Measuring Embedding Efficacy in Capturing Review Contents

Sarah Brockman **Anonymous** **Anonymous**
{sbrockman, anonymous, anonymous}@cs.umass.edu

Abstract

We investigate how well various embeddings techniques capture high-level text review contents and meaning using reviews from RateMyProfessors.com. We examine embeddings such as GloVe, BERT, and word2vec in both a simple model setting using logistic regression and a higher-capacity model setting using neural networks, which show that embeddings fit to our data outperform those that are not. We also show that advanced embedding techniques in simple models outperform simple embeddings in complex models through a series of experiments. Finally, we visualize embedding clusters, and show that BERT embeddings perform exceptionally well at capturing underlying review contents.

Code submitted by: Sarah Brockman

1 Introduction

In this project, we compare how different embedding methods perform in their ability to capture the main ideas contained in text reviews. The reviews and tags representing a high-level overview of their content are taken from RateMyProfessors.com. We train multiple classifiers to identify the correct tags with various embedding techniques. How the review text is embedded may have a large impact on performance, and one embedding technique may outperform others on its ability to capture this high-level content.

Some previous work has been performed on investigating how well embeddings capture the semantic meaning of sentences. For example, White et al. (2015) used a two-way semantic classification task on different embeddings. They used an autoencoder and bag of words, among some other techniques. They report that the bag of words model was one of the best. However, their task is a bit different from what we present here. Their semantic classification task involves classifying documents into two very different categories. Our tag labelling task is much more nuanced and requires much more fine-grained content capturing. Additionally, each review has not one but *three* potential meanings, making the task more difficult, and more strain put on the embeddings to accurately capture the meaning of the sentence.

2 Dataset

Our dataset consists of data from the website RateMyProfessors.com (Cheddar, Inc., 2020). Each review on the site can be accompanied by up to three tags (minimum of zero) that highlight the main feelings the review conveys regarding the professor. There are 20 different tags on the site that a reviewer can choose to be associated with their review. A few examples of the tags include "respected", "lecture heavy" and "accessible outside of class". The tags span most of the typical complaints or praises students may have toward a particular class or the professor's method of teaching it. To acquire a large number of reviews and their corresponding tags, we wrote our own web crawler in Python which sends HTTP requests to the site.

Each professor on RMP has a unique teacher ID (`tid`) that appears in the URL for their page. Each `tid` is an integer, which allows us to easily loop over numerous IDs in our code and make a request to the site. For our first round of data collection, we looped over 10,000 IDs and collected about 38,000 reviews. Each professor that appears has a maximum of 20 reviews in the dataset. Table 1 shows the statistics for our initial data collection. Nearly three-quarters of the reviews had all three tags, but a fair number also had none. Not many reviews contained one or two tags.

In most of our experiments, we use only those re-

Review Tag Counts

Number of tags	Number of reviews
None	13,192
One	1,493
Two	2,964
Three	20,889
Total	38,538

Table 1: Number of reviews with each number of tags.

views with three tags. The minimum review length in the dataset was 2 characters, with maximum length 701 and average length 245.78. Using a simple white space tokenizer, we obtained some basic statistics related to the words in the corpus. The total number of words is 881,962 and the vocabulary size is 18,778 unique words. These values may vary if a different tokenizer is used. For example, to improve compatibility with pre-trained embeddings it would be beneficial to use a tokenizer similar to the one used to train the embeddings. Table 3 shows the counts for each unique tag in the dataset, which can appear at most once per review. These values were calculated across only reviews with three tags.

To preprocess our data, we made all characters lowercase and removed punctuation. There were numbers in a fair amount of reviews but those can provide some important additional meaning. For example, a review could use numbers to specify number of homework problems. This may indicate the professor gives many (or few) homework problems, and the review may contain the tag "lots of homework". Some example reviews and their respective tags from our initial RMP dataset can be found in Table 2. The first review mentions difficult readings, and mentions that grading is fair and that the professor is always willing to help. The corresponding tags "gives good feedback", "get read to read", and "clear grading criteria" highlight these main points. The second review mentions a lot of homework and that the professor cares, which gives rise to the tags "lots of homework" and "caring", but does not explicitly mention the third tag, "respected". This will be a significant form of noise in our data; some reviews may not mention the tags the associated tags. However, implicit mention of the tags without exact word-for-word appearance in the review should still be able to be captured by the models.

The data we scraped from RMP did not require much preprocessing, only those steps mentioned above. We decided not to perform any spelling correction during preprocessing. While we ran our preliminary experiments, we collected more data from RMP, and we use this larger dataset for all of our experiments that appear later in the report. Large amounts of data are important for generalizability and are necessary for learning our own embeddings as this task is challenging with little data. Overall, the larger dataset has a total of 283,282 data points and includes the `tid` as well, in addition to the comment and tags of the review. Keeping track of the `tid` allows us to ensure that reviews from the same professor do not appear in both the training and test sets. This allows us to be certain that we are not "training on the test set", a common source of inflated accuracies in machine learning. The dataset statistics reported in this section reflect those in the larger dataset, albeit with scaled up values due to the larger number of data points. We use the initial dataset for these statistics because it is fairly representative of the whole dataset.

