best model (parameters selected based on the best accuracy on the validation set) did not overfit the data on the training instances (89.73%).

The observation matches the expectation of models and dataset. This dataset is obviously not a tabular dataset and therefore not suited for decision tree. However, models which find patterns in the data, like deep fully connected neural networks or GRU models, are very suited for this task.

The chosen number of 100 epochs is sufficient for training the model, as the asymptotic behaviour of the validation-accuracy after 70 epochs as shown on the image Fig. Model accuracy.

During the experiments, many combinations of hyperparameters tuning were performed, as well as many architectures. The greatest achievable accuracy on the test set was 89.49%. Possible explanations of why the GRU model cannot perform better are presented below:

6. Data noisiness. The data were assumed to be perfectly selected, but it is possible that some of them are not perfectly defined as positive or negative. To eliminate this possibility, every instances should be checked.

7. Additional data are required to create a model that can generalize better on new instances. To verify this possibility, subsets of the whole dataset have been used to show the evolution of the prediction accuracy on the test set. In Fig 9, an asymptotic trend is observed and proves that adding data to the current dataset will not improve the accuracy of the GRU predictions on unseen instances.

8. The preprocessing used is not strong enough to represent the complexity of the sentences' meaning. This phenomenon is shown when the performances of two random forest models, one using tf-idf and the other glove preprocessing method, were compared (step 1 and 2 of our method).

9. A model is never perfect as it is meant to simplify a complex problem. Therefore, using a more complex model could participate in achieving better results. By taking a look at state-of-the-art architectures like ResNet [14] or DenseNet [15] (or even more complex examples), it is obvious that high-end performing architectures cannot be found by a simple Grid Search.

Finally, to step back even more on the results, it is important to notice that a good accuracy does not necessarily mean that the sentiment analysis is well performed. For example, if a sarcastic sentence is inputted in the model, the later will output a result which is totally opposed to the real meaning. On those sentences, the model would predict as a "idiot savant".

VII CONCLUSION

In this study an NLP problem, focusing on Sentiment Analysis, was examined. Therefore, data analysis, data preprocessing, model selection, and their corresponding hyperparameters were investigated.

Based on the Agile Methodology,

- an easy preprocessing and baseline model (Decision Tree and Random Forest) was firstly implemented.
- different approaches of preprocessing the given data (TF-IDF and GloVe) were evaluated with the baseline model.
- using the prioritized preprocessing method GloVe, more complex Deep Learning models (Fully connected Neural Network and Gated Recurrent Unit) were implemented.
- using the prioritized model GRU, the dataset was examined for sufficient quantity by evaluating different sized subsets.

The superior results of the deep learning models FCNN and GRU compared to the Decision Tree classifier show that for a non-tabellaric dataset,

The sentiment analysis-dataset is a complex, non-tabular dataset without clearly defined attributes. Features that distinct the data therefore have to be extracted prior to classification. The superior results of the deep learning models FCNN and GRU, which are suited to comprehend arbitrarily complex data, compared to the Decision Tree classifier, which perform best on easy interpretable data structures, proof this assumption.

Data-Preprocessing with the GloVe-Dictionary resulted in higher accuracy on the test-data and less overfitting compared to the TF-IDF-Algorithm. This could be explained by the more complex vectorized representation of the tokens with GloVe, which carries a higher amount of information, compared to the single TF-IDF-values per token.

Compared with state-of-the-art results on the dataset (Chapter V) and given the fact, that in our work a single model and preprocessing was used for generalizing on yelp, imdb and amazon sentences combined instead of individual models and preprocessing, the used GloVe-Vectorizer and GRU-model with an accuracy of 89.49 % is very well suited to classify the sentiment-analysis-dataset.

REFERENCES

[1] Dimitrios Kotzias1 Misha Denil Nando De Freitas, Padhraic Smyth, "From Group to Individual Labels using Deep Features", http://mdenil.com/media/papers/2015-deep-multi-instance-learning.pdf

[2] Kaggle https://www.kaggle.com/docs/kernels#technical-specifications

[3] Stephen Robertson, "Understanding Inverse Document Frequency: On theoretical arguments for IDF",

[4] Jeffrey Pennington, Richard Socher, Christopher D. Manning, "GloVe: Global Vectors for Word Representation", https://nlp.stanford.edu/pubs/glove.pdf

[5] Natural Language Toolkit https://www.nltk.org/

[6] Pretrained Glove Matrix https://nlp.stanford.edu/projects/glove/

[7] Cho and al., "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation", https://arxiv.org/pdf/1406.1078v3.pdf

[8] Zachary C. Lipton, John Berkowitz, Charles Elkan, "A Critical Review of Recurrent Neural Networks for Sequence Learning", 2015, https://arxiv.org/pdf/1506.00019.pdf

[9] Stanford University School of Engineering, "Lecture 8: Recurrent Neural Networks and Language Models", 2017, https://www.youtube.com/watch?v=Keqep_PKrY8&t=1080s

[10] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling", 2014, https://arxiv.org/pdf/1412.3555v1.pdf

[11] https://scikit-learn.org/stable/

[12] https://github.com/autonomio/talos

[13] https://scikit-learn.org/stable/modules/tree.html?fbclid=IwAR1plcIhlvB966BB8PXunSEfxq3KThG5kgv_QxfqrBUi3vHIyDPTHzjpBMY#tips-on-practical-use

[14] K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition: RESNET https://arxiv.org/pdf/1512.03385.pdf

[15] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, Densely Connected Convolutional Networks https://arxiv.org/pdf/1608.06993.pdf