CSE 517 Course Project Final Report

Regina Cheng
rcheng6@uw.edu

Joyce Zhou
jyzhou15@uw.edu

1 Introduction

In this report we replicated a 2019 workshop paper (Wang et al., 2019) that presents a probabilistic clustering algorithm based on the BERT Next Sentence Prediction model. The paper claims its algorithm is effective for clustering Reddit posts that indicate similar author interests. Following the original paper, we collected 8800 Reddit posts and compared the performance of generating 30 clusters using BERT (Devlin et al., 2018) based clustering algorithm compared to two baseline algorithms: Word2Vec embedding (W2V) (Mikolov et al., 2013) and Latent Dirichlet allocation (LDA) topic modelling (Blei et al., 2003). In addition to the original experiments, we also tested the effect of smaller sample sizes on the three different clustering methods, as well as the impact of different cluster numbers on the three algorithms.

We now present an overview of the pipeline and results of our experiments. We first collected posts and metadata from the r/Advice subreddit from Reddit, following the original paper's data inclusion criteria. Then we passed the data to the three methods (BERT based probabilistic clustering algorithm, Word2Vec and LDA) and generated similarity tables to be used for clustering. Next, we clustered the posts using the similarity tables generated by each model. Lastly, we calculated Same Author Score (SAS) and Jaccard Score to evaluate cluster quality. We implemented every step in this pipeline following the paper's instruction. We report the evaluation scores for each experiment, as well as the run time of every step in the pipeline.

2 Contributions

In addition to replicating the experiment in the original paper, in this report we also present three contributions concerning aspects that were underdiscussed in the original paper: 1) we show the robustness of the BERT based clustering algorithms on smaller sample sizes; 2) we show the robustness of the BERT based clustering algorithms on different cluster numbers; 3) we report the computational expense of the entire clustering pipeline. Altogether, we discuss the feasibility for future researchers of using the BERT-based clustering algorithm in the task of identifying similar online text-based productions.

2.1 Hypotheses from original paper

The original paper does not provide a clear hypothesis on their experiments. They compared the clustering performance on similar Reddit posts using a BERT-based probabilistic clustering algorithm with clustering results generated by traditional embedding methods Latent Dirichlet Allocation (LDA) and Word2Vec (W2V). We clarify the apparent hypothesis from their paper here.

- **Original H0**: Clusters formed with BERT-based clustering algorithm score at least as well (by SAS and Jaccard scoring) as those clusters formed by W2V-based and LDA-based clustering algorithms.

2.2 Hypotheses addressed in this work

- **H1**: With a smaller sample size, BERT-based clustering is not as robust when compared to LDA-based clustering or Word2Vec-based clustering.
- **H2**: With a smaller cluster size (i.e. more clusters), BERT-based clustering is not as robust when compared to LDA-based clustering or Word2Vec-based clustering.

2.3 Experiments

- Experiment 0 (for Original H0): We replicated the experiments in the original paper by comparing the performance of BERT-based,

W2V-based and LDA-based clustering algorithms on 8800 sampled posts.

- Experiment 1 (for H1): How robust are the models to sample size? In the original paper, the authors used a sample size of 8865. Especially because BERT for Next Sentence Prediction and the following similarity clustering algorithms are computationally expensive, we would like to explore whether its high performance still holds for smaller sample sizes.

  In this experiment, we replicated the experiment in the original paper, except using smaller sample sizes. Specifically, we compared the results (Same Author Scores and Jaccard Scores) generated from 1000 posts, 2200 posts (25% of original sample size) and 4400 posts (50% of original sample size). We would like to see if BERT embedding and clustering is robust (still performs better than Word2Vec and LDA) with less data.

- Experiment 2 (for H2): How robust are the models to cluster count? In the original paper, the authors did not discuss how cluster count may affect the model performance. In fact, they did not report how the number of clusters, which is m = 30, was decided in their experiments. We see this as an opportunity to add clarity to the original paper, as well as exploring how different models may respond differently to cluster numbers.

  In this experiment, we used the original size, 8800 posts as our sample and adjusted the number of clusters produced by our clustering algorithm. We compared the results (Same Author Scores and Jaccard Scores respectively) for different cluster numbers (2 to 100, incrementing by 10). We plot the results and cluster number for BERT, Word2Vec and LDA.

3 Code

We implemented the entire pipeline consisting of data collection, similarity matrix generation for each of the three methods, clustering, evaluation of clustering, and the experiments.

