NATURAL LANGUAGE PROCESSING

WITH DISASTER TWEETS

TEAM MEMBERS
-Manasa Reddy
- Yashika Chopra
-A.S Navya
-A.S.N Rishika

INTRODUCTION

- This project is based on the classification of disaster related tweets into real and not real categories.
- These days, social media is playing a crucial role in providing information before, during, and after certain unforeseen circumstances such as natural disasters.
- One of the major social media platforms used for the collection of disaster related information is twitter.
- Tweets may contain information of multiple purposes which include seeking help, expressing grief, warning others about the disaster etc.

- The key component in Natural language processing involves understanding the events described in the text.
- So, identifying the tweets which actually talk about a real disaster is very important and we have tried multiple word representations, topic modeling techniques and text classification techniques for this purpose.
- We have used certain evaluation metrics like accuracy, precision, recall and f1-score to test the models.

OBJECTIVES

1. Compare different word embeddings with topic classification/modelling techniques.
2. See how supervised and unsupervised techniques work for determining whether a tweet is about a real disaster or not.
3. Exploring recently developed models like BerTweet and see if they yield better results than the existing ones.

Data set

- From a competition hosted on Kaggle
- 7613 data samples containing 7503 unique tweets
- Each sample in the data has the following 5 attributes:

1. id - a unique identifier for each tweet
2. text - the text of the tweet
3. location - the location the tweet was sent from (can be blank)
4. keyword - a particular keyword from the tweet (can be blank)
5. target = 1 if the tweet is about a real disaster, = 0 if not

Data set

An example of a tweet that is talking about a real disaster.

id	keyword	location	text	target
4			Forest fire near La Ronge Sask. Canada	1

And this is an example of a tweet which does not talk about a real disaster.

id	keyword	location	text	target
53	ablaze	London, UK	On plus side LOOK AT THE SKY LAST NIGHT IT WAS ABLAZE	0

Text Preprocessing

The following preprocessing steps have been performed on the dataset :

1. Removed the location column as 33% of the location data contained null values

2. Removed the keyword column as we were not using it

3. Converted the entire tweet text to lower case to avoid increase in the size of the vocabulary for words written in different cases

4. Removed all the duplicate tweets

5. Removed internet links starting with http/s, www and hashtags and user mentions. The Internet links need to be deleted from the tweets, as we were concentrating on the textual content only.

Text Preprocessing (contd.)

6. Removed digits from the tweets

7. Removed punctuations

8. Removed stop words for tf-idf. Stop words are removed to eliminate low-level information from the text in order to give more focus to the important information.

9. Shuffled the rows in the data frame

10. Tokenization was performed for models using Word2Vec, LDA2Vec, models using FastText, models using GloVe. Tokenization helps in interpreting the meaning of the text by analyzing the sequence of the words.

Models used

For Word representations:

1. Word2Vec - a word embedding model, uses neural network to learn word associations from a corpus, can use CBOW or Skipgram, we used CBOW, fine tuned using our data, takes words (tokens) as input
2. FastText - a word embedding model, good for out of vocabulary (OOV) words, fine tuned on our data, does better than Word2Vec on syntactic tasks
3. GloVe - an unsupervised learning algorithm for obtaining word representation of words, we use the pre trained word vectors
4. TF-IDF - a numerical statistic to depict the importance of a word in a document, used it to generate word vectors
5. BERTweet - obtained by fine tuning the BERT model, is a transformer model

Topic Classification vs Topic Modelling

Topic modelling is **a 'unsupervised' machine learning technique**, its a type of statistical model for discovering the abstract "topics" that occur in a collection of documents.

Topic classification is **a 'supervised' machine learning technique**, one that needs training before being able to automatically analyze texts.

