Adcopy Predictor

Rithika Harish Kumar
rithikachordia7@gmail.com

August 23, 2018

Abstract

Ad copy predictor helps in predicting the performance of the Ad copy data. By performance, it means the number of clicks or conversions or conversion rate or click through rate. The goal of the project is to build a recommendation system for the user to help them decide the ad copy data, by comparing the performance(clicks) for any given data. The order of words in the sentence is also considered while predicting the number of clicks.

Keywords— Ad copy; prediction; clicks; sentence representation; regression

1 Introduction

The goal of an online advertisement is to sell a product or get a new lead or make a customer sign up or get the user to view or download on your site. All this is possible only if the customer clicks the Ad. The text of the ad copy is the second and third lines of an ad displayed on a search engine results page or any other web page and is between the title and the display URL. The ad copy is an excellent way not only to describe where the advertisement link leads, but also to make the advertisement seem as persuasive as possible to any visitors who might be interested, as they have already searched for a particular keyword that has brought the ad up. No matter how good your targeting is, if you've got the right audience but the wrong ad copy, is anyone really going to click on your ads? So, it is necessary that the advertiser choses the data wisely. This paper gives a solution to it by recommending or predicting the clicks for any given ad copy data based on the previous Ad performance data.

2 Approach

The block diagram in figure 1 explains the approach to our problem statement. The datasets are obtained from the Ad performance reports. Datasets are preprocessed to get the Ad copy data which is the concatenation of headline 1, headline 2 and description. Unique sentences are only considered. The clicks for the unique sentences are obtained by taking the mean of the clicks of the sentences. After preprocessing, the feature representations of the sentences are obtained by six different models to compare their performance and decide on the best one. The sentences representations and their labels(clicks) are trained by Random forest regression. The regression model is used to predict the number of clicks for any ad copy data.

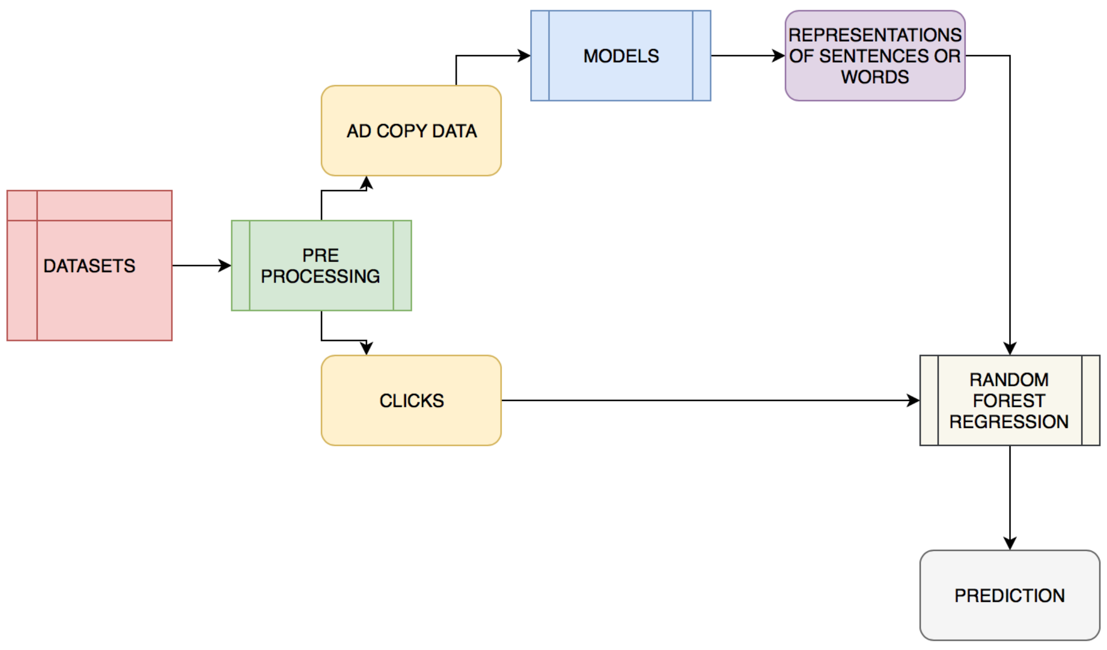


Figure 1: Block Diagram

3 Experiments

The various models used to obtain the features of the sentences are:

