Emotion analysis in dataset

Meriem Aggoune
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

Lauri Heikka
Faculty of Information Technology and Electrical Engineering
Univ. of Oulu
Oulu, Finland
lauri.heikka@student.oulu.fi

Anusha Ihalapathiranae
Faculty of Information Technology and Electrical Engineering
Univ. of Oulu
Oulu, Finland
anusha.ihalapathirana@student.oulu.fi

***Abstract*—We document a series of experiments on using a variety of natural language processing methods to classify emotions conveyed by sentences. In particular, we compare word categorizations, semantic similarity approaches and machine learning approaches. The machine learning approaches consist of classification models based on bag-of-words representations and convolutional neural networks applied on word embeddings. Our results indicate that machine learning approaches outperform string matching and semantic similarity with a considerable margin. Within the machine learning approaches, convolutional neural networks with word embeddings provide an improvement in accuracy of around 3-5 percentage points over bag-of-words models. The best accuracy on a hold-out validation set, around 93%, is achieved with a CNN approach using pre-trained word2vec embeddings. However, custom-trained word embeddings provide similar performance with less computational overhead.**

***Index Terms*—natural language processing, text classification, sentiment analysis**

I. Group info

Group 10.

Members: Meriem Aggoune, Lauri Heikka, Anusha Ihalapathiranae

Title: Emotion Analysis in Dataset

Github: https://github.com/MeriemLil/NLP_Project

II. Introduction

Emotions are feelings that caused by a situation person interact with. It is also associate with person's character, mood, personality and motivation. People expressed their feelings and emotions using verbally and non-verbally such as words, intonation of voice, facial expressions, gestures, and tears. Nowadays people tend to communicate their ideas, opinions, and feelings through social medias, Such as tweeter, facebook, instergram. People filled with lot emotions and they commonly use written text to express their emotions and feelings through social medias. Categorizing text into emotion types is known as emotion analysis. Sentiment analysis detect positive, negative, and neutral feelings from text and emotion analysis detect the types of feelings, emotion state, in the text. Such as love, anger, joy, sadness, surprise, and fear[c]

A lot of research had been carried out on sentiment analysis and emotion analysis problem in many different languages. Chaffar et al.[d] have demonstrate their work on supervised machine learning approach to recognize the emotion types using emotion annotated dataset which combined news headlines, fairy tales and blogs. They used bag of words and N-grams and proved that support vector machines classifier shows better performance and generalization than the other classifiers they used.

Johnson et al.[e] introduced a low complexity word level deep convolutional neural network for text categorization and Dos Santos et al. [f] propose new convolutional neural network that exploits from character to sentence level information to perform sentiment analysis of short text such as tweeter messages. And they found that combine small text with prior knowledge is effective.

III. Data and Methodologies

A. Datasets and sources

Our main dataset of interest is a labeled dataset of 20 000 sentences with between 3 and 66 words. The dataset is provided by [**?**]. The sentences are labeled to convey one of six different emotions: fear, anger, joy, love, surprise and sadness. The dataset is collected following the approach of [**?**]. The sentences are tweets, and the emotion labels are hashtags at the end of the tweet. The sentences are preprocessed in that they contain no punctuation or special characters and all words are lowercase.

The dataset has a predefined split into training, test and validation samples, with 16 000, 2 000 and 2 000, respectively. Because the semantic similarity and string matching approaches do not employ any predictive modeling, in these sections we use the all 20 000 sentences to measure prediction accuracy. For the machine learning approaches, we use the training set for model training, the test set for model selection and hyperparameter tuning, and reserve the validation dataset as a final benchmark, to ensure a fair comparison accross approaches.

In addition, we use pre-trained, 300-dimensional word2vec embeddings from [?] and pre-trained fastText embeddings from [?]. Both datasets consist of 300-dimensional word embeddings, with embeddings for 3 and 1 million english words, respectively. For the string matching, we use the Harvard General Inquirer dataset, which contains categorizations for around 11 000 english words.

1) Harvard Inquirer and String Matching: The first method we explore is to categorize each word in Harvard General Inquirer[a] to each emotion type. We used filter technique to this categorization, as an example, to identify words related to love emotion, we include Harvard General Inquirer words with Affil label and exclude words associated with Negative label and save data in our SQLite database.

We combined training, testing and validation data sets together and generate complete dataset to use in future tasks. Use complete dataset to perform string matching between every sentence in the previously created database and each emotion category.

2) Empath Client Categories: In next step we used Empath Client[b], a text analyzing tool with pre-build category set, to compute empath category similarity by mapping labeled sentences in dataset and empath categories using hypernyms. We also map using exact empath emotion category.

