Dreaddit: A Reddit Dataset for Stress Analysis in Social Media (2019)

Advanced Machine Learning - Final Project

Shir Cohen 205790124, Guy Assa 204118616

1. Introduction

1.1. Problem statement

Stress detection with neural and traditional supervised learning models reached F1-score of 0.8065 with pretrained BERT-base model and 0.798 F1-score of LogReg + Word2Vec + features model as mentioned in Dreaddit: A Reddit Dataset for Stress Analysis in Social Media (2019).

As for the innovations made in the past 3 years since the article was published, we will try to improve their model and achieve better estimation parameters for the exact same goal, with new approaches and tools, focusing on word representation (word-embedding methods).

Our dataset is Dreaddit, a new text corpus of lengthy multi-domain social media data used to identify stress. This dataset consists of 190K posts from five different categories of Reddit communities. There are 3.5K total label segments taken from 3K posts.

The performance of the improved models will be measured by Precision, Recall, and F1-score, and the F1-score will be used to compare the models.

2. Related work

Emotion-Infused Models for Explainable Psychological Stress Detection (Turcan et al., NAACL 2021)

This paper is a continuation of the anchor paper using the same dataset. In this work, the authors were focused on exploring the use of multi-task learning as well as emotion based language model fine-tuning. The models in this work rely on two types of enhancements to the neural representation learned by models like BERT: multi-task learning and pre-training or fine-tuning.

Detection of Mental Health Conditions from Reddit via Deep Contextualized Representations (Jiang et al., Louhi 2020)

This paper addresses the problem of automatic detection of psychiatric disorders from the linguistic content of social media posts. They build a large scale dataset of Reddit posts from users with eight disorders and a control user group and classification models based on deep contextualized representations and demonstrate that they outperform the LIWC feature based logistic regression baseline by a large margin.

Bag of Tricks for Efficient Text Classification (Joulin et al., EACL 2017)

This paper presents a simple and efficient baseline for text classification called **fastText**.
In this work, they explore ways to scale these baselines to a very large corpus with a large output space, in the context of text classification. They show that linear models with a rank constraint and a fast loss approximation, which are considered a strong baseline for text classification problems, can train on a billion words within ten minutes with the right features, while achieving performance on par with the state-of-the-art. They provide evaluation results of the fastText approach on two different tasks, namely tag prediction and sentiment analysis.

The model architecture is based on an improvement of the bag of words (BoW) sentence representation that can share parameters among features and classes by factoring the linear classifier into low rank matrices with rank constraints and hierarchical softmax which improve running time and computational complexity. The results showed that fastText obtains performance on par with recently proposed methods inspired by deep learning, while being much faster.

GloVe: Global Vectors for Word Representation (Pennington et al., EMNLP 2014)

This paper presents a new global log bilinear regression model that combines the advantages of the two major model families in the literature: global matrix factorization and local context window methods which called **GloVe**, for Global Vectors, because the global corpus statistics are captured directly by the model.

This model efficiently leverages statistical information by training only on the nonzero elements in a word-word co occurrence matrix, rather than on the entire sparse matrix or on individual context windows in a large corpus. The model produces a vector space with meaningful substructure, as evidenced by its performance of 75% on a recent word analogy task. It also outperforms related models on similarity tasks and named entity recognition.

Distributed Representation, LDA Topic Modelling and Deep Learning for Emerging Named Entity Recognition from Social Media (Jansson & Liu, 2017)

This paper shared a task on emerging and rare entity recognition from user generated noisy text such as tweets, online reviews and forum discussions. To accomplish this challenging task, they explore an approach that combines **Latent Dirichlet Allocation (LDA)** topic modelling with deep learning on word level and character level embeddings.
The LDA topic modelling generates topic representation for each tweet which is used as a feature for each word in the tweet.

As an optional input to the modelling process, we introduce LDA topic modelling as an option in this paper. The LDA topic modelling generates topic representation for each text which is used as a feature for each word in the post. In our implementation, the number of topics was chosen to correspond to the number of subreddits, and the LDA model results were added.