- Word2vec
- Bag of words (Tf-Idf)
- Google universal encoder
- Skip-thoughts
- Simple encoder-decoder
- InferSent

3.1 Word2vec

The pretrained word embeddings of the google Word2vec model is used. The model is loaded from the gensim package[4]. The pre-trained Google word2vec model was trained on Google news data, it contains 3 million words and was fit using 300-dimensional word vectors. Sentence is represented by the sum of the features of the words obtained from the Word2vec model.

3.1.1 Advantages

- Easy to load the model.
- Handles OOV words.
- Less complex and faster.

- Polyglot can be used to support many languages.

3.1.2 Disadvantages

- Does not consider the order of the words in the sentences.

3.2 Bag of words (Tf-Idf)

The TfidfVectorizer of the sklearn[5] feature extraction is used to extract the features of words in the sentences. Firstly, the term frequency of every word in the sentence is counted and tabulated across the unique words in the whole document. Then the log of total number of sentences to the number of sentences having the word is calculated and multiplied to the term frequency. Hence, the number of features for every sentence is the unique number of words present in the document.

3.2.1 Advantages

- Can support different languages.
- Less complex and faster.
- No need to handle OOV words.

3.2.2 Disadvantages

- Does not consider the order of the words in the sentences.

3.3 Google universal encoder

The module Google universal encoder 2[3] is used for obtaining features for the sentences. The model can be loaded from the tensor flow hub. The encoding model makes use of a deep averaging network (DAN) whereby input embeddings for words and bi-grams are first averaged together and then passed through a feed-forward deep neural network (DNN) to produce sentence embeddings. The encoder training data's sources are wikipedia, web news, web question answer pages and discussion forums. Unsupervised learning is augmented with training on supervised data from the Stanford Natural Language Inference (SNLI) corpus. The DAN encoder takes as input a lowercased PTB tokenized string and outputs a 512 dimensional sentence embedding.

3.3.1 Advantages

- Considers the order of the words in the sentence.

3.3.2 Disadvantages

- Takes time to load the model, but works fine after loading. Other models perform better on the basis of error rate.

3.4 Skip-thoughts

Skip thoughts are neural networks model for learning fixed length representations of sentences in any Natural Language without any labelled data or supervised learning. A pre-trained embedding is available for skip-thoughts model but it was trained on a book corpus. So, It cannot be used for our datasets as we have Ad copy data and there may be many OOV words. A model was created by combining all the unique sentences of the preprocessed data from every dataset and training on the skip-thoughts[1] network. 100 iterations were performed to obtain the encoder model. The number of features obtained is equal to the maximum length of the sentences considered when training. The encoder model was further used to extract sentence representations.

3.4.1 Advantages

- Considers the order of the words in the sentence.

3.4.2 Disadvantages

- Takes more time and cost to train comparatively.
- Performance was bad compared to other models.

3.5 Simple encoder-decoder

This is similar to the skip-thoughts model. But instead of creating a model and then performing transfer learning to encode, we directly predict[2] the features from the encoder after few epochs. Also, instead of Long short term memory (LSTM) as in skip-thoughts, Gated Recurrent unit (GRU) is used for faster performance. The maximum sentence length was set to 20 and before extracting the features the model was trained for 10 iterations. The sentence representations obtained form this model has 20 features.

3.5.1 Advantages

- Considers the order of the words in the sentence.

3.5.2 Disadvantages

- Takes more time compared to other models but is faster and better performing than that of skip-thoughts.

3.6 InferSent

InferSent[6] is a sentence embedding model provided by Facebook. Infersent version 2 is used here. SNLI corpus is trained on Bi-LSTM max pooling network with crawl word vectors to generate the Infersent encoder model. It provides 4096 features.

