Date: December 14th, 2023.

Disclaimer This latex template has been obtained from IEEE ACCESS for educational purpose.

Comparative analysis of text embedding-based recommendations

ANDY CASAÑAS[1], JOAQUÍN DUNNER[1], JOSÉ VALLADARES[1], MARTÍN PEÑALOZA[1]
[1]Pontifical Catholic University of Chile

Course: IIC3633 Recommender Systems.

ABSTRACT In this paper, we study the generation of recommendations based on text descriptions. For this purpose, four embedding generators belonging to different families are used: TF-IDF, Word2Vec, BERT and GPT-2. According to the literature and their particular structures, all these methods present strengths and weaknesses inherent to their particular implementations. The objective is to analyze the performance of the different methods in different domains or datasets, looking at how the different embedding generators influence the recommender systems. For this purpose, recommendations are generated in three datasets of different natures: one in jokes recommendation (Jester Jokes), another in beer recommendation (Rate Beer) and the last one in movie recommendation (Movies). The analysis methodology is similar for all cases. The results and discussion are presented and analyzed in detail in the corresponding sections.

INDEX TERMS Recommender systems, text embeddings, GPT-2, BERT, TF-IDF, Word2Vec.

I. INTRODUCTION

In recommender systems, the general aim is to deliver the best possible result with the information at hand. Hence the importance of being able to capture the information, and the patterns within it, in the best possible way. The information available comes from different sources, and therefore there are different ways to process each type of data. Within these sources, we have behavioral, textual, visual information, among others. This paper focuses on the information that comes from generating text embeddings.

In this paper we study the performance of different techniques used in natural language processing (NLP) on datasets from different recommendation domains. This is done in a context of content-based recommendation based on the generated embeddings. Four specific embeddings generators will be studied: TF-IDF, Word2Vec, BERT and GPT-2.

A central idea of the paper is that the different models point to different techniques for capturing the information obtained from text descriptions:

- **TF-IDF**. The origins of this method date back to 1972, thanks to Karen Spärck Jones who devised a statistical interpretation of term specificity called *Inverse Document Frequency* (IDF) [12]. TF-IDF focuses on looking at the number of times a word appears in a text (TF) and the importance of a word in a collection of texts (IDF). It allows differentiating between important and unimportant words depending on their frequency in the corpus and their frequency of repetition.
- **Word2Vec**. It is an NLP technique published in 2013 [8] [9]. It represents words as vectors in a vector space where semantically similar words are closer to each other. This model is able to capture the context and meaning of words, but presents a finite dictionary of embedding options.
- **BERT**. Introduced in October 2018 by Google researchers [4] [10]. BERT is bidirectional and can analyze the context of a word based on the surrounding words both before and after it in the sequence. This allows it to capture the contextual meaning of words in a text. Because of this it is useful for understanding and processing human language.
- **GPT-2**. Developed by OpenAI and first released in partial form in February 2019 and in full form in November 2019. It is a transformer capable of understanding and generating text that is contextually relevant and coherent. It learns patterns and relationships in text information. This model is not only able to understand the context of words, but can also see a coherent continuation pattern for them.

Thus, seeing that different models have different approaches and apply different techniques, we can understand that each model has its strengths and weaknesses. This diversity in the models makes it interesting to study how the models will behave and perform in different dataset domains.

II. RELATED WORK

A review of different pre-trained language models that were available between 2018 and 2020 and represented the state of the art at that time is presented in Aßenmacher et al. [2]. This study focused on identifying the differences between the models, but does not perform quantitative comparisons between them. The models addressed in that work are: Word2Vec, FastText, ULMFiT, ELMo, GPT, BERT, GPT2, XLNet, RoBERTa, ALBERT.

In Raffel et al. [11] they follow a similar line by analyzing the comparability between different transformer-based models. They perform several experiments with respect to the learning transfer capability of a transformer encoder-decoder architecture, varying the pre-training target, pre-training resources and parameter size.