We split our dataset into training, validation, and test sets prior to model development. The test set is completely held-out and reserved until the end of model development so we can faithfully compare the performance of each of the different embedding methods. The train/validation/test split for all of our experiments unless otherwise specified is 80/10/10%.

3 Model Baselines

One of the first goals of the project was to obtain our classification baselines. Since our task is multi-label classification (predicting the three most likely tags for each review), we will be using a few different metrics. The three simplest metrics measure perfect match, two-thirds match, and single match percentages between the target and predicted labels. Thus, we will have three different baselines for these metrics. In all three baselines, the "classifier" we are comparing against always predicts the three most common tags in the dataset. The *perfect match* metric computes the percentage of reviews for which the three predicted tags match the ground truth tags perfectly. The *two-thirds match* metric computes the percentage of reviews for which at least two of the three predicted tags match. Finally, the *single match* metric computes the percentage of reviews for which only one of the predicted tags match the true tag. These baselines calculated on our larger dataset can be found in Table 4 (all val-

RateMyProfessors Dataset Examples

Tag 1	Tag 2	Tag 3	Review Text
gives good feedback	get ready to read	clear grading criteria	ben is a great professor oftentimes academic readings can be difficult to grasp but ben simplified them very well during lectures grading is fair and lectures are engaging ben knows his stuff and is always willing to help
respected	lots of homework	caring	there is a lot of homework given in this class but brannon really cares about his students and help you out i highly suggest taking his class

Table 2: Example entries in the RateMyProfessors dataset. Each review in the dataset is accompanied by three tags that highlight the main ideas of the review text.

Individual Tag Counts

Tag	Count
gives good feedback	6376
caring	5363
respected	4502
participation matters	4001
skip class? you won't pass	3864
clear grading criteria	3723
amazing lectures	3596
accessible outside class	3511
tough grader	3295
get ready to read	3287
hilarious	3099
inspirational	2880
lots of homework	2770
lecture heavy	2511
extra credit	1995
test heavy	1601
group projects	1577
graded by few things	1404
so many papers	832
beware of pop quizzes	641

Table 3: Counts of each unique tag in the initial dataset.

ues hereafter will be computed on the larger dataset, unless otherwise specified).

These metrics will be computed for each of our models and the results will be compared against these baselines. If a model does not beat these baselines, then it is likely not learning anything significant and should not be favored over other models. The baselines also serve as a quick check to make sure the models have been developed correctly and do not contain any underlying errors.

In addition to these metric baselines, we also experimented with a simple logistic regression classifier to serve as a more reasonable classifier performance baseline (i.e., more similar to how our later models will likely perform). This model was built using components from scikit-learn (Pedregosa et al., 2011). For the multilabel task, a separate logistic regression model was trained for each of the tags. Then, the three tags with the highest probability are used as the final prediction. We tested four different versions of logistic regression: one with the review text represented as word counts (bag of words), another with the review text represented as word counts (bag of words) with stopwords filtered out, one with review features extracted using the scikit-learn TF-IDF vectorizer and another with review features extracted using the scikit-learn TF-IDF vectorizer with stopwords filtered out. We used scikit-learn's standard english stopwords list for filtering. The bag of words review representation encodes each review as a vector with the number of times each word in the corpus appears in the review. The TF-IDF feature vectors provides weights for each word in the review based on how rare the word is in the corpus. Words that appear in fewer documents have higher weights and words that appear in many documents have lower weights (Jones, 1972). Table 5 shows the accuracies for the four models on the test portion of our initial dataset using the three accuracy metrics discussed earlier in the section. Both models performed very well on the single-match accuracy. This is to be

Naive Baseline Accuracies

Baseline	Accuracy
Perfect match	0.01522
Two-thirds match	0.20370
Single match	0.59538

Table 4: Perfect, two-thirds, and single match baselines for the respective metrics.

LR Baseline Accuracies - Test Set

Features	Single	Two-thirds	Perfect
BOW	0.78648	0.30867	0.02021
TF-IDF	0.83629	0.37820	0.03944
BOW_{SW}	0.80029	0.31558	0.02366
$TF\text{-}IDF_{SW}$	0.83431	0.39349	0.03353

Table 5: Accuracies for single, two-thirds, and perfect match metrics on initial test dataset for logistic regression with two different feature vectorizers: bag of words (BOW) and TF-IDF with stopwords (SW) and without.