3.6.1 Advantages

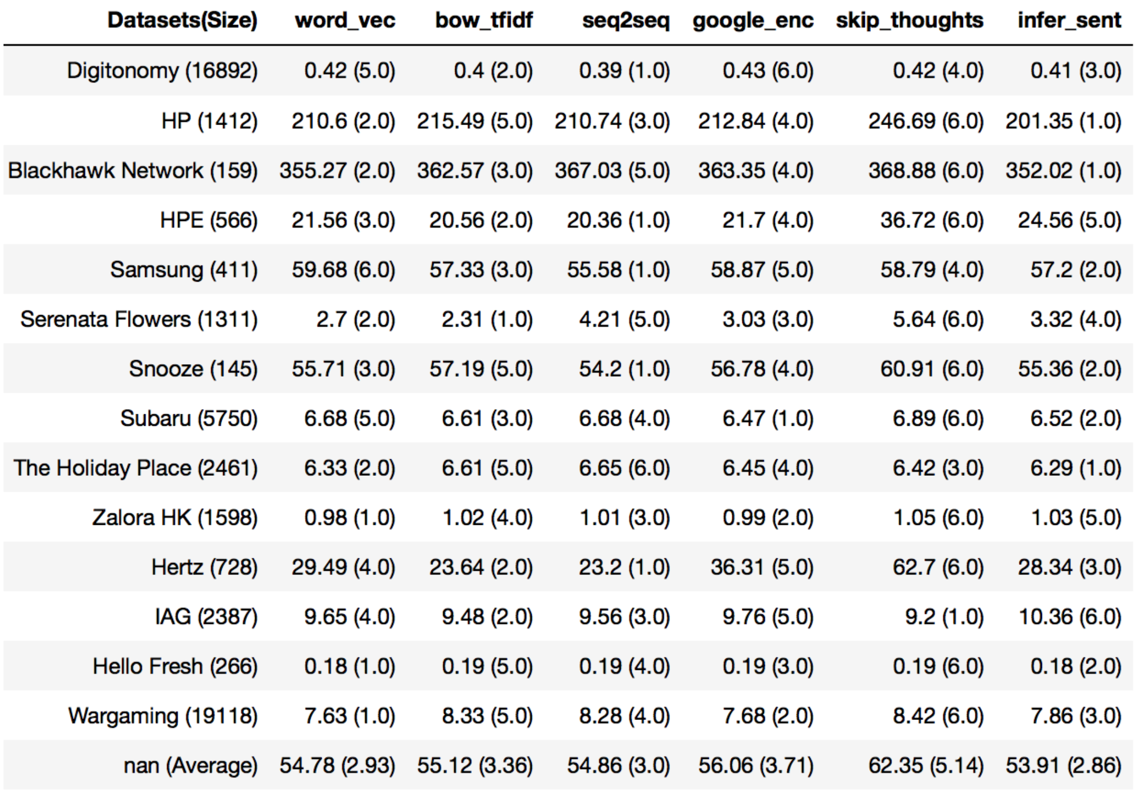
- Considers the order of words in the sentence.
- Performs comparatively better and has a lower error rate.

3.6.2 Disadvantages

- Does not support any other language except English. To support other languages, training for every language is to be done separately.

3.7 Evaluation

The performance of the model was determined using the mean absolute error and the mean square error. 5-fold cross validation was performed on every dataset to obtain the error scores in the figure 2. Ridge, XGB and Auto-sklearn regression was also tried on the worst performing dataset to check whether they performed better. But Random forest(RF) regression yielded better results. So, the whole evaluation and comparison was performed on RF. For regression the parameters were set to the following. The n_estimators is 500 , the max_features is sqrt and min_samples_leaf is 60. A tabulation of the error scores for every dataset was recorded against all the models. Friedman and Post-hoc tests was performed with a null hypothesis that all the models have similar performance. The alpha value is set to 0.05. The figure 2 shows the mean absolute error for every dataset