Code is available at:
https://github.com/cephcyn/cse517project.

The experiment can be run with the script `run.sh`, with sample size and cluster numbers adjustable with variables located at the beginning of the script.

4 Experimental setup and results

4.1 Model description

4.1.1 BERT-Based Probabilistic Clustering

The Bert Based Probabilistic Clustering algorithm consists of two parts: a BERT part, and a similarity calculation part. The output of the algorithm is a similarity table ready to use for clustering.

As guided by the paper, we used the pretrained BERT Next Sentence Prediction model (uncased base) implemented by HuggingFace Transformers. [1]. The model takes in two sequences, and outputs the probability that the first is followed by the second. In the paper, the authors argue that this probability can also be interpreted as similarity between two Reddit posts. Specifically in their case, they use BERT to generate the probability of the title from post A followed by the text of post B, and then deduce the similarity of post A and post B.

The most intuitive way is to construct a $n * n$ dimensional table (where n is the sample size) of probability of all pairs of posts followed by each other. This requires running BERT n^2 times, which is very expensive. As a preliminary test, we experimented with generating the similarity table using BERT directly. It turns out that generating a 1*1000 array of similarities (the likelihood of title A followed by each of the n posts, $n = 1000$) took 5540 seconds, which means generating the full 1000*1000 similarity matrix would take 1000 times longer. Therefore, it is not feasible to use this method.

The authors in the paper present an alternative way of generating the $n * n$ similarity table, which consists of two algorithms they devised (Algorithm 1 and Algorithm 2 in p309 - p310 of the paper). The explanation of the mechanism of the similarity generation algorithms can be found in Section 3.2 and 3.3 in the original paper. We implemented this algorithm as described by the paper. For this algorithm, we used the same hyperparameters ($p = 5$, $m = 30$) as in the paper, independent of our other experiment parameters.

In the paper, the authors also finetuned the

[1] https://github.com/huggingface/transformers

BERT model using over 300,000 posts from subreddits that are similar to r/Advice. Although we tried to replicate this finetuning step, we failed to do so due to our limited hardware computation capacity. We constructed a labelled dataset for finetuning which contains 82,406 text-text / title-title / text-title pairs, labelled with binary tags "from the same author" or "not from the same author". We have also implemented the finetuning code. Both the dataset and the finetuning code are ready to use and available on our repository. However, we always got insufficient GPU memory error with our GTX 970 GPU when finetuning. Therefore, the experiments in this report describe our BERT results generated without finetuning.

Implementation of BERT-related components took 20 hours, including implementation of finetuning and overall debugging time.

4.1.2 Word2Vec

We implemented the Word2Vec embedding algorithm as described in the original paper. For each post, we first tokenized and lemmatized each word in the post, removing English stop words, numbers and punctuation. Then we used the 300-dimension embedding from the Word2Vec model from the Gensim library [2]. We skipped the words in the posts that were not in the Word2Vec model. Next, following the paper, we implemented two variants of Word2Vec embedding:

Word2Vec Weighted: We assigned weights to each word (all 300 dimensions) based on the inverse of its relative frequency in the sentence. The embedding of a sentence was calculated by summing up the weighted embeddings of all words, with a dimension of 1*300.

Implementing this algorithm took 4 hours, including debugging time.

Word2Vec SIF: Using the embeddings we obtained from Word2Vec-Weighted, we performed Principal Component Analysis on the embeddings to obtain a transformation of the embeddings and a list of principal components. We first calculated the projection of each embedding onto the first principal component. Next, we subtracted that projected vector from each embedding to obtain a new set of post embeddings with the first principal component removed.

Implementing this algorithm took approximately 4 hours, including debugging time.

4.1.3 LDA

We implemented the LDA topic modeling algorithm as described in the original paper. For each post, we did the same preprocessing as we did for Word2Vec (tokenize, remove stop words, remove numbers and punctuation, and lemmatize). Following the paper, we implemented two variants of LDA-based embeddings:

LDA BoW: We computed a bag of words (BoW) representation of each document. We then used the LDA implementation in Gensim [3] to compute 30 topic clusters for the entire document set. We used 30 topic clusters for each document because that is the parameter that was used in the original paper. The embedding of each document was calculated by computing its topic distribution, generating a 1*30 sparse vector.