In Yang et al. [13] perform an ablation study to explicitly compare their XLNet with BERT. In this ablation study, they train six different XLNet-based models where they modify different parts of the models to quantify how these design choices influence performance. At the same time, they restrict themselves to an architecture of the same size as BERT-base and use the same lexical resources for pre-training.

Liu et al. [7] vary their RoBERTa model with respect to model size and use of pre-training resources to perform an ablation study to achieve comparability with BERT. Lan et al. [5] go a step further by comparing their ALBERT model with BERT with respect to run time and model width/depth.

In Zhang et al. [14] proposed to use pre-trained language models (PLMs) such as BERT and GPT as recommender systems. The proposed method was evaluated on the MovieLens movie recommendation dataset using zero-training and fine-training settings. In the zero-training setting, it was observed that PLMs significantly outperformed the random recommendation baseline, although a strong linguistic bias was noted when using PLMs as recommenders. In the fine-training setting, this bias was reduced with training data available; however, PLMs performed worse than traditional recommender systems such as GRU4Rec.

III. METHODOLOGY

The methodology implemented for the experiments was as follows. First, we pre-processed the datasets to keep users who have several items consumed in common. Then, with each model we generated embeddings, for the items that were selected from the datasets, with the relevant text for each item. The users are represented as a weighted average of the rating they gave to the item and the embedding that was generated. Thus, the dataset is divided into two portions, one for training and the other for testing. Finally, cosine similarity is used to find the similarity between the users and the elements in the testing set.

The following explains how the embeddings are generated in the four employed methods:

In the case of TF-IDF, the training set is concatenated with the testing set, generating a list from the column used to create the embeddings. The class TfidfVectorizer form *scikit-learn* library is used to convert this list into a TF-IDF matrix. The resulting matrix reflects the importance of each word. This matrix is then split into two parts (training and testing), and the resulting embeddings are stored in a new column.

For Word2Vec, a similar process of concatenation between the training and testing sets is followed, generating a list from the column intended to generate the embeddings. The synopses are tokenized, breaking them into individual words, creating a list of lists where each sublist contains the tokenized words of the synopsis. The Word2Vec class from the *gensim* library is used to train a Word2Vec model using the Continuous Bag of Words (CBOW) method, which aims to predict the target word (center) given a surrounding context. Subsequently, embeddings are generated for each word in each synopsis using the trained Word2Vec model. The resulting list is split into training and testing, and these embeddings are stored in a new column.

In the case of BERT, BertTokenizer and BertModel are imported from the *transformers* library. The "bert-base-uncased" pretrained model is used, a variant of BERT trained on lowercase text without case distinction, along with its tokenizer. The input text is tokenized, and the necessary special tokens for BERT (CLS at the beginning and SEP at the end) are added, returning identifiers as PyTorch tensors. A forward pass is made through the model to obtain the embeddings, which are extracted from the last hidden layer. Then, average pooling is performed to obtain a single embedding vector, applying this function first to the column intended to extract the embeddings in the training set and then in the testing set.

As for GPT-2, GPT2Tokenizer and GPT2Model are imported from the *transformers* library. The pretrained GPT-2 model is used to obtain embeddings from input text and its corresponding tokenizer. Similar to BERT, the input text is tokenized, and identifiers are returned as PyTorch tensors. A forward pass is made through the model to obtain embeddings from the last hidden layer. Then, average pooling is carried out to obtain a single embedding vector, applying this function first to the column intended to extract the embeddings in the training set and then in the test set.

Finally the methodology to measure the performance was as follows. First, we analyze whether the recommended item was delivered in the order of user preference or not. In addition, the time taken by the recommender to generate the recommendation is added. Finally, the nDCG of the delivered items is obtained. All metrics are compared with a SVD matrix factorization baseline, with 20 latent factors.

IV. EXPERIMENTS

A. JESTER JOKES DATASET

This dataset contains 100 jokes with $73,422$ user reviews. Since most users have no interactions with most of the jokes, only users who have consumed all the items are chosen. Thus, you end up with a completely filled matrix, which includes 100 items (jokes) and $14,116$ users. The dataset is divided into two sets of training and testing, where 80 items are assigned to the former and 20 to the latter.