Datasets(Size)	word_vec	bow_tfidf	seq2seq	google_enc	skip_thoughts	infer_sent
Digitonomy (16892)	0.42 (5.0)	0.4 (2.0)	0.39 (1.0)	0.43 (6.0)	0.42 (4.0)	0.41 (3.0)
HP (1412)	210.6 (2.0)	215.49 (5.0)	210.74 (3.0)	212.84 (4.0)	246.69 (6.0)	201.35 (1.0)
Blackhawk Network (159)	355.27 (2.0)	362.57 (3.0)	367.03 (5.0)	363.35 (4.0)	368.88 (6.0)	352.02 (1.0)
HPE (566)	21.56 (3.0)	20.56 (2.0)	20.36 (1.0)	21.7 (4.0)	36.72 (6.0)	24.56 (5.0)
Samsung (411)	59.68 (6.0)	57.33 (3.0)	55.58 (1.0)	58.87 (5.0)	58.79 (4.0)	57.2 (2.0)
Serenata Flowers (1311)	2.7 (2.0)	2.31 (1.0)	4.21 (5.0)	3.03 (3.0)	5.64 (6.0)	3.32 (4.0)
Snooze (145)	55.71 (3.0)	57.19 (5.0)	54.2 (1.0)	56.78 (4.0)	60.91 (6.0)	55.36 (2.0)
Subaru (5750)	6.68 (5.0)	6.61 (3.0)	6.68 (4.0)	6.47 (1.0)	6.89 (6.0)	6.52 (2.0)
The Holiday Place (2461)	6.33 (2.0)	6.61 (5.0)	6.65 (6.0)	6.45 (4.0)	6.42 (3.0)	6.29 (1.0)
Zalora HK (1598)	0.98 (1.0)	1.02 (4.0)	1.01 (3.0)	0.99 (2.0)	1.05 (6.0)	1.03 (5.0)
Hertz (728)	29.49 (4.0)	23.64 (2.0)	23.2 (1.0)	36.31 (5.0)	62.7 (6.0)	28.34 (3.0)
IAG (2387)	9.65 (4.0)	9.48 (2.0)	9.56 (3.0)	9.76 (5.0)	9.2 (1.0)	10.36 (6.0)
Hello Fresh (266)	0.18 (1.0)	0.19 (5.0)	0.19 (4.0)	0.19 (3.0)	0.19 (6.0)	0.18 (2.0)
Wargaming (19118)	7.63 (1.0)	8.33 (5.0)	8.28 (4.0)	7.68 (2.0)	8.42 (6.0)	7.86 (3.0)
nan (Average)	54.78 (2.93)	55.12 (3.36)	54.86 (3.0)	56.06 (3.71)	62.35 (5.14)	53.91 (2.86)

Figure 2: MAE Tabulation

corresponding to every model. The ranks are mentioned within brackets. The number of unique sentences for every dataset is mentioned within brackets near the name of the dataset. The last row corresponds to the average of the error score and the ranking. The Freidman score is 15.08 and the p-value is 0.01. From the p-value it is evident that the null hypothesis is rejected. So, all algorithms do not have same performance and at least one of them behaves differently. The figure 3 is a graph with different models along x-axis and their ranking along y-axis. The figure 4 is a tabulation of the post-hoc