LDA TF-IDF: Using the bag of words representation we previously computed, we computed a term frequency-inverse document frequency (TF-IDF) representation of each document. The embedding of each document was then calculated in the same way as we did for LDA BoW, except using the TF-IDF word counts instead.

4.1.4 Generating Similarity Matrices and Clustering

Similarity Matrices: For Word2Vec and LDA, we calculated the $n * n$ similarity matrices (where n is the sample size) using cosine similarities between embeddings for each pair of posts as described in the original paper. The output of the BERT-based algorithm is a $n * n$ similarity matrix that is ready to use for clustering, so we did not need to do extra computation to generate this matrix.

Another side note: In the paper the authors mentioned generating similarity matrices using title-post pairs for Word2Vec and LDA. However, it is only vaguely discussed without any implementation details, and we don't understand how title-text similarity matrix for W2V and LDA models makes sense to implement, so we only implemented post-post similarity matrix.

Clustering: Following Algorithm 3 in p310 of the paper, we implemented the clustering algorithm. The algorithm takes in a $n * n$ similarity table generated by BERT, W2V or LDA, an integer n as the sample size, and another integer m as the number of output clusters. The algorithm

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

[3] https://radimrehurek.com/gensim/models/ldamodel.html

works as it randomly selects a post in the sample, find posts that are most similar to it based on the similarity matrix, then put them in a cluster. Next the algorithm randomly select another post in the rest unclustered posts then repeat until there is no unclustered posts left. The algorithm outputs a dictionary that contains key as the cluster id and values as a list of post ids in that cluster. The original paper lacks a description on the size of the clusters, our algorithm outputs most clusters with the same size = $\texttt{ceil}(\frac{n}{m})$ and a smaller last cluster (size = $n - \texttt{ceil}(\frac{n}{m}) * (m-1)$).

Implementing the algorithms to generate similarity matrix and perform clustering took us approximately 10 hours, including debugging time.

4.2 Datasets

On February 5th 2020, we collected the most recent 465,000 posts from r/Advice using Pushshift API [4]. Following the data inclusion criteria in the original paper, we removed posts 1) with a score lower than 3 and 2) without textual information in the body of the posts. This resulted in 39,440 posts that were usable for our experiments. Each data point contains a post id as the unique identifier, the post author id, the post title and the post text.

On March 12th 2020, for each of the Reddit users who have authored a post in our dataset, we also collected a list of subreddits that they have submitted to or commented in using Pushshift API and PRAW [5].

For both of these types of datasets, the scripts for scraping and preprocessing the data are available in our GitHub repository. The datasets we obtained is also available to download as part of our GitHub repository.

4.3 Evaluation metrics

The original paper used two different scoring methods to evaluate the quality of clusters generated by BERT-based clustering and baseline algorithms. We implemented each scoring method according to its descriptions.

4.3.1 Same Author Score (SAS)

The Same Author Score represents the proportion of posts with same authors being clustered into the same cluster. We rephrase its description from the original paper here.

$$S_{SAS} = \frac{\sum_{i=1}^{I} \sum_{j}^{J_i} \delta_{\textit{clust}(j_0),\textit{clust}(j_1)}}{\sum_{i=1}^{I} J_i} - \frac{1}{m}$$

In this equation:

- I is the total number of authors.
- j is a pair of posts by the same author. j_0 and j_1 are the elements of this pair. J_i is the number of all possible such pairings for author i.
- *clust()* is a function that returns the cluster number of a post.
- $\delta_{i,j}$ is a function where $\delta = 1$ if $i = j$, otherwise $\delta = 0$.
- $\frac{1}{m}$ is the constant representing the likelihood of two posts being in the same cluster by chance.

4.3.2 Jaccard Scoring

The Jaccard Score measures similarity of interests between authors of posts in the same cluster. We use this equation to compute Jaccard score (note that it is slightly different from what the original paper used, due to reasons described below):

$$S_{Jaccard} = \sum_{c=1}^{C} \sum_{j=1}^{J_c} \left(\frac{|L_{j_0} \cap L_{j_1}|}{|L_{j_0} \cup L_{j_1}|}\right) * \frac{C}{P^2}$$

In this equation:

- C is the total number of clusters.
- j is a pair of posts in the same cluster. j_0 and j_1 are the elements of this pair. J_c is the number of all possible such pairings from cluster c.
- L_i is the set of subreddits that the author of post i has contributed to (either posted or commented in).
- P is the total number of posts.