Each user is represented by the weighted average of their reviews (a number scaled to be between 0 and 1) and the embeddings of each item, in the form of a matrix product. Subsequently, each user's degree of similarity to the new embeddings of the test items is calculated, using cosine similarity as a metric. For each joke, the similarities are ordered from highest to lowest and a list of recommendations is generated.

Finally, a similarity metric nDCG is calculated for the whole testing set (all 20 items). Also, a baseline matrix factorization recommendation with SVD is computed. This is done using the *scikit-surprise* library, which allows to obtain the matrix factorization from the training and testing datasets. The main results are as follows:

Method	nDCG@20	Time [s]
GPT-2	0.23512	59.65012
BERT	0.23597	38.85705
TF-IDF	**0.26278**	42.96842
Word2Vec	0.24255	**20.57679**
Baseline SVD	0.02473	32.10575

TABLE 1: Results for Jester Jokes Dataset

The processing times in Table 1 correspond to the time it takes to generate the complete recommendation, not counting the time it takes to extract the embeddings. These were extracted externally and imported. In this case, the fastest processing method is Word2Vec, with time close to 20 seconds. This may be relevant, depending on the type of implementation of the recommender system.

Regarding the performance metrics, it is observed that all methods outperform the baseline with factors close to 10. The best recommendations are delivered by the TF-IDF method, with 0.26278. In second place is Word2Vec, with 0.24225, although the difference between it and the GPT-2 and BERT methods is minimal (less than 0.01).

B. DATASET RATE BEER

This dataset (Lerro et al. [6]) has $40,213$ users, $110,419$ items and $2,855,232$ of beer reviews. In this case, contrary to the Jester Jokes dataset, very few users share reviews of the same items. Therefore, the 30 users with the most interactions are considered. Among them alone, there are more than $60,000$ reviews of items. To filter, a sample of 30 item reviews is selected for each user, which is equivalent to the training set. The testing set corresponds to the items that the 30 users share, which are 9 beers (not repeated in the training).

As in the Jester Jokes dataset, each user is characterized as a matrix multiplication between normalized ratings (between 0 and 1) and the embeddings of the consumed jokes. Then, cosine similarity is taken between the user vectors and the embeddings of the beers in the testing set, to obtain a list of recommendations. These are ordered from highest to lowest and used to generate the statistics. The baseline is then calculated as a matrix factorization with SVD.

The Table 2 with results obtained for nDCG of all items in the testing set.

Method	nDCG@9
GPT-2	0.16991
BERT	**0.17388**
TF-IDF	0.16931
Word2Vec	0.16858
Baseline SVD	0.02799

TABLE 2: Results for Rate Beer Dataset

The performance metric shows a clear improvement of over 5 times of all methods over the baseline. In this case, the difference between the best and worst model (BERT and Word2Vec, respectively) is only 0.00530. In other words, the results of all the methods employed give very similar results, consistently better than the baseline. Even though the difference is minimal, it is observed that both families of models in which the context is taken into consideration (GPT-2 and BERT) have better results than the cases in which the context is not taken into account (TF-IDF and Word2Vec).

C. DATASET MOVIES

From this dataset, we used the files *movies_metadata.csv*, *link.csv*, and *rating.csv*. It searches for common movie items in the three files, yielding $265,879$ users, $7,437$ items, and $7,396$ movie overviews. The 30 users with the most interactions, which present more than $6,000$ movie overviews, are considered. A sample of 30 items is selected for each user, which is equivalent to the training set. The testing set corresponds to the movies that the 30 users share, which is 32 movies.

As in the previous datasets, each user is characterized as a matrix multiplication between normalized ratings (between 0 and 1) and the embeddings of the movies consumed. Then, cosine similarity is taken between the user vectors and the movie embeddings of the testing set, to obtain a list of recommendations. These are ordered from highest to lowest and used to generate the statistics. The baseline is then calculated as a matrix factorization with SVD.