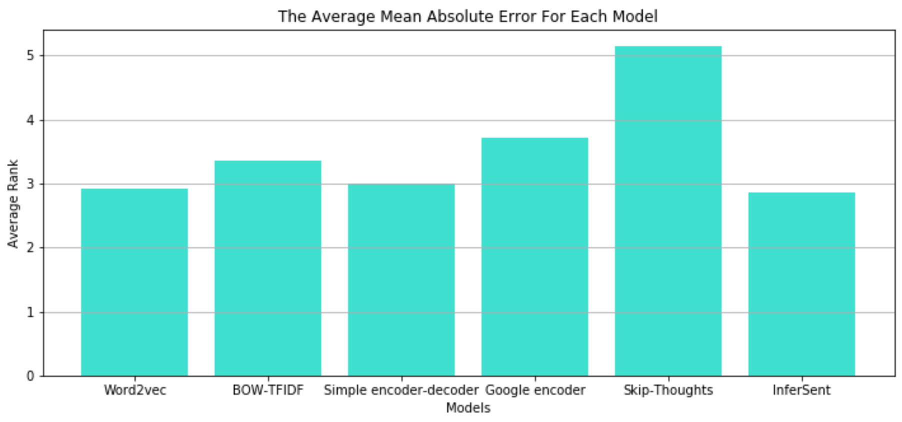
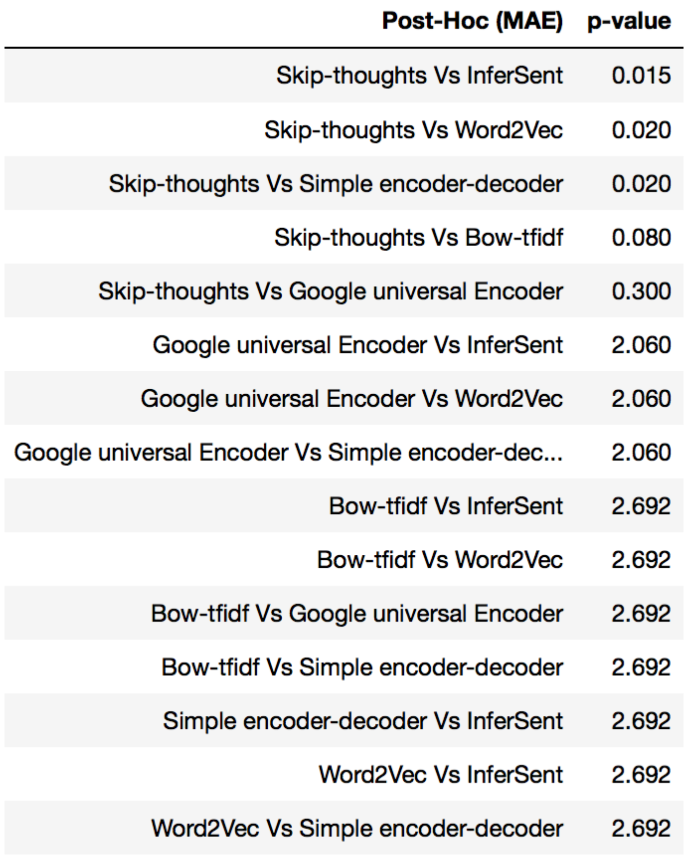