Our Adjustments

The original paper does not clarify on several of the edge cases related to Jaccard score, so we elaborated on these edge cases in our implementation of Jaccard score.

[4] https://github.com/pushshift/api

[5] https://github.com/praw-dev/praw

What happens when authors no longer exist (i.e. they have deleted their accounts) and we cannot get a list of their participated subreddits? We decided to assume that authors who no longer exist are the same as throwaway accounts, so we treat them as if they have only participated in one subreddit. Since we obtained posts from r/Advice, we assume that deleted accounts have only participated in r/Advice.

Furthermore, what if authors have only ever posted submissions to subreddits and have never commented? The original paper's Jaccard scoring formula is flawed because it fails for authors with no comments. To remedy this, we modified the formula to be based on the union and intersection of subreddits that a user has ever *interacted* with, instead of distinguishing between comments and submissions. This modification preserves the core concept of the original formula.

4.4 Hyperparameters

The hyperparameters in the paper across a generic experiment include:

- Size of the Reddit posts sample
- Number of clusters formed
- Number of times we perform a clustering/scoring loop to compute the average score for each clustering method

In our experiments, we tested different values for the first two parameters.

Model-specific hyperparameters include:

- Dimension of Word2Vec embeddings (always 300)
- Number of topics for LDA (always 30)
- Chunk size for LDA (always 100)
- Number of passes for LDA (always 20)
- Number of iterations for LDA (always 400)
- BERT hyperparameters (m = 30, p = 5), explanations in Section 3.2 and 3.3 in the original paper

4.5 Results

In Experiment 0 (E0), we replicated the experiment in the original paper, that is, comparing the SAS and Jaccard scores of different models (explained in Section 4.1). We ran the models on the same 8800 posts (sample size $n = 8800$) and generated 30 clusters ($m = 30$) from each model. We reported the time (in seconds) for generating the similarity matrices using each model in Table 1. We repeated for 100 times as described in the original paper. We then reported the mean, range and standard deviation of the resulting SAS and Jaccard scores in Table 1 and the comparison of SAS in Figure 1, Jaccard in Figure 2.

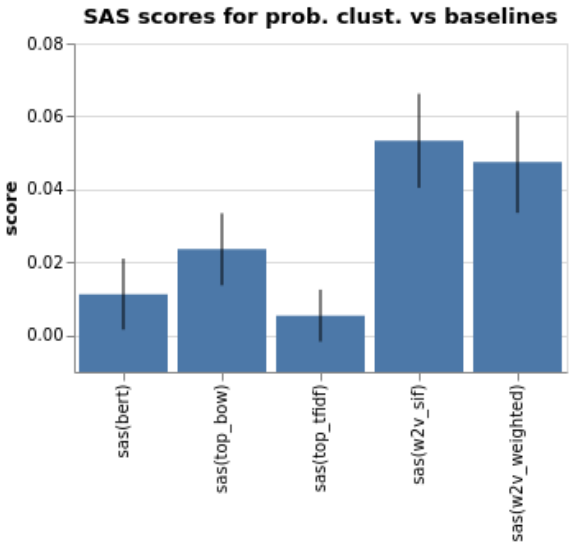

Figure 1: SAS score comparison between the models for 8800 posts and 30 clusters

We found that the BERT-based clustering (unsurprisingly) took a lot longer time to generate the similarity table. Contradictory to what was described in the original paper, BERT generally produces worse clustering results in average than the baseline models. Based on the SAS score, BERT is only a little better than LDA-TFIDF and is worse than W2V-based clusterings. Based on Jaccard scoring, all models produced results that only differ in small scales, with BERT performing slightly lower than the baseline models. Therefore, our results from E0 does not support H0. This deviation from the results of the original paper is likely due to the lack of finetuning BERT in our study.