3. Anchor Paper Review

3.1. Anchor paper implementation

The anchor paper is ‘Dreaddit: A Reddit Dataset for Stress Analysis in Social Media’

References for external code repositories:

- https://paperswithcode.com/dataset/dreaddit
- https://www.kaggle.com/code/sohommajumder21/bert-tokenizer-with-9-models-nlp-stress-analysis
- https://github.com/fsentin/dreaddit
- https://medium.com/@dilip.voleti/classification-using-word2vec-b1d79d375381
- https://towardsdatascience.com/fine-tuning-bert-for-text-classification-54e7df642894
- https://mccormickml.com/2019/07/22/BERT-fine-tuning/#1-setup

Our mission was to implement the anchor paper methods for classification to establish a baseline for improvement by implementing innovations in the second part of this project.

First, we pre-processed the Reddit dataset by converting the labels to numerical features and removing irrelevant features. We follow the paper's decision to focus on the features with the highest Pearson correlation (r >= 0.4). Then, we perform text pre-processing methods to clean the data such as lower case, remove stop words, remove punctuation, and common words.

Second, we implemented the classification models mentioned in the paper.
We manage to reach the same results as the paper on the **Majority Baseline** model.
In addition, we experimented with supervised non-neural models, including Support Vector Machines (SVMs), Logistic Regression, Naïve Bayes, and decision trees with text representation of **Bert embedding and Word2Vec embedding**.

We managed to reproduce the best model result of the **Logistic Regression** classifier with Word2Vec embedding, high correlation features, and high agreement data. This model achieves an F1-Score of ~77.

Finally, we fine-tuned the pre-trained **Bert-base model** on our dataset for three epochs with the parameters mentioned in the paper. In this case, we did not reach the same results as the paper. Our assumption is that the gap lies in the pre-trained Bert process, which was not indicated in the article explicitly.

Link to the Git repository

3.2. Anchor paper results

We first experienced the non-neural models. We decided, and it is not mentioned in the article, to experience each model with some embedding vectorization. Thus, we present the results of each non-neural model with both BERT embedding and Word2Vec embedding.

As we can see, and the article declares - Logistic Regression is best performed regarding F1-score, and overall measurements (precision and recall) do not fall far from the peak. Thus, it is reasonable to choose the Word2Vec Logistic Regression model.

Model	**BERT**			**Word2Vec**		
	Precision	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
Naive Bayes	**0.7697**	0.0398	0.0753	0.6560	0.7621	0.7046
Random Forest	0.6867	0.6043	0.6335	0.7193	0.6615	0.6943
Decision Tree	0.6750	**0.8123**	0.7370	0.6750	**0.8123**	0.7370
AdaBoost	0.7273	0.7554	0.7403	0.7054	0.8450	0.7684
XGBoost	0.7240	0.7592	0.7406	0.7253	0.7692	0.7458
SVM	0.6995	0.7989	0.7453	0.7208	0.7648	0.7413
Logistic Regression	0.7244	0.7708	**0.7463**	**0.7496**	0.7901	**0.7688**

In a comparison with the anchor paper, we see similar results for the models in the paper and our implementation that was intended to replicate them.

Model	**Anchor Paper**			**Our Implementation**		
	Precisio n	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
Majority Baseline	0.5161	1.0000	0.6808	0.5161	0.5161	0.6808
LogReg w/ pre-trained Word2Vec + features	0.7346	0.8103	0.7706	0.7496	0.7901	0.7688
LogReg w/ pre-trained BERT + features	0.7704	0.8184	0.7937	0.7244	0.7708	0.7463
BERT-base	0.7518	0.8699	0.8065	0.7054	0.8450	0.7684

This fulfils the essence of the article, although these are not exactly the same results, the ratio remains the same and leads to the same conclusions. As we can see in the table above, the majority baselines have the same results in both implementations. Logistic regression with Word2Vec achieves similar results in all metrics (F1-Score of 77) and the implementation with BERT has some gaps as mentioned in the previous section.

We figured that the better word2vec results would be due to the fact that we trained our model on text (3,553 posts) while the anchor paper trained its model on the entire corpus, leading to better results.

4. Innovation

4.1. Overview

Our main goal of the innovation part was to use the same data set of the anchor paper, however to improve the prediction measurements significantly.

As the anchor paper suggests, we tried to improve both Deep learning methods (using BERT), and Machine learning approaches as well.

At first, we tried some text pre-processing methods in order to achieve better results without changing anything in the method itself. After evaluating, we observed no significant change in one case, and major decrease in the other case, so our conclusion was that it is not regarding more sophisticated pre-processing.