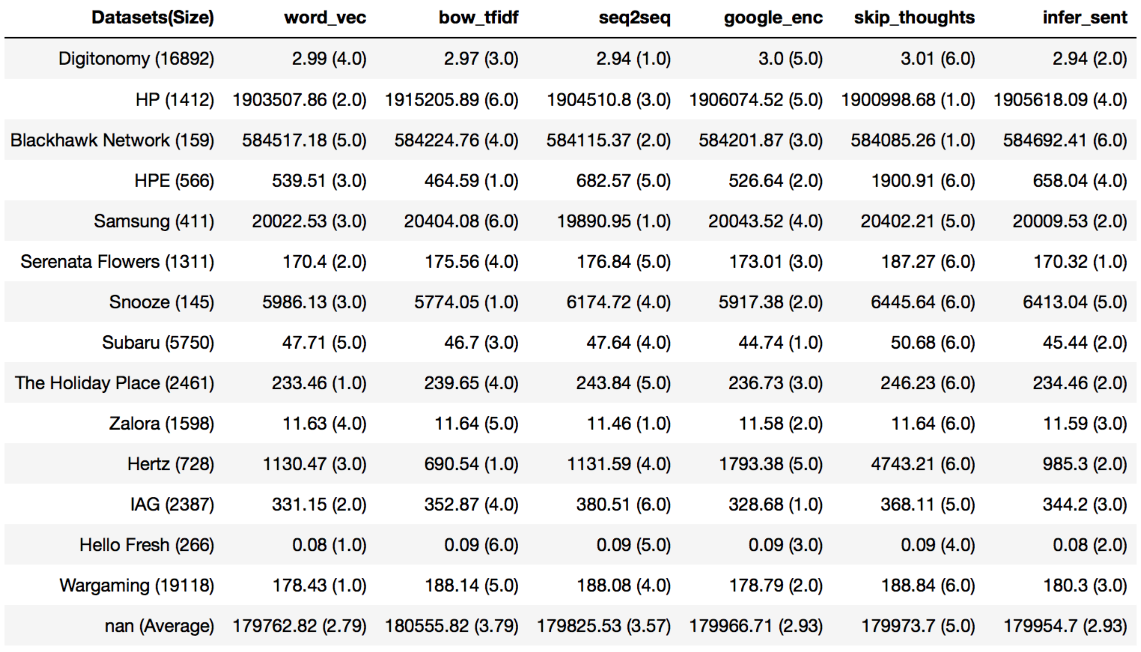
Figure 3: MAE RankGraph



Post-Hoc (MAE)	p-value
Skip-thoughts Vs InferSent	0.015
Skip-thoughts Vs Word2Vec	0.020
Skip-thoughts Vs Simple encoder-decoder	0.020
Skip-thoughts Vs Bow-tfidf	0.080
Skip-thoughts Vs Google universal Encoder	0.300
Google universal Encoder Vs InferSent	2.060
Google universal Encoder Vs Word2Vec	2.060
Google universal Encoder Vs Simple encoder-dec...	2.060
Bow-tfidf Vs InferSent	2.692
Bow-tfidf Vs Word2Vec	2.692
Bow-tfidf Vs Google universal Encoder	2.692
Bow-tfidf Vs Simple encoder-decoder	2.692
Simple encoder-decoder Vs InferSent	2.692
Word2Vec Vs InferSent	2.692
Word2Vec Vs Simple encoder-decoder	2.692

Figure 4: MAE Post-Hoc

analysis of the models with their p-values. It can be inferred that InferSent and Word2vec perform comparatively better. Simple encoder-decoder and Word2vec do not have much of a difference. The simple encoder-decoder can be used to support other languages. The figure 5 shows the mean square error for every dataset corresponding to every model. The Freidman score is 13.92 and the p-value is 0.016. From the p-value it is evident that the null hypothesis is rejected. So, all algorithms do not have same performance and at least one of them behaves differently. The figure 6 is a graph with different models along x-axis



Datasets(Size)	word_vec	bow_tfidf	seq2seq	google_enc	skip_thoughts	infer_sent
Digitonomy (16892)	2.99 (4.0)	2.97 (3.0)	2.94 (1.0)	3.0 (5.0)	3.01 (6.0)	2.94 (2.0)
HP (1412)	1903507.86 (2.0)	1915205.89 (6.0)	1904510.8 (3.0)	1906074.52 (5.0)	1900998.68 (1.0)	1905618.09 (4.0)
Blackhawk Network (159)	584517.18 (5.0)	584224.76 (4.0)	584115.37 (2.0)	584201.87 (3.0)	584085.26 (1.0)	584692.41 (6.0)
HPE (566)	539.51 (3.0)	464.59 (1.0)	682.57 (5.0)	526.64 (2.0)	1900.91 (6.0)	658.04 (4.0)
Samsung (411)	20022.53 (3.0)	20404.08 (6.0)	19890.95 (1.0)	20043.52 (4.0)	20402.21 (5.0)	20009.53 (2.0)
Serenata Flowers (1311)	170.4 (2.0)	175.56 (4.0)	176.84 (5.0)	173.01 (3.0)	187.27 (6.0)	170.32 (1.0)
Snooze (145)	5986.13 (3.0)	5774.05 (1.0)	6174.72 (4.0)	5917.38 (2.0)	6445.64 (6.0)	6413.04 (5.0)
Subaru (5750)	47.71 (5.0)	46.7 (3.0)	47.64 (4.0)	44.74 (1.0)	50.68 (6.0)	45.44 (2.0)
The Holiday Place (2461)	233.46 (1.0)	239.65 (4.0)	243.84 (5.0)	236.73 (3.0)	246.23 (6.0)	234.46 (2.0)
Zalora (1598)	11.63 (4.0)	11.64 (5.0)	11.46 (1.0)	11.58 (2.0)	11.64 (6.0)	11.59 (3.0)
Hertz (728)	1130.47 (3.0)	690.54 (1.0)	1131.59 (4.0)	1793.38 (5.0)	4743.21 (6.0)	985.3 (2.0)
IAG (2387)	331.15 (2.0)	352.87 (4.0)	380.51 (6.0)	328.68 (1.0)	368.11 (5.0)	344.2 (3.0)
Hello Fresh (266)	0.08 (1.0)	0.09 (6.0)	0.09 (5.0)	0.09 (3.0)	0.09 (4.0)	0.08 (2.0)
Wargaming (19118)	178.43 (1.0)	188.14 (5.0)	188.08 (4.0)	178.79 (2.0)	188.84 (6.0)	180.3 (3.0)
nan (Average)	179762.82 (2.79)	180555.82 (3.79)	179825.53 (3.57)	179966.71 (2.93)	179973.7 (5.0)	179954.7 (2.93)