LR Baseline Accuracies - Train Set

Features	Single	Two-thirds	Perfect
BOW	0.97959	0.85283	0.49494
TF-IDF	0.95166	0.64950	0.14673
BOW_{SW}	0.98286	0.86831	0.51831
$TF\text{-}IDF_{SW}$	0.94889	0.62872	0.13156

Table 6: Accuracies for single, two-thirds, and perfect match metrics on initial training dataset for logistic regression with two different feature vectorizers: bag of words (BOW) and TF-IDF with stopwords (SW) and without.

expected because only one of the three target tags must match the predicted tag. The single match baseline is 0.58926 as can be seen in Table 4, and both models surpassed it. The models did exhibit a drop in performance on the two-thirds and perfect-match accuracies, but this is expected behavior due to the definition of the metrics, and both still improved upon the baseline by a significant amount. The TF-IDF features perform a margin better than the BOW features for all three metrics. Although the single-match accuracy is very high for both models, the two-thirds and perfect-match accuracies are still quite low. More advanced models and feature embedding techniques should be able to improve upon these initial accuracies. Finally, both models performed similarly with and without stopwords, so we will have to conduct more tests in the future to determine if the filtering is worthwhile.

Table 6 contains the same models and metrics evaluated on the training set for comparison with the test set. The bag of words model obtained significantly higher accuracies than TF-IDF on the training set but lower on the test set. This indicated that the BOW model was overfitting on the training data. The performance on the training dataset is not incredibly pertinent, but it will help us understand when models are overfitting. Additionally, the performance of the BOW model with respect to the perfect-match metric indicates that more advanced models with more complex function approximation abilities may be able to achieve much better performance in regards to matching all tags.

Although we will not explicitly report the results obtained on the training set in future experiments, we will still be checking these values to ensure we have implemented our models correctly and are not overfitting. From now on, we include stopwords in our training data, since removing them does not seem to have a very large effect on performance.

4 Exploration of Embeddings in Simple Models

Since the core goal of our work is to explore how well different embedding techniques capture review topics and contents, we need to compare these embedding techniques to both our baselines in Section 3 as well as against each other. The embeddings we have selected for study include word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and BERT (Devlin et al., 2018).

We first experimented with GloVe (Global Vectors for Word Representation) embeddings trained on Wikipedia data. GloVe embeddings are learned in an unsupervised manner in which word co-occurrences are calculated from the entire corpus. This global co-occurrence approach differs from embedding techniques that focus more on local context for individual words, such as word2vec (Pennington et al., 2014). Since the optimal length of the embedding vectors themselves can vary between tasks, we experiment with four different lengths: 50, 100, 200, and 300. Length 300 were the largest embedding vectors available for the Wikipedia GloVe set. The set contains embeddings for 6 billion unique tokens (Rare-Technologies, 2018). First, we fit four simple logistic regression models using the GloVe embeddings, so we can easily compare with our baselines before moving to complex models. The performance for these four LR models with respect to our three accuracy metrics can be found in Table 7.

The accuracies for all three metrics increases as the size of the embedding vector increases. The models with embedding sizes 100, 200, and 300 all outperform the BOW baseline LR model with stopwords included. However, none of the models