3) Semantic Similarity: asd

4) SentiStrength sentiment scores: As part of the project specification, we also use the SentiStrength client [?], to compute sentiment scores. These can be potentially be used as an additional feature in any approach to distinguish between the negative and positive emotions.

B. Bag-of-Words Models

asd

C. Convolutional Neural Networks

We replicate the convolutional neural network architecture used in [?]. The architecture is outlined in 1. Sentences are represented as $n \times k$ matrices of word vectors, where n refers to the length of the sentences and k to the dimension of the word vectors. All sentences are padded to length n.

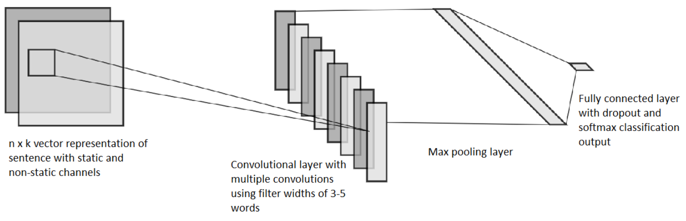
n x k vector representation of sentence with static and non-static channels
Convolutional layer with multiple convolutions using filter widths of 3-5 words
Max pooling layer
Fully connected layer with dropout and softmax classification output

Fig. 1. Example of a figure caption.

In [?], pre-trained Word2Vec embeddings from [?] [?] are used. In addition to replicating this approach, we consider a few alternatives. First, we test pretrained fastText embeddings . Second, we explore explore training our own word2vec and fastText embeddings. The word2vec model... The fast-text model... For both models, we use a maximum distance between predicted words of five, and use early stopping on a word analogy evaluation task.

We experiment with 50-, 100- and 300-dimensional word embeddings for the custom-trained word embeddings. With the pre-trained embeddings, we are restricted to the commonly available 300-dimensional word embeddings.

IV. Results and Discussion

A. Word categorizations and semantic similarity

TABLE I
WORD CATEGORIZATIONS AND SEMANTIC SIMILARITY ACCURACIES

Classifier	**Accuracy**
Harvard String Matching	0.336
Empath Client	0.160
Empath Client with exact categories	0.183
Semantic Similarity - One synset	0.269
Semantic Similarity - All synset	0.240

B. Model selection for bag-of-words

Table V reports the accuracies on the test dataset used for model selection.

TABLE II
MAXIMUM ACCURACY ON THE TEST DATASET ACROSS DIFFERENT BAG-OF-WORDS SETUPS

parameter	***value***	***score***
vectorizer	CountVectorizer	0.886
	TfidfVectorizer	0.888
stopwords	nltk	0.888
	none	0.878
	sk	0.884
lemmatize	0	0.888
	1	0.886
max_features	100	0.396
	500	0.728
	1000	0.865
	2000	0.880
	3000	0.886
classifier	DecisionTreeClassifier	0.866
	GradientBoostingClassifier	0.849
	LogisticRegression	0.878
	MultinomialNB	0.859
	RandomForestClassifier	0.888
	SVC	0.876

C. Validation set performance of best models

Table III reports the accuracies on the test dataset used for model selection.

V. Overall discussion and related literature

...

As mentioned in the introduction, the state of the art for text classification tasks has moved beyond word level embedding representations considered here. For example, [?], [?] and

TABLE III
VALIDATION AND TEST SET ACCURACIES FOR BEST MODELS

Classifier	**Accuracy**	
	Validation	***Test***
MultinomialNB	0.681	0.694
LogisticRegression	0.872	0.864
RandomForestClassifier	0.899	0.89
CNN fastText pretrained	0.925	0.928
CNN Word2Vec own	0.928	0.929
CNN fastText own	0.928	0.931
CNN Word2Vec pretrained	0.930	0.928

TABLE IV
VALIDATION PRECISION AND RECALL FOR BEST MODELS

Classifier	**Metric**	
	Precision	***Recall***
MultinomialNB	0.734	0.681
LogisticRegression	0.875	0.872
RandomForestClassifier	0.898	0.899
CNN fastText pretrained	0.926	0.925
CNN Word2Vec own	0.93	0.928
CNN fastText own	0.928	0.928
CNN word2vec pretrained	0.930	0.930

[**?**] employ pre-trained bidirectional encoder representations to achieve state-of-the-art results in sentence classification. However, constructing a replication and a fair comparison of these models is beyond the scope of this study. [1] Exploring alternative setups for training the word vectors is likewise beyond the scope of this paper.

The pretrained embeddings are quite large in size, while custom embeddings trained on the emotions dataset are much smaller. Furthermore, because model performance with embeddings trained on the emotions dataset is very close in performance to the pretrained embeddings, the custom models provide a lightweight alternative for model serving. It is important to note, however, that the emotions dataset contains very homogenous sentences across the training, test and validation datasets. This causes the vocabulary of the custom word embeddings to be much smaller than the pretrained word embeddings. Therefore the model is more likely to suffer in performance on out-of-distribution sentences.