From that point, we continue to investigate ways to improve the basic model, which is as suggested in the anchor article - a model with the 3 lexical features: `lex_liwc_Tone`, `lex_liwc_Clout` ,`lex_liwc_i`

For our next step, we investigate some other feature generation:

First, we tried to count the number of spelling mistakes as a counter for stress. This task is quite hard and not so accurate, thus we abandoned it.

The second feature was the number of punctuations. As some sources suggest (like this one, for example), punctuation has a meaningful role in expressing emotion, and stress is one of them. Due to this fact, a new feature that we selected is the normalized number of punctuations per text. We calculated it by counting the number of punctuations, divided by the length of the text.

Then, we moved on to feature engineering of word representations using ML tools.

We present here the following methods:

- Word2Vec (as suggested in the anchor paper)
- LDA topic modeling
- Glove
- Textfast
- BERT

Word2Vec:

As a brief reminder from the anchor paper, the word2vec is a technique for natural language processing published in 2013. The word2vec algorithm uses a neural network model to learn word associations from a large corpus of text.
In our case, we trained the algorithm on our corpus (as suggested in the article)

LDA topic modeling:

Topic Modeling in NLP seeks to find hidden semantic structure in documents. Latent Dirichlet Allocation (LDA), which was a technique proposed in 2000 for population genetics and re-discovered independently by ML-hero Andrew Ng et al. in 2003. LDA states that each document in a corpus is a combination of a fixed number of topics. A topic has a probability of generating various words, where the words are all the observed words in the corpus. Although all the posts were taken from 5 different subreddits with a clear topic, we observed that much of the data (text which is part of a post), is not related directly by content to the topic it came from.

LDA topic modeling can help us classify the data into unseen clusters and thus help with the task of classification, as suggested in this paper.

GloVe:

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. GloVe first presented by Pennington et al. 2014, from Stanford.

Both W2V and glove are methods to produce a word vectorization, however, the two models differ in the way they are trained, and hence lead to word vectors with subtly different properties. Glove model is based on leveraging global word to word co-occurrence counts leveraging the entire corpus, while Word2vec on the other hand leverages co-occurrence within local context (neighbouring words).

Although both methods can lead to similar results, there are places where we can see an advantage to GloVe like in this case.

In our case, and with contrary to W2V, we used pre-trained corpus in order to vectorize our data. Out of the corpuses that exist, we chose `'glove.6B.100d.txt'` as for a trial and error and some cost effective (or in our case size-accuracy) measures.

FastText:

fastText is an open-source library, developed by the Facebook AI Research lab. Its main focus is on achieving scalable solutions for the tasks of text classification and representation while processing large datasets quickly and accurately.It achieves this computational efficiency and accuracy by using both bag of words and Hierarchical Softmax to address classification and training word representations of text.

Thus, FastText is essentially an extension of the word2vec model, however with a slight difference - word2vec treats each word in the corpus like an atomic entity and generates a vector for each word. In this sense Word2vec is very much like Glove — both treat words as the smallest unit to train on. FastText treats each word as composed of character ngrams. So the vector for a word is made of the sum of this character n grams.

According to their research, fastText stacks impressively in both accuracy and training and testing times against previously published state-of-the-art models.

In the FastText case, we use a pre-trained corpus to achieve better results. We used the English corpus `'cc.en.300.bin'`

Thus, we decided to go deep and explore all existing novel options that the anchor papar's article did not use, in order to get better performance.

BERT:

We already implemented the BERT in the first part (the scores can be seen in the Anchor paper chapter). However, during our research for innovation we realized we did not perform our best regarding the BERT.

Thus, in order to achieve a better score (and even better than the article, as you'll see in the results), we have decided to fine-tune the model's parameters and reconstruct (only small reconstruction) of the BERT in order to manipulate it towards our task and better classification. The reconstruction itself can be seen in the notebook in the Implementation section. However, in a nutshell, we tried to build a BERT classifier that best suits our task - small data, binary classification, long embedding (some posts are relatively long).