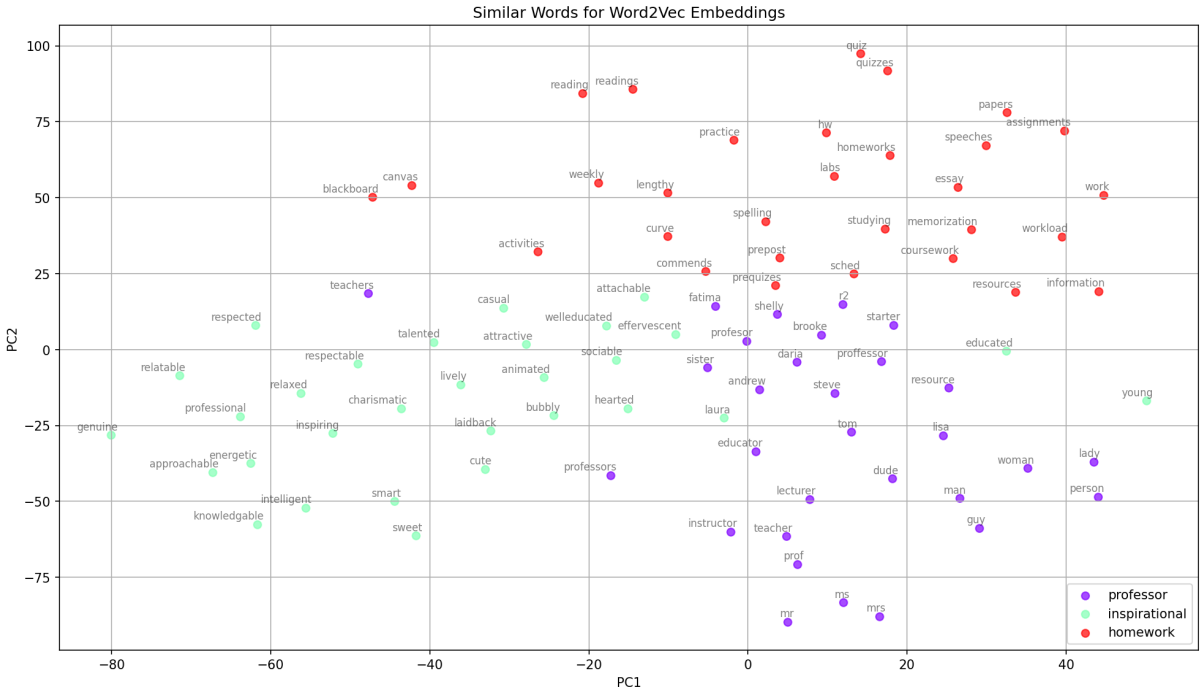

Figure 1: Top 30 most similar words to "professor", "inspirational", and "homework" based on word2vec embeddings.

GloVe Accuracies - Test Set

GloVe Size	Single	Two-thirds	Perfect
50	0.75641	0.28895	0.02169
100	0.78962	0.30478	0.02729
200	0.81090	0.33488	0.03199
300	0.81943	0.34473	0.03426

Table 7: Accuracies for single, two-thirds, and perfect match metrics on test dataset. Models are logistic regression classifiers trained with various size frozen GloVe embeddings. Stopwords are included.

outperform the TF-IDF baseline. This indicates that the GloVe embeddings may need to be used in conjunction with more complex models in order to outperform both of these simple embedding techniques.

4.1 Word2Vec

The next embedding technique we considered was word2vec. We compute the word2vec embeddings using our own training data, so these embeddings are not frozen and pretrained like the GloVe embeddings. The performance of various sizes of word2vec embeddings can be found in Table 8. The word2vec embeddings outperform the GloVe embeddings in every setting. This is partially expected due to the fact that the word2vec embeddings are fit to our training dataset, whereas the GloVe embeddings are pretrained on more general Wikipedia data.

Word2Vec Accuracies - Test Set

Word2Vec Size	Single	Two-thirds	Perfect
50	0.80369	0.32203	0.02928
100	0.82028	0.35033	0.03369
200	0.83710	0.37298	0.03834
300	0.84469	0.38018	0.03962

Table 8: Accuracies for single, two-thirds, and perfect match metrics on test dataset. Models are logistic regression classifiers trained with various size word2vec embeddings.

To ensure the word2vec embeddings were actually capturing meaningful information, we experimented with plotting some words and their top 30 most similar words based on embedding similarity. Figure 1 shows the top 30 most similar words for the words "professor", "inspirational", and "homework". The similar words are found based on embedding similarity, and then the words are plotted in a two-dimensional space using Principal Component Analysis (PCA). The clusters in this plot make sense from a human judgement standpoint. Pronouns and nouns that refer to a person, such

as "educator" and "lecturer", are clustered with "professor" in the lower right. Nouns that refer to assignments and schoolwork, such as "coursework" and "problems", are clustered with "homework" in the upper right. Finally, positive adjectives such as "caring" and "insightful" are clustered on the left with "inspirational".

4.2 BERT

The final embedding technique we consider is BERT. We experimented with both pretrained, frozen BERT embeddings and fine-tuned embeddings. These performance of the frozen BERT embeddings is quite similar to the performance of the GloVe embeddings, as can be seen in Table 9. It does well, but still not significantly different from BOW and TF-IDF feature vector baselines. This is likely because these BERT embeddings are frozen and not fine-tuned to our dataset. For this same reason, word2vec outperformed BERT as well.

BERT Accuracies - Test Set

	Single	**Two-thirds**	**Perfect**
BERT	0.80813	0.32113	0.02813
BERT$_{FT}$	0.88133	0.44333	0.05558

Table 9: Accuracies for single, two-thirds, and perfect match metrics on test dataset for pretrained frozen BERT and fine-tuned BERT (BERT$_{FT}$).

Fine-tuned BERT performed the best out of all the embedding techniques by a large margin. This is due to the fact that the fine-tuned BERT embeddings are further trained to fit our data. Thus, they are better at capturing the meaning behind certain words and the contents of the reviews. The fine-tuned BERT embeddings are also larger than the word2vec embeddings, which partially explains why they perform better.

4.3 Simple Model Discussion

In this section we analyzed three different embedding techniques and their performances when applied in a simple model setting, namely logistic regression. Fine-tuned BERT performed the best out of all the embeddings, by far. Word2Vec was the next highest in terms of accuracies, followed by GloVe which was quite similar to the TF-IDF baseline. This behavior is explained by the fact that fine-tuned BERT and word2vec are fit to our RMP dataset, whereas GloVe is not. To better investigate how well the embeddings are capturing the tags in the embeddings, we can visualize embedding clusters based on their tag sets. This is one of the most important facets of our work; it shows how well the embedding techniques are at capturing the meaning behind the tags.