...

VI. CONCLUSION

conclude.

VII. LATEX EXAMPLES

To be removed.

A. Equations example

$$a + b = \gamma \tag{1}$$

[1]There is a minimal, unofficial implementation of BERT [**?**] embeddings on the dataset we study [**?**] available at Kaggle by the dataset author. This implementation appears to exceed the accuracy of the best studied CNN model by around 0.3 percentage units.

Use "(1)", not "Eq. (1)" or "equation (1)", except at the beginning of a sentence: "Equation (1) is . . ."

B. Some Common Mistakes

- The word "data" is plural, not singular.
- In your paper title, if the words "that uses" can accurately replace the word "using", capitalize the "u"; if not, keep using lower-cased.
- Be aware of the different meanings of the homophones "affect" and "effect", "complement" and "compliment", "discreet" and "discrete", "principal" and "principle".
- Do not confuse "imply" and "infer".
- The prefix "non" is not a word; it should be joined to the word it modifies, usually without a hyphen.
- There is no period after the "et" in the Latin abbreviation "et al.".
- The abbreviation "i.e." means "that is", and the abbreviation "e.g." means "for example".

ACKNOWLEDGMENT

We thank Mourad Oussalah and other seminar participants for their helpful comments during the Natural Language Processing and Text Mining course project seminar.

APPENDIX A
ADDITIONAL TABLES

TABLE V
AVERAGE ACCURACY ACROSS DIFFERENT BAG-OF-WORDS SETUPS

parameter		
	value	***score***
vectorizer	CountVectorizer	0.719
	TfidfVectorizer	0.727
stopwords	nltk	0.725
	none	0.703
	sk	0.74
lemmatize	0	0.723
	1	0.723
max features	100.0	0.361
	500.0	0.631
	1000.0	0.830
	2000.0	0.844
	3000.0	0.845
classifier	DecisionTreeClassifier	0.702
	GradientBoostingClassifier	0.729
	LogisticRegression	0.733
	MultinomialNB	0.704
	RandomForestClassifier	0.744
	SVC	0.723

TABLE VI
BEST CNN MODEL BY MODE

	Metric		
name	***mode***	***Accuracy***	***Loss***
CNN Word2Vec own	multichannel	0.927	6.404
	non-static	0.929	6.44
	static	0.900	10.404
CNN fastText own	multichannel	0.921	9.463
	non-static	0.931	6.599
	static	0.770	25.254
CNN fastText pretrained	multichannel	0.928	6.326
	non-static	0.928	6.133
	static	0.908	8.786
CNN word2vec pretrained	multichannel	0.928	6.583
	non-static	0.927	6.668
	static	0.906	9.025

TABLE VII
HYPERPARAMETERS FOR THE BEST CNN MODELS

	Metric			
Hyperparameter	***CNN fastText pretrained***	***CNN Word2Vec own***	***CNN fastText own***	***CNN word2vec pretrained***
dev acc	0.928	0.929	0.931	0.928
dropout	0.5	0.75	0.75	0.7
num feature maps	100.0	150.0	100.0	100.0
regularization	0.0	0.001	0.001	0.0
word dim	300.0	50.0	50.0	300.0

TABLE VIII
CONFUSION MATRICES

	Label					
Classifier	***joy***	***sadness***	***anger***	***fear***	***love***	***surprise***
CNN Word2Vec	664 32	525 26	254 16	196 34	158 30	60 5
own	40 1264	25 1424	21 1709	16 1754	20 1792	21 1914
CNN fastText	674 38	531 32	258 25	171 12	155 22	67 15
own	30 1258	19 1418	17 1700	41 1776	23 1800	14 1904
CNN fastText	664 36	520 17	259 24	195 35	148 27	64 11
pretrained	40 1260	30 1433	16 1701	17 1753	30 1795	17 1908
CNN word2vec	673 36	524 29	255 15	189 27	156 23	64 9
pretrained	31 1260	26 1421	20 1710	23 1761	22 1799	17 1910
LogisticRegression	672 119	516 73	229 23	155 23	122 10	51 7
	32 1177	34 1377	46 1702	57 1765	56 1812	30 1912
MultinomialNB	683 364	514 268	93 3	60 3	12 0	0 0
	21 932	36 1182	182 1722	152 1785	166 1822	81 1919
RandomForestClassifier	659 72	508 43	241 21	185 36	142 21	62 10
	45 1224	42 1407	34 1704	27 1752	36 1801	19 1909