Models used

For classifying:

1. *Topic modelling (Unsupervised)-*

a. Latent semantic analysis (LSA) + Logistic regression- LSA for dimensionality reduction to ease computation, logistic regression for classification

b. Latent Dirichlet Allocation (LDA) - is a generative model, treats documents as probabilistic distribution sets of words or topics

Models used

2. *Topic classifiers (Supervised)-*

I. *ML based*:

a. Extreme gradient boosting (XGBoost) - a decision-tree based ML algorithm, an optimization of gradient boosting

b. Support Vector Machines (SVM) - a supervised ML algorithm used for both classification and regression, we are using it for classification

II. *DL based:*

c. Multilayer perceptrons (MLP) - a class of feed-forward ANN, most basic deep neural networks

d. Long short-term memory (LSTM) - an artificial recurrent neural network (RNN), differs from MLP as it stores the last output in memory and uses that as input for successive step and hence capable of learning long-term dependencies

Word embedding + Topic modelling/classifier combinations

We have used the above mentioned word embeddings along with the topic modelling/classification techniques as follows:

1. Word2Vec embeddings with
a. MLP
b. LSTM
c. XGBoost
d. SVM

2. FastText embeddings with
a. MLP
b. LSTM
c. XGBoost
d. SVM

3. GloVe embeddings with

   a. MLP
   b. LSTM
   c. XGBoost
   d. SVM

4. TF-IDF vectors with MLP

5. LSA with Logistic Regression

6. LDA with Word2Vec (LDA2Vec)

7. BERTweet with MLP

ML vs DL

DL techniques like MLP and LSTM performed well but we wanted to use basic ML algorithms to see if similar results can be obtained with lesser resources and time.

Hence we explored SVM and XGBoost.

Experimental Setup

- Used Google Colaboratory
- Same hardware for all models, therefore uniform conditions for all models

Experimental Setup (contd.)

Word2Vec

Word2Vec	Sg = 0 (=> CBOW) Window size = 5 Size (Dimensions) = 100 Min_count = 1 Workers = 4

We tried with different window sizes like 3 and 7 and observed that 5 worked best of all that we tried.

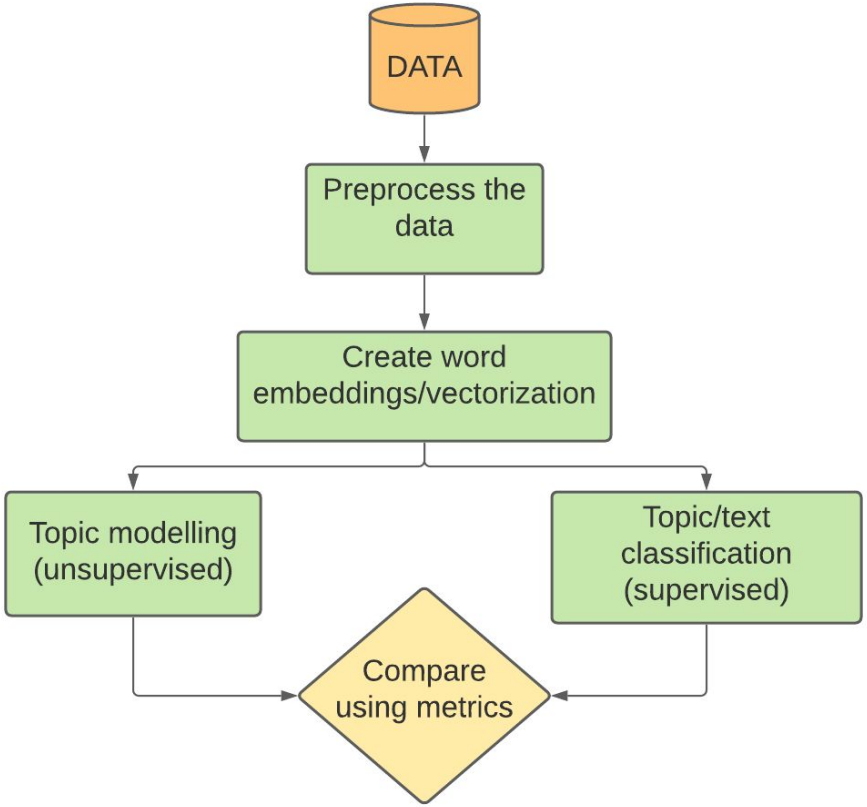
Experimental setup (contd.)