Figure 2 shows a comparison of the embedding clusters for TF-IDF, GloVe, word2vec, and BERT. The embeddings are plotted for reviews belonging to two different tag sets: "gives good feedback, caring, respected" and "test heavy, lecture heavy, tough grader". These two tag sets quite different in their polarity (one has positive tags, while the other has tags that are likely viewed as negative across reviews). Fine-tuned BERT has the most distinct clusters of the five. This means that the reviews that have the same corresponding tags have very similar embeddings. Word2Vec has fairly pronounced clusters, but much closer together than fine-tuned BERT. TF-IDF has slightly less clear clusters than word2vec. GloVe and frozen BERT do not have as neat of clusters, but some smaller groups of similar reviews can be seen. This shows that fine-tuned BERT is by far the best at capturing the underlying review content specified by the tags.

To gain more understanding of how the different models compare to one another, we can analyze their precision and recall for each tag. This allows us to see if certain models are better or worse at predicting certain tags, or if they just happen to predict the same set of tags frequently and still obtain good results that way. We analyze the precision and recall for TF-IDF features (with stopwords), GloVe embeddings of size 300, and word2vec embeddings of size 300. These values can be seen in Table 10. The precision values for GloVe and word2vec are higher than those for TF-IDF for nearly every tag. This means GloVe and word2vec have lower Type I error rates (false positives). The recall values, however, are much lower for GloVe and word2vec than for TF-IDF. This means that GloVe and word2vec have many instances where they should predict a certain tag but fail to do so. There are tradeoffs between these methods, especially since they are very close in performance. If the user cares more about Type I error than Type II error, they may want to choose GloVe or word2vec.

The precision and recall values also tell us what tags are challenging for the models. The most challenging tag to predict is "graded by few things". This tag has the lowest precision and recall across all three embedding methods. This means that this