Figure 5: MSE Tabulation

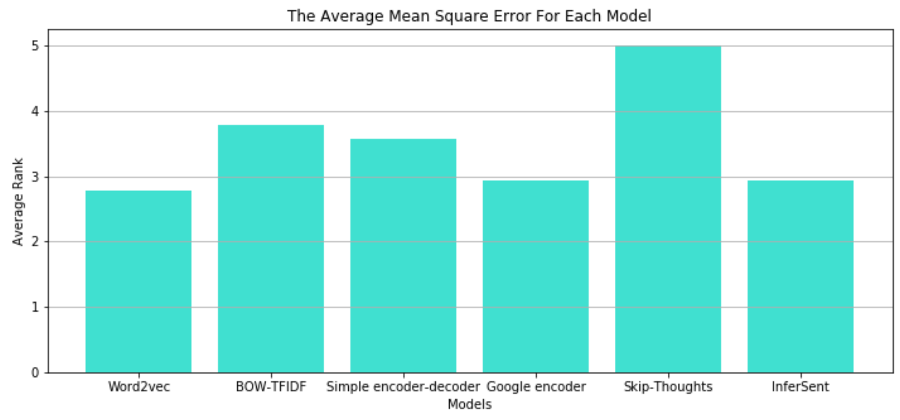
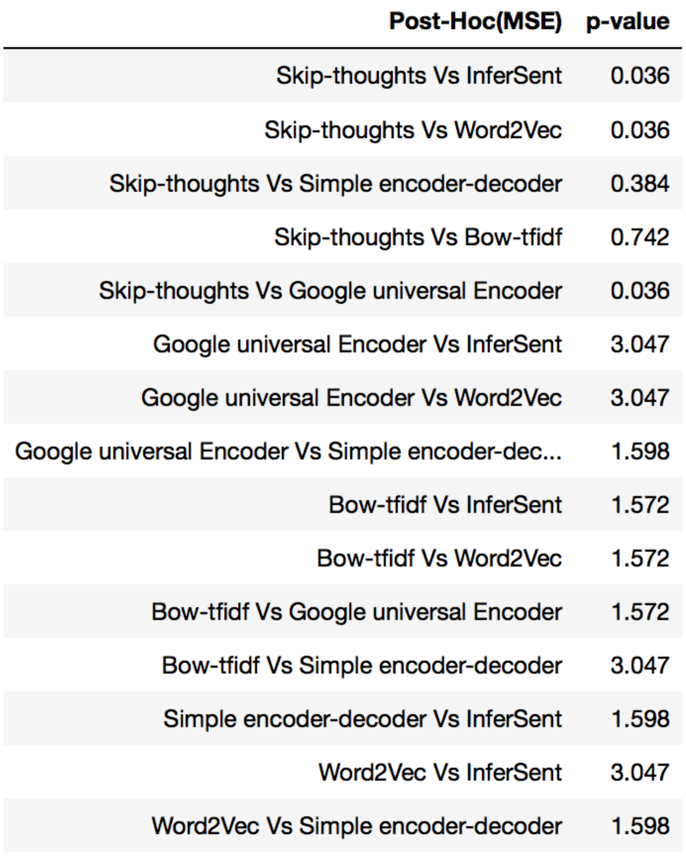


Figure 6: MSE RankGraph

and their ranking along y-axis. The figure 7 is a tabulation of the post-hoc analysis of the models with their p-values. It can be inferred that InferSent and Word2vec perform comparatively better. For both the error scores Skip-thoughts perform the worst. As far as the exact error scores are concerned, the regression model parameters can be adjusted for better performance. The above parameters were kept constant for every model because the baseline was to compare the performance of the various encoders and choose the better one.