5 Experiments beyond the original paper

5.1 Experiment 1

In Experiment 1 (E1), we extended the original experiment by exploring the effect of different sample sizes on the models, with the purpose of examining the robustness of the models to smaller samples. We used the sample size $n = 1000$ (12%

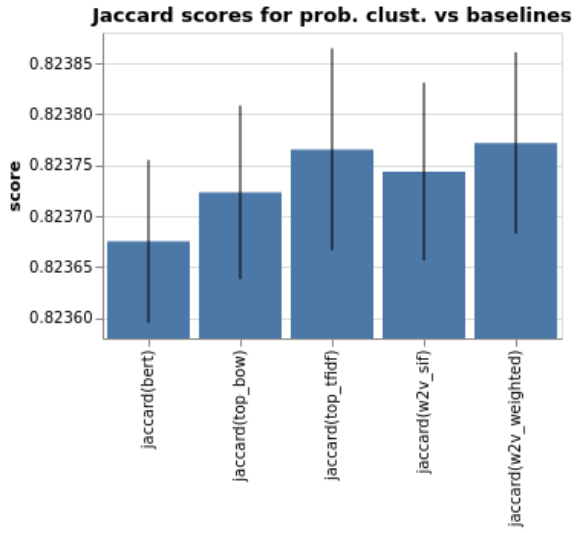

Figure 2: Jaccard score comparison between the models for 8800 posts and 30 clusters

E0 (n = 8800, n = 30)

Model	SAS
BERT	0.011 [-0.014, 0.032] (0.010)
W2V-W	0.047 [0.021, 0.099] (0.014)
W2V-SIF	0.053 [0.026, 0.080] (0.013)
LDA-BOW	0.024 [0.000, 0.051] (0.010)
LDA-TFIDF	0.005 [-0.01, 0.02] (0.007)

Model	Jaccard
BERT	0.824 [0.823, 0.824] (0.000)
W2V-W	0.824 [0.824, 0.824] (0.000)
W2V-SIF	0.824 [0.824, 0.824] (0.000)
LDA-BOW	0.824 [0.824, 0.824] (0.000)
LDA-TFIDF	0.824 [0.824, 0.824] (0.000)

Table 1: Time (in seconds) used for embedding, SAS and Jaccard for different models. (Mean [range] (sd))

E1 (1000, 30)

Model	SAS
BERT	0.067 [-0.033, 0.395] (0.108)
W2V-W	0.202 [-0.033, 0.538] (0.126)
W2V-SIF	0.527 [-0.033, 0.967] (0.312)
LDA-BOW	0.265 [-0.033, 0.824] (0.161)
LDA-TFIDF	-0.016 [-0.033, 0.110] (0.045)

Model	Jaccard
BERT	0.070 [0.069, 0.070] (0.000)
W2V-W	0.070 [0.070, 0.071] (0.000)
W2V-SIF	0.070 [0.069, 0.071] (0.000)
LDA-BOW	0.070 [0.069, 0.071] (0.000)
LDA-TFIDF	0.070 [0.069, 0.071] (0.000)

Table 2: Time (in seconds) used for embedding, SAS and Jaccard for different models. (Mean [range] (sd))

E1 (2200, 30)

Model	SAS
BERT	0.007 [-0.033, 0.118] (0.033)
W2V-W	0.049 [-0.033, 0.148] (0.044)
W2V-SIF	0.061 [-0.033, 0.179] (0.044)
LDA-BOW	0.011 [-0.033, 0.088] (0.034)
LDA-TFIDF	0.007 [-0.033, 0.088] (0.029)

Model	Jaccard
BERT	0.053 [0.053, 0.054] (0.000)
W2V-W	0.054 [0.053, 0.054] (0.000)
W2V-SIF	0.054 [0.053, 0.054] (0.000)
LDA-BOW	0.053 [0.053, 0.054] (0.000)
LDA-TFIDF	0.053 [0.053, 0.054] (0.000)

Table 3: Time (in seconds) used for embedding, SAS and Jaccard for different models. (Mean [range] (sd))

of original sample size), $n = 2200$ (25%) and $n = 4400$ (50%), and ran the models on the different sizes respectively. In this experiment, we held the number of clusters constant so $m = 30$. For each sample size, we reported the time for generating the similarity matrices using each model, and the resulting SAS and Jaccard in Table 2 & 3.

We found that with smaller sample sizes, BERT is still consistently out-performed by the baseline models. This may again be caused by that fact that our BERT model is not finetuned. Furthermore, SAS scoring of BERT shows a lot of variability across different sample sizes. This supports our H1 that BERT is not as robust to smaller sample sizes.