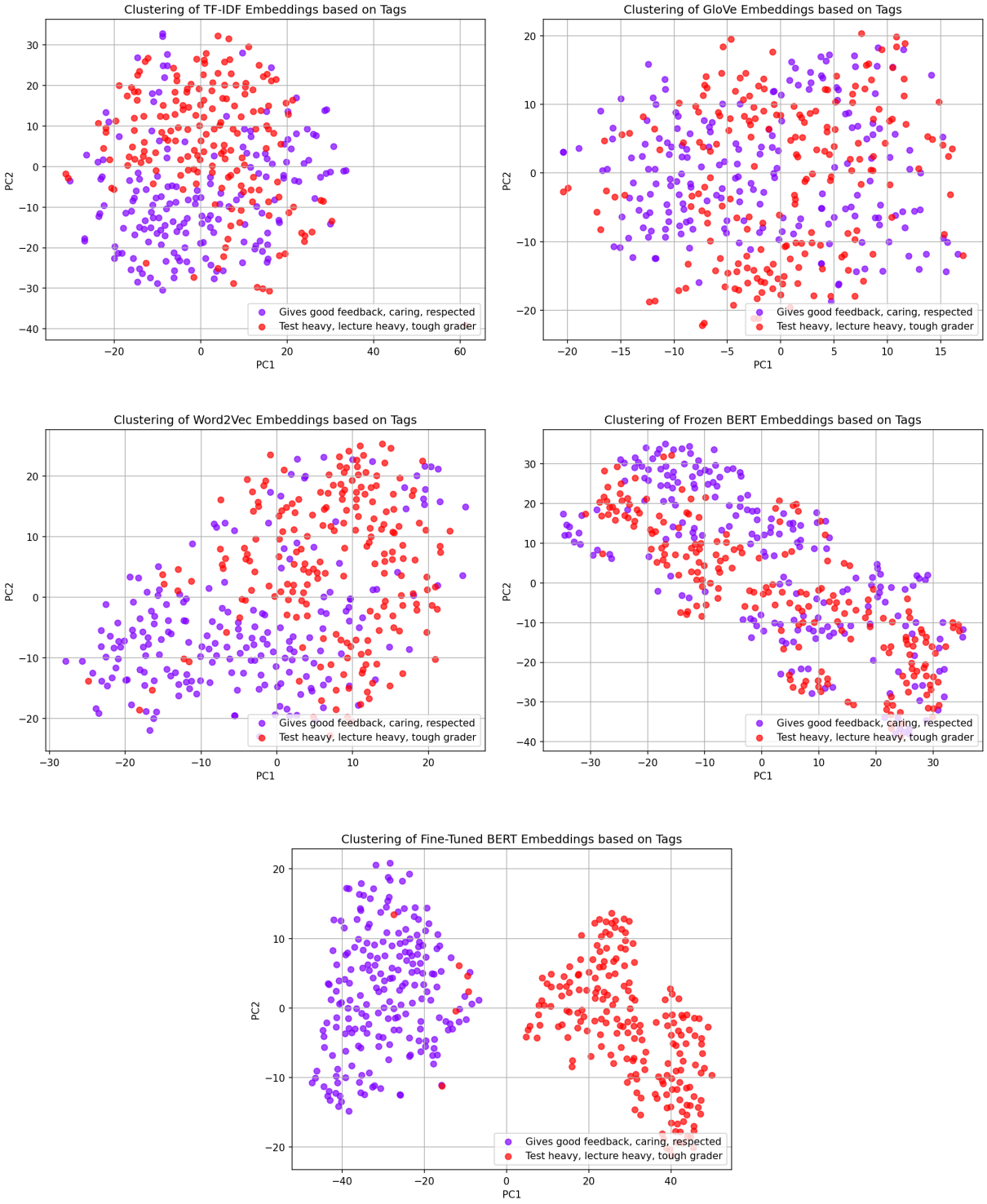

Figure 2: Clustering of review embeddings for TF-IDF, GloVe and Word2Vec (both size 300), Frozen BERT, and Fine-Tuned BERT for example positive and negative tag sets. Plotted using Principal Component Analysis (PCA). Here, Fine-Tuned BERT shows exceptional ability to cluster reviews based on meaning behind the tags, word2vec and TF-IDF show slightly pronounced clusters, while GloVe and Frozen BERT show poor clustering of review embeddings.

Precision and Recall Results

Tag	$\text{PRE}_{\text{TF-IDF}}$	$\text{REC}_{\text{TF-IDF}}$	$\text{PRE}_{\text{GloVe}}$	$\text{REC}_{\text{GloVe}}$	PRE_{w2v}	REC_{w2v}
gives good feedback	0.46058	**0.74521**	0.59506	0.24695	0.59770	0.29754
respected	0.37505	0.41786	0.44808	0.05453	0.44095	0.06539
lots of homework	0.41934	0.42687	0.54145	0.07509	0.55459	0.11498
accessible outside class	0.41065	0.30520	0.46913	0.01967	0.55276	0.05695
get ready to read	0.45996	0.42931	0.56974	0.09820	0.59058	0.14165
participation matters	0.37199	0.32998	0.47770	0.01794	0.50574	0.04211
skip class? you won't pass	0.36541	0.41550	0.37853	0.01670	0.46366	0.03340
inspirational	0.44727	0.29173	0.45454	0.06049	0.48140	0.07604
graded by few things	0.33279	0.11976	0.32258	0.00581	0.27272	0.00697
test heavy	0.43041	0.31305	0.40284	0.04701	0.48417	0.08462
group projects	0.57314	0.26599	0.61052	0.06455	0.61924	0.08235
clear grading criteria	0.37715	0.33774	0.35220	0.01544	0.47407	0.03529
hilarious	0.49005	0.37021	0.50513	0.08401	0.54391	0.10997
beware of pop quizzes	**0.58986**	0.17367	0.38888	0.01899	0.47826	0.02985
amazing lectures	0.44121	0.40786	0.43046	0.05519	0.45454	0.07925
lecture heavy	0.40000	0.38135	0.45421	0.04411	0.53603	0.08466
caring	0.44503	0.64568	0.52903	0.14340	0.56193	0.19517
extra credit	0.55060	0.31576	0.63120	0.09478	0.63247	0.11821
so many papers	0.37854	0.31576	0.26666	0.01366	0.40000	0.03416
tough grader	0.55840	0.69045	**0.67517**	**0.26961**	**0.69004**	**0.36505**

Table 10: Precision (PRE) and recall (REC) for each tag for TF-IDF (with stopwords), GloVe-300, and word2vec-300 embeddings on logistic regression.

tag is not predicted often, which means the models are never confident that this is one of the correct tags to apply.

5 Exploration of Embeddings in Advanced Models

Up until this point, only logistic regression has been used to make the predictions. This places a large restriction on the capacity of the model to capture relationships between the features in its input. It may be the case that certain representations of the input have more complex relationships within its features and thus require a higher capacity model to fully exploit their potential. Models such as neural networks have much larger capacity and thus may be able to capture these relationships if they exist. However, if these relationships do not exist, are too complex for the size of the network, or the neural network is unable to find them then the increased capacity will likely result in a reduction in test accuracy due to the increase variance of the model and overfitting. As a result, we decided use a simple feed-forward neural network with one hidden layer composed of 50 neurons to explore if this might be the case. We first compute the relevant baselines consistent with the logistic regression experiments. Then, we compute the neural network performance with GloVe and word2vec embeddings. We do not include BERT embeddings in this stage because training them is exceptionally time consuming, and they already performed very well in logistic regression.

5.1 Baselines

First, we used the BOW and TF-IDF features to train the baseline neural networks. The performance of these models can be found in Table 11. Compared to the performance of logistic regression, BOW improved while TF-IDF's performance declined. This bridges the gap that was observed between the two representations in logistic regression. As a result, the neural networks using simple features are competitive with or better than GloVe with logistic regression. This shows that simpler embeddings with higher capacity models are not necessarily better than more advanced embeddings in simple models, and there may be a large benefit in using these advanced embeddings.