Some properties of our BERT -

- Different text preprocessing
- Maximum length of sentence - 300
- Batch size - 16
- Added special tokens (`[CLS]` and `[SEP]`)
- Hidden size of BERT - 768
- Hidden size of our classifier - 50
- Number of labels - 2
- AdamW optimizer with default learning rate and epsilon
- More details in the notebook

Another method that is not presented in our paper but worth mentioning is User-related post aggregation - as we have the knowledge that the posts in the data set were splitted into different texts, we tried to cluster all texts that belong to the same post.

We did it in order to try and classify more correctly the labels of stress (that were classified manually by humans). The process was as follows: We first aggregated all parts of the same post, and then classified it according to the majority vote of each sub-post. Let's say we had 3 parts of a given post, which was classified as 1,1,0 , so the post itself got a score of 1. If a given post had an even number of sub-posts, and the stress/no stress label was equal, we determine the label by summing the "confidence" feature (which indicates how many humans labelled it - 1 equals all labelled it as its label).

This method reduced our samples (as we merged the sub-posts), however we managed to classify more correctly other parts of the post, under the assumption that a person in stress will express it across the whole post, even if he did not use any "stressful" words. However, after doing so the sample size decreased and the results weren't significant enough so we did not include this part.

4.2. Implementation

Link to the Git Repository

4.3. Results

In this section we will discuss the results of our analysis.

It is important to keep in mind that we have tried multiple analyses and will present here the most relevant work that has been done.

Important note - while exploring related works to compare with, we found the following article, that is part of September 2022 volume of the journal (which means it is hot from the oven) and basically tried to accomplish our mission, with similar methods, however they applied the methods over 2 other data sets that we did not do. This article is using a new set of features, constructed from other sources of lexicons other than LIWC that the anchor paper (and we as well) are using, and this is part of the explainable gap between the results.

Our comparison will be with regards to the Anchor paper (denoted as “Anchor paper”), our work (denoted as “Our paper”) and the new paper that we discovered (denoted as “New paper”). *the “New Paper” reports only F1 score*

Model	**Anchor Paper**			**Our paper**			**New paper**
	Precision	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**	**F1-Score**
LogReg w/ Word2Vec + features	0.7346	0.8103	0.7706	0.7564	0.7913	0.7735	0.798
BERT-base	0.7518	0.8699	0.8065	0.8263	0.8222	0.8228	0.8479

Although we did not managed to reach to results of the new paper (they did not refer to how they constructed the models, only F1 score), it is noticeable that due to the reconstruction of BERT-base we managed to perform much better than the anchor paper (with related to the logistic regression we got the same score).

For the comparison between our paper to the new paper -

Model	**Our paper**			**New paper**
	Precision	**Recall**	**F1-Score**	**F1-Score**
LogReg w/ LDA + features	0.6821	0.7154	0.8968	-
LogReg w/ Word2Vec + features	0.7564	0.7913	0.7735	0.798
LogReg w/ GloVe + features	0.7829	0.8211	0.8015	-
LogReg w/ FastText + features	0.8271	0.8428	0.8348	0.8604
BERT-base	0.8263	0.8222	0.8228	0.8479

As we can see, our results improve the measurements of the anchor paper, as we tried to accomplish. The best performance is FastText, which performs significantly on the data set, with the pre-trained corpus. However, we can see that the new paper did achieve better results in all sections.

Part of the explanation is due to the new feature that the new article extracted and used, however the article shows that also without the features at all (only embedding of FastText), they managed to get better F1 score - 0.8487 (we are trying to reach the authors of the paper in order to better understand their models constructions).

5. Summary

During this work, we used the ‘Dreaddit: A Reddit Dataset for Stress Analysis in Social Media’ paper as our baseline that deals with classifying stress from Reddit social media posts of users. They examine both machine learning and deep learning approaches to achieve the best results for stress classification.

Our purpose was to improve their model in order to achieve better results with new approaches of word embedding and fine-tuning of models. We started with adding new features such as number of punctuations and topic modelling with LDA and then improved the word embedding with new techniques such as GloVe and fastText embedding.

We managed to get good results that exceeded the baseline with our best model logistic regression with correlated features, total punctuations and fastText embedding that achieved an F1 score of 0.8348, an improvement of ~3%.

As a result of our experiments, we find out that extending the features with relevant information from the data along with different word embedding techniques can lead to improving the model. Despite this, the model was trained on the same dataset as the anchor paper, and therefore we believe that it should also be tested on other datasets in the future.