E1 (4400, 30)

Model	SAS
BERT	0.028 [-0.016, 0.089] (0.022)
W2V-W	0.05 [0.002, 0.116] (0.022)
W2V-SIF	0.062 [-0.007, 0.116] (0.024)
LDA-BOW	0.032 [-0.016, 0.089] (0.021)
LDA-TFIDF	0.009 [-0.033, 0.046] (0.015)

Model	Jaccard
BERT	0.222 [0.222, 0.223] (0.001)
W2V-W	0.222 [0.222, 0.224] (0.000)
W2V-SIF	0.223 [0.222, 0.224] (0.000)
LDA-BOW	0.222 [0.222, 0.223] (0.000)
LDA-TFIDF	0.222 [0.221, 0.224] (0.000)

Table 4: Time (in seconds) used for embedding, SAS and Jaccard for different models. (Mean [range] (sd))

5.2 Experiment 2

In Experiment 2 (E2), we extended the original experiment by exploring the robustness of the models when generating different numbers of clusters. We used the constant sample size 8800 as the original experiment, and ran the models to generate 2, 20, 30, ..., 100 clusters respectively. We reported the resulting average SAS and Jaccard scores to the different numbers of clusters respectively in Figure 3 and 4.

We found that for Jaccard scores, all models show the same trend: with the numbers of clusters increaset, the Jaccard scores also increase (with an exception on 80 clusters). For SAS, all the models show a trend of decreasing SAS when the number of clusters increase. Although the averaged SAS scores generated by BERT model are consistently lower than the baseline models except LDA-TFIDF, the SAS for BERT is more stable than other models as the number of clusters increase. This is contradictory to H2 - BERT is actually more robust to larger number of clusters comparing to baseline models.

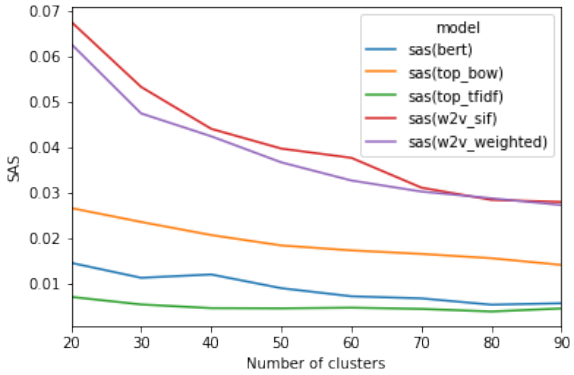

Figure 3: SAS scores generated from cluster numbers 20, 30, ..., 100

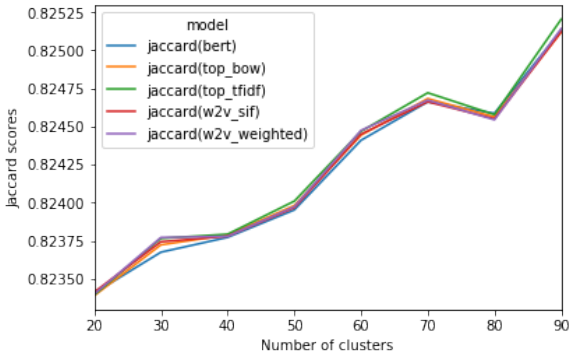

Figure 4: Jaccard scores generated from cluster numbers 20, 30, ..., 100

6 Computational requirements

While we did not finish finetuning BERT, we report the computational cost of the experiments reported in this document.

The hardware used in this report are CPU - AMD 3800X and GPU - GTX 970 for BERT for BERT probabilistic cluster matrix building and finetuning. We ran all clustering and scoring code

Author subreddit scraping

Dataset Size	Process Time(s)
1000	29.167
2200	490.336
4400	144.986
8800	503.415
13200	1876.134

Table 5: Our recorded time (in seconds) used for scraping author subreddit information. Note that this is an underestimate as the scraping APIs were interrupted several times by Reddit connection issues.

Model embed runtime (s)

Dataset Size	BERT	W2V	LDA
1000	24398	4.6464	43.0147
2200	58587	12.2305	114.8945
4400	106874	20.3448	264.5117
8800	213419	54.2024	618.4189

Table 6: Our recorded time (in seconds) used for generating model embeddings (or, in the case of BERT, the similarity matrix).

on `nlpg.cs.washington.edu` servers.

Here is a summary of the time it took to complete each component of our pipeline on this hardware:

It took us approximately 3 hours to scrape all of our original post dataset (465,000 posts) from r/Advice.