4 Results

The major focus area of the project is to consider the order of words in the sentences of the Ad copy data and to evaluate the performance of different sentence representation models. Among the four models which considered the order of the sentences, InferSent performs



Post-Hoc(MSE)	p-value
Skip-thoughts Vs InferSent	0.036
Skip-thoughts Vs Word2Vec	0.036
Skip-thoughts Vs Simple encoder-decoder	0.384
Skip-thoughts Vs Bow-tfidf	0.742
Skip-thoughts Vs Google universal Encoder	0.036
Google universal Encoder Vs InferSent	3.047
Google universal Encoder Vs Word2Vec	3.047
Google universal Encoder Vs Simple encoder-dec...	1.598
Bow-tfidf Vs InferSent	1.572
Bow-tfidf Vs Word2Vec	1.572
Bow-tfidf Vs Google universal Encoder	1.572
Bow-tfidf Vs Simple encoder-decoder	3.047
Simple encoder-decoder Vs InferSent	1.598
Word2Vec Vs InferSent	3.047
Word2Vec Vs Simple encoder-decoder	1.598

Figure 7: MSE Post-Hoc

comparatively better. But it supports only the English language. So, to support other languages training can be performed in order to get the particular language model using InferSent. It is observed that the mean square error is large especially for datasets like HP and Blackhawk Network. It is maybe because of the presence of outliers due to the variance in the number of clicks. To check that, regression was tried to predict conversions, conversion rate and click through rate. The url https://docs.google.com/spreadsheets/d/10VE437vG26NhCCvv0tjWZy9G6wnIwtNTruw9o9B05nI/edit?usp=sharing leads to the predicted results of serenata dataset. All the predictions were obtained by using the Infersent model with Random Forest Regression. The MSE and MAE were negligible when conversion rates and click through rate was considered, but was a little high for conversions.

5 Conclusion

A total of six models were tried to evaluate better sentence representations for Ad copy data. Among which InferSent seemed acceptable. The results are no way conclusive as the study was performed only with reference to the number of clicks. The results would have been more better if conversion rate was considered as the labels for the sentences instead of clicks. It is because the clicks show high variance and it is more prone to fall under the outlier effect. This is also the reason why mean absolute error was considered for results substantiation than mean square error.

A Git hub Guide

The code for the above project can be found on the link https://github.com/campanja/summer-internship/tree/ritzie96. The "combined4models.ipynb" notebook is used to predict results for four models which are: Bag of words (TF-IDF), Word2Vec, Google Universal Encoder and InferSent. The models.py has the definition for the Infersent model. The "deep_framework.ipynb" notebook is used to predict results for the simple encoder-decoder model. The skipthought folder has "train.py" file is used train the corpus to get the encoder model. The notebook "skip-thoughts.ipynb" is used to load the skip-thought encoder model and predict results. The RF_results folder has tabulations of all the error measures on every dataset for every model. All results of these tables are based on Random forest regression. The "mae_friedman_test.ipynb" has the results for the freidman test performed on the mean absolute error values of every dataset against every model. The "mse_friedman.ipynb" has the results for the freidman test performed on the mean square error values of every dataset against every model. The plots folder has the plots obtained from freidman and post-hoc tests. The folder Eval_rep has the reports of the post-hoc tests. The folder serenata_test has the predicted values for unseen data. The predicted results for one month of dataset being trained. And the results for four month of dataset being trained are included.

References

[1] Sanyam Agarwal. 'An implementation of Skip-Thought Vectors in PyTorch'. *Git repository hosted at GitHub.* https://github.com/sanyam5/skip-thoughts

[2] Yonatan Hadar. 'Unsupervised sentence representation with deep learning'. https://blog.myyellowroad.com/unsupervised-sentence-representation-with-deep-learning-104b90079a93

[3] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, Ray Kurzweil. Universal Sentence Encoder 2 https://arxiv.org/abs/1803.11175

[4] Word2Vec https://radimrehurek.com/gensim/models/word2vec.html

[5] Bag of words - TFIDF http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[6] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, A. Bordes Supervised Learning of Universal Sentence Representations from Natural Language Inference Data https://arxiv.org/abs/1705.02364