NN BOW and TF-IDF Accuracies - Test Set

Features	Single	Two-thirds	Perfect
BOW	0.83790	0.35700	0.03460
TF-IDF	0.83235	0.36240	0.03700
BOW_{SW}	0.81805	0.33570	0.03370
$TF\text{-}IDF_{SW}$	0.82325	0.34420	0.03335

Table 11: Accuracies for single, two-thirds, and perfect match metrics on initial test dataset for neural networks with two different feature vectorizers: bag of words (BOW) and TF-IDF with stopwords (SW) and without.

5.2 GloVe

The GloVe embeddings performed significantly worse in the neural network than in logistic regression. Comparing the results found in Table 7 and Table 12, the decrease was of varying significance and was present for all embedding sizes for all three metrics. The most significant relative drop was with embeddings of size 100 for the perfect match metric, with a decrease of about 30%. This indicates that GloVe may not be capturing the content in the reviews, and a higher capacity model is propagating these inaccuracies in a larger scale.

NN GloVe Accuracies - Test Set

GloVe Size	Single	Two-thirds	Perfect
50	0.74065	0.25990	0.01930
100	0.74045	0.26335	0.01985
200	0.78560	0.31135	0.02750
300	0.80985	0.33690	0.03060

Table 12: Accuracies for single, two-thirds, and perfect match metrics on test dataset. Models are neural networks trained with various size frozen GloVe embeddings. Stopwords are included.

5.3 Word2Vec

The word2vec embeddings performed consistently better when used in the neural network. Comparing the results found in Table 8 and Table 13, word2vec improved for all embedding sizes for all metrics. The amount of improvement varied, with the most significant relative improvements occurring in the perfect match metrics.

We also ran word2vec for larger embedding sizes to determine if performance would continue to improve with increased size. This turned out to not be the case. Word2Vec size 500 did improve over the size 300 embeddings for both Single and Two-thirds accuracy scores, but failed to improve over the Perfect accuracy score. The same can be said for the size 1000 embeddings. When comparing the size 1000 embeddings with the size 500 embeddings however, all three accuracy scores were worse. Therefore we were able to conclude that increasing the embedding sizes does not always improve performance. The results from both the size 500 and size 1000 embeddings are still better than any of the results from the previous word2vec results with the logistic regression model. Neural networks with word2vec as a whole perform better than the logistic regression word2vec models.

NN Word2Vec Accuracies

Word2Vec Size	Single	Two-thirds	Perfect
50	0.80860	0.33885	0.03535
100	0.83230	0.36850	0.03890
200	0.84260	0.38385	0.04110
300	0.84740	0.38920	0.04570
500	0.85110	0.39265	0.04265
1000	0.85070	0.39000	0.04345

Table 13: Accuracies for single, two-thirds, and perfect match metrics on test dataset. Models are neural networks trained with various size word2vec embeddings.

6 Error Analysis

In this section, we will analyze some reviews that were difficult to predict tags for, as well as reviews where the models made very inaccurate predictions. This will help us determine where our models may be going wrong and where they can potentially be improved. We will look at a few incorrectly tagged reviews for each model. The following is a review for which TF-IDF logistic regression mislabelled all three tags:

> "he demands respect by respect you must know exactly what he wants you to know at all times be prepared to put a lot of time and effort if you want a good grade the pop quizzes can take you by surprise easily if you dont stay on top of the ungraded homework softspoken recommend sitting towards the front."

The true tags for this review are "lots of homework", "so many papers", and "tough grader", whereas the predicted tags were "gives good feedback", "get ready to read", and "caring". The predicted tag "gives good feedback" does not really

make much sense here, but was probably predicted as a sort of default tag since it is the most common tag in the dataset. "Get ready to read" does make some sense since the review mentions putting in a lot of time and effort. The "caring" tag also does not make much sense here, but could have been predicted due to the presence of the word "respected", which it may coincide with a lot. Overall, this review does not have a lot of keywords for any of the tags, so this review represents a form of noisy review in our dataset, for which assigning tags is somewhat tricky and arbitrary.

Below is a review mislabelled by the GloVe-300 logistic regression classifier:

> "dont do it man just dont."

It is a very succinct review, so naturally it is very difficult to assign tags to normally. The true tags for this review are "accessible outside class", "amazing lectures", and "extra credit". The predicted tags are "skip class? you won't pass", "test heavy", and "lecture heavy". This is an interesting case, because the review itself sounds very negative. Saying "don't do it man" implies that the reader should not take a class with this professor (indicating that they did not like his or her style of teaching). However, the true tags all have positive connotations, whereas the predicted tags are what one would expect the tags to be for this review (ones with more negative connotations). A potential remedy to reviews such as this one would be to impose a certain character threshold for reviews, and leave out reviews that do not have enough characters are left out of the dataset.

Finally, below is a review that was mislabelled by the word2vec-300 classifiers:

> "i really enjoyed taking kasies class but it seems that some days if you give an answer in class be prepared to fight with her and being shamed in front of the class but also dont stay quiet because she likes to call on students however she does give extra credit and works with the student a lot if you take it into your own hands."

This review's true tags "get ready to read", "skip class? you won't pass", and "graded by few things". The predicted tags are "gives good feedback", "clear grading criteria", and "caring". The first predicted tag makes some sense due to the metnion of how the instructor facilitates in class discussion. There is nothing in the review that mentions grading, so the second predicted tag does not quite fit. Finally, "caring" is definitely not a tag that should be used for this review, given how the student sometimes feels "shamed" in class.

It is clear in the above examples that this task can be quite challenging for some reviews that have no mention of anything pertaining to any of the tags. These reviews are a large source of noise in our dataset, and pose a challenge to the models. Thus, it would be helpful to see if the same reviews were a source of difficulty for TF-IDF, GloVe, and word2vec. In fact, all the models have difficulty labelling the same set of 2,959 reviews, which includes the above three examples. The reviews in this set are difficult to label in general, so all models struggle with them.

7 Discussion and Future Work

We first created a new dataset collected from RateMyProfessors.com, which contains text reviews alongside corresponding tags that illustrate the content of the review at a high-level. Then, we created some simple baselines based on how many tags are predicted correctly, which we use for comparing the results of our various models. We also established few simple model baselines, which consist of bag of words and TF-IDF features fed into a multi-label logistic regression classifier and a neural network.

The core of our work consisted of experimenting with different embedding techniques, namely GloVe, word2vec, and BERT. Our goal was to estimate how well these embedding techniques capture review contents and meaning. We first experimented with different size GloVe embeddings trained on Wikipedia data. As the embedding size grew, the performance on the logistic regression classification task also improved. However, the performance of the best GloVe embeddings was roughly similar to the TF-IDF baseline. We then tried different size word2vec embeddings, which outperformed TF-IDF and GloVe. This is expected, since the word2vec embeddings are fit to our dataset. Finally, we experimented both with a pretrained, frozen BERT and a fine-tuned BERT. Frozen BERT performed roughly the same as GloVe, but fine-tuned BERT performed the best out of all embedding techniques by a large margin. This is due to the fact that BERT embeddings are large (length 768) and are fine-tuned to our specific

dataset.

We visualized the embeddings for these different techniques on two different tag sets, and showed that fine-tuned BERT was clearly the best at clustering embeddings based on underlying review meaning, and therefore fully capturing the content mentioned in the tags. We also analyzed the precision and recall for each of the tag classes to determine which ones pose challenges for the classifiers.

Finally, we used the embeddings in a neural network, a higher capacity model than logistic regression. We showed that the higher capacity does not necessarily imply better performance. Importantly, the bag of words and TF-IDF performance in the neural net model did not outperform the logistic regression performance of the more advanced embedding techniques. This shows that there is a great benefit of using these more advanced embedding techniques, particularly BERT, even in simple models.

One of the most promising avenues for future work in this area is fitting a larger neural network and tuning the hyperparameters for the various embeddings. We did not perform much hyperparameter tuning in our neural net experiments, and this would likely lead to significant performance increases over our neural net accuracies and those for logistic regression. Additionally, we did not use BERT embeddings in our neural net experiments, so using BERT in conjunction with a large high-capacity neural net with fine-tuning enabled would likely lead to very large performance increases on this task. It would also be interesting to investigate different techniques for increasing the performance on the Perfect metric. All models stayed below 5% accuracy on this metric, which is understandable but still low.

8 Conclusions

This work investigated which embedding techniques were best at capturing the content and meaning in review text. When compared with bag of words, TF-IDF, GloVe, and word2vec, fine-tuned BERT embeddings performed the best in both classification accuracy and embedding clustering. Word2Vec performed second-best, due to the fact that it is fit to our dataset like fine-tuned BERT. GloVe performed the worst of the advanced embeddings, hovering around the same accuracies as the bag of words and TF-IDF baselines.

Overall, all embedding techniques showed surprising ability to predict correct tags for the reviews. All embedding techniques were able to predict at least one correct tag consistently in the logistic regression setting. The embeddings used in neural networks showed that using these advanced embeddings in a simple model such as logistic regression is better than using simple or naive embeddings in highly complex models.

References

Cheddar, Inc. 2020. Rate my professors. https://www.ratemyprofessors.com/.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Karen Spärck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21.

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.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Rare-Technologies. 2018. Gensim-data. https://github.com/RaRe-Technologies/gensim-data#models.

Lyndon White, Roberto Togneri, Wei Liu, and Mohammed Bennamoun. 2015. How well sentence embeddings capture meaning. In *Proceedings of the 20th Australasian document computing symposium*, pages 1–8.