7 Discussion and recommendations

In this report, we discussed the performance of BERT-based probabilistic clustering algorithm, and clustering in comparison to two baseline embedding algorithms (W2V and LDA) on the task of clustering similar Reddit posts. Although the original paper suggests that BERT outperforms the baselines, we found that without finetuning, BERT actually generally performs worse than the base-

Clustering runtime (s), varying dataset sizes

Dataset Size	BERT	W2V	LDA
1000	0.01061	0.07052	0.048965
2200	0.02466	0.2301	0.1512
8800	0.08879	1.5668	0.8509

Table 7: Our recorded time (in seconds) used for generating 30 clusters based on the embeds and similarity tables generated from each model.

Clustering runtime (s), varying cluster counts

Num Clusters	BERT	W2V	LDA
20	0.1170	1.1177	2.0807
30	0.08879	1.5668	0.8509
40	0.1831	2.5230	1.4592
50	0.2504	2.6538	1.5056
60	0.3178	2.6669	1.6319
70	0.3724	2.8099	1.6970
80	0.4216	2.9076	1.8158
90	0.5140	2.9696	1.9037
100	0.5570	3.1283	1.9136

Table 8: Our recorded time (in seconds) used for clustering 8800 posts based on the embeds and similarity tables generated from each model.

SAS Scoring runtime (s), varying dataset sizes

Dataset Size	BERT	W2V	LDA
1000	0.0002334	0.0002324	0.00021815
2200	0.0004839	0.0004637	0.00047755
8800	0.002435	0.002338	0.002383

Table 9: Our recorded time (in seconds) used for SAS scoring 30 clusters based on the embeds and similarity tables generated from each model.

Jaccard Scoring runtime (s), varying dataset sizes

Dataset Size	BERT	W2V	LDA
1000	0.2994	0.29325	0.2928
2200	1.5046	1.54735	1.53545
8800	13.2359	13.99765	13.88065

Table 10: Our recorded time (in seconds) used for Jaccard scoring 30 clusters based on the embeds and similarity tables generated from each model.

SAS Scoring runtime (s), varying cluster counts

Num Clusters	BERT	W2V	LDA
20	0.002525	0.002550	0.002704
30	0.002435	0.002338	0.002383
40	0.002498	0.00287	0.002856
50	0.002672	0.002807	0.002808
60	0.002760	0.002877	0.002890
70	0.002767	0.002921	0.002875
80	0.002865	0.002928	0.002988
90	0.002834	0.002968	0.002987
100	0.002903	0.002903	0.002860

Table 11: Our recorded time (in seconds) used for SAS scoring 8800 posts based on the embeds and similarity tables generated from each model.

Jaccard Scoring runtime (s), varying cluster counts

Num Clusters	BERT	W2V	LDA
20	29.7165	35.1004	35.0141
30	13.2359	13.99765	13.88065
40	14.6562	18.2664	18.3352
50	12.8145	14.6988	14.6989
60	11.1064	12.2416	12.2249
70	9.6302	10.3814	10.3389
80	8.5103	8.8622	8.8448
90	7.6976	7.7887	7.7920
100	6.9701	7.0075	6.9865

Table 12: Our recorded time (in seconds) used for Jaccard scoring 8800 posts based on the embeds and similarity tables generated from each model.

lines in both evaluation metrics. Given the computational resources needed for finetuning BERT and the extraordinarily long time to run BERT-based probabilistic clustering algorithm to generate the similarity tables for clustering, we recommend future researchers to choose W2V for the same task if they do not have access to sufficient resources for finetuning and running BERT.

However, we also found that BERT was more robust to larger numbers of clusters (in other words, generating more reliable smaller clusters) than the baselines. This finding is contradictory to our hypothesis. Future work can explore how robust a finetuned BERT is with respect to larger number of clusters, as well as what source of data for finetuning produces better performance. This was briefly discussed in the original paper but we were unable to explore further in this direction due to limits on available resources.

As for our own learning experience, this project was both challenging and enjoyable. The most challenging part was the lack of computing resources that were efficient enough for replicating this paper - we did not sufficiently anticipate this issue when choosing this paper in the beginning. We did not expect that `nlpg` servers were insufficient for running BERT and underestimated how long certain processes would take. However, we gained invaluable experience through designing and implementing the pipeline for this experiment, especially in application of publicly available libraries towards NLP purposes, as well as experience in experiment design and reporting.

References

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Zhilin Wang, Elena Rastorgueva, Weizhe Lin, and Xiaodong Wu. 2019. No, you're not alone: A better way to find people with similar experiences on Reddit. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 307–315, Hong Kong, China. Association for Computational Linguistics.