Table 3 with results obtained for nDCG for all items in the test set.

Method	nDCG@32
GPT-2	**0.46307**
BERT	0.46058
TF-IDF	0.45652
Word2Vec	0.44879
Baseline SVD	0.002620

TABLE 3: Results for Movies Dataset

The performance metric shows a large improvement of all methods over the baseline. In this case, the difference between the best and worst model (GPT-2 and Word2Vec, respectively) is only 0.0053. In other words, the results of all the methods used are very similar, consistently better than the baseline. Even though the difference is minimal, it is observed that both families of models in which the context

is taken into consideration (GPT-2 and BERT) have better results than the cases in which the context is not taken into account (TF-IDF and Word2Vec).

V. DISCUSSION

For all the datasets studied, it is observed that the methods have a superior performance to the baseline, chosen as matrix factorization in this case. Table 4 shows the methods with best (denoted as #1) and second best results (#2). Also, the table includes the difference between the two best performance metrics (Difference #1 and #2), as well as the difference between the best and worst metric (Difference #1 and #4).

	Jester Jokes	Rate Beer	Movies
#1 model	TF-IDF	BERT	GPT-2
#1 nDCG	0.26278	0.17388	0.46307
#2 model	Word2Vec	GPT-2	BERT
#2 nDCG	0.24255	0.16991	0.46058
Baseline SVD	0.02473	0.02799	0.002620
Difference #1 and #2	0.02023	0.00397	0.0249
Difference #1 and #4	0.02766	0.00530	0.01428

TABLE 4: Summary of results

In all cases, the results are far superior to the baseline, exceeding it by up to ten times. However, it should also be noted that the difference between the results is minimal. Table 4 shows that the difference between the two best results and between the best and worst results is very small. This speaks to the fact that all recommenders perform very similarly in terms of their recommendations.

It follows then that the BERT and GPT-2 methods, which have the ability to capture the context of words in sentences, do not outperform the methods that count words or generate fixed embedding, such as TF-IDF and Word2Vec, respectively. Thus, the thesis that emerges is that context does not play a relevant role in recommendation, at least in the proposed datasets.

In the case of Jester Jokes, it is observed that the methods they recommend best are TF-IDF followed by Word2Vec. One explanation for this may be based on the semantic structure of jokes, which according to Castro (2011) is always oriented towards the end of the joke. In other words, the ending is the most relevant part when making a funny joke. Along these lines, a recommender who understands the context may have no way to take advantage of this, since attention should preferentially be placed on the ending. These simple patterns in endings can be captured by traditional methods, such as TF-IDF and Word2Vec (by GPT-2 and BERT as well, but without the use of context).

Another way to approach the problem is to consider the nature of jokes. Many theorists think that jokes make us laugh because of the difference they embody between what we expect and reality (Berger [3]). In this view, there is a very subjective component to every joke, which can be modeled as a random component. This can also be corroborated in everyday experience: sometimes one does not know why one laughs at something. In that sense, rather than deep semantic patterns, there may be topics that statistically make more people laugh than others. In those cases, it is not necessary to know the context, but rather to find those patterns.

In the case of Rate Beer, the methods with context (albeit by a very small margin) outperform the methods without context. In the case of beers, Attardo [1] mentions that the three most relevant characteristics for choosing beers are: flavor, fermentation type and color. If context-aware recommenders (GPT-2 or BERT) are able to capture these patterns, this could explain why they achieve a better result. However, this thesis does not seem to hold, given the small difference with the results obtained in the case of TF-IDF and Word2Vec. Again, it seems that the patterns are in word repetitions, rather than in contextual and/or semantic factors.

Finally, in the case of Movies, a great improvement of all the proposed methods is observed in relation to the baseline. The best performances are shown by the methods with context (GPT-2 and BERT), as well as in the Rate Beer dataset, although not significantly, which may conclude that studying context is an important factor, but not the only one, in movie recommendation. One explanation could be that the most relevant information for recommendations may be contained in shorter fragments of text and the ability to understand the full context is not as crucial. In addition, there may be subjective or cultural aspects that determine users' taste for movies that cannot be captured by any text-based method. As in the previous datasets, the Word2Vec and TF-IDF methods achieve good performances, with lower computational cost, so they can be taken into account when choosing the method to implement.

Thus, it can be observed that the three datasets show similar behavior: none of the methods have a significantly higher recommendation power than the others. As explained above for each dataset, this seems to be due to the fact that recommenders that involve context (such as GPT-2 and BERT) are not able to incorporate this factor in their recommendations. Moreover, it appears from the results that repetitive and static patterns provide more relevant information for the recommendation. Such patterns can be captured by all the proposed methods, which explains the similar results for the three datasets.

In regards to other works, it's important to mention that for the beer dataset, a mean absolute error (MAE) of 0.5833 and a root mean square error (RMSE) of 0.5833 were achieved. The model used for this was one called the Collective Factor Model (CFM). Comparing the different scores of RMSE, MAE and NDCG is not easy. Having a higher NDCG means a better performance terms of ranking the most relevant items for recommendations. In the other hand, RMSE is used when we want to predict a certain score or rating and we get the difference between our output and the real answer. These metrics, while not directly interlinked, serve as vital indicators signifying the attainment of promising outcomes in predicting scores within recommender systems. They stand as pivotal benchmarks, offering insights into the initial success of predictive models. Leveraging these indicators can

guide iterative refinements in recommendation algorithms, fostering the pursuit of improved recommendation quality.

VI. CONCLUSIONS

The motivation of this work is to compare how different methods of generating text embeddings behave in different domains. This in order to analyze which one captures the most relevant information reliably in order to deliver better recommendations.

To achieve our goal we process the datasets to stay with users that have several elements consumed in common. Then with each model we generate embeddings for these items and then use cosine similarity to find the similarity between the users and the items in the testing set.

For performance we analyze whether the recommended item was delivered in the order of user preference or not. Finally, we get the nDCG of the delivered items and compare with a SVD matrix factorization baseline, with 20 latent factors.

In the experiments carried out, it was observed that all methods significantly improved the baseline, for which a SVD matrix factorization is used. In two of the three datasets, the context-based models (GPT-2 and BERT) perform better, although the improvement is very small. Therefore, when choosing the method, an analysis must be made between accuracy and computational cost, since the context-free models (Word2Vec and TF-IDF) have a very good performance, are much faster and less computationally demanding.

The particularity of the results obtained is that no dataset has an outstanding result with respect to the others. All the methods always perform much better than the baseline, but among them the results are similar. It is then proposed that the results are not mostly influenced by contextual and/or semantic aspects, since, if they were, methods such as GPT-2 or BERT should show better results than TF-IDF and Word2Vec. The thesis is that in the chosen contexts, repetitive patterns correlate better with the data, so that BERT and GPT-2 do not have a significant advantage over the other methods.

As future work, it is proposed to implement new techniques for generating embeddings for text and compare them with the methods obtained. Also, the number of baselines can be increased, adding others such as most popular or random, in order to obtain a more robust comparison. In this way, better correlations can be found or causal relationships can be ruled out. The goal of future work should be to find the best embedding generator in multiple domains and, at the same time, the one that can clearly differentiate itself from the results of its competitors. To this end, in addition to new methods, other datasets from other domains, such as games, restaurants, songs, books, etc., should be tested.

REFERENCES

[1] Salvatore Attardo. The semantics of humor. In The Linguistics of Humor: An Introduction. Oxford University Press, 06 2020.

[2] Matthias Aßenmacher and Christian Heumann. On the comparability of pre-trained language models, 2020.

[3] Arthur Asa Berger. Why we laugh and what makes us laugh:'the enigma of humor'. Europe's Journal of Psychology, 9(