- For the neural networks that we used, we experimented with 1 and 2 hidden layers, but only 1 hidden layer was sufficient.
- For the neurons, we experimented with 2/3rd the number of neurons in the previous
  layer with a difference of around 10 neurons.
- For the activation function, we tried relu and tanh but relu gave better results.
- We used accuracy as the main metric for training.
- For the optimizer, we tried gradient descent, stochastic gradient descent and adam. Adam gave better results.
- For the loss function, we used binary cross entropy.

Experimental setup (contd.)

For the neural networks, we have done a validation split on our training data.

And for avoiding overfitting the training data, we have used early callback which will calculate validation loss at the end of each epoch and if the validation loss increases over a few epochs, it stops training.


DATA
Preprocess the data
Create word embeddings/vectorization
Topic modelling (unsupervised)
Topic/text classification (supervised)
Compare using metrics

EVALUATION METRICS:

The following metrics were used:

1. F1-score: F1-score was used for comparing the following models.

Word2Vec with MLP, Word2Vec with LSTM, Word2Vec with SVM, FastText with MLP, FastText with LSTM, FastText with SVM, Glove with MLP, Glove with LSTM Glove with SVM, TF-IDF with MLP, LSA with Logistic Regression

2. Cross-Validation score: Cross-validation score was used to evaluate the following models.

- Word2Vec with XGBoost

- FastText with XGBoost
- Glove with XGBoost

3. Coherence score: Coherence score was used to evaluate the LDA2Vec model.

4. Accuracy: Accuracy was used to evaluate the BERTweet model.

RESULTS:

- Out of all the models tested, BERTweet with Logistic regression gave the best accuracy.
- Out of all the topic classifiers (MLP, LSTM, SVM) used with the word embeddings (Word2Vec, FastText, GloVe), LSTM turned out to be the best classifier.
- XGBoost worked the best with GloVe as compared to Word2Vec and FastText.
- ●Supervised techniques like MLP, LSTM, XGBoost, SVM gave better results than unsupervised techniques like LDA and LSA

	F1-score
Word2Vec with MLP	0.778
Word2Vec with LSTM	0.794
Word2Vec with SVM	0.684
FastText with MLP	0.746
FastText with LSTM	0.794
FastText with SVM	0.694
GloVe with MLP	0.768
GloVe with LSTM	0.782
GloVe with SVM	0.81
TF-IDF with MLP	0.758

LSA (TF-IDF + SVD) + Logistic Regression	0.644

	Mean of Cross validation score
Word2Vec with XGBoost	0.6876
FastText with XGBoost	0.6864
GloVe with XGBoost	0.7989

	Coherence value
Word2Vec with LDA (LDA2Vec)	0.3096

	Accuracy
BERTweet with Logistic Regression	0.80656

CONCLUSION:

- In order to meet our objective we tried various classifiers and among those LDA gave the least accuracy.
- All models had comparable accuracy around 0.76 - 0.79 except BERTweet + MLP which gave the highest accuracy of around 0.80
- LSA, which is a dimensionality reduction technique on combining with the classifier, logistic regression, also gave results similar to LDA.
- XGBoost and SVM were not able to capture the semantic details behind the tweets, so even they gave less accuracy.
- After these, we experimented with two deep learning models, MLP and LSTM and their performances were almost the same and better than the previously used techniques.

- LSTM performed slightly better than MLP as it can detect the long term semantic relationship in tweets.
- Therefore, supervised techniques like LSTM, MLP performed better than unsupervised techniques like LDA and LSA.
- Out of all the models, using the transformer based model, BERTweet with MLP gave the highest accuracy. But it consumes a lot of time and GPU.

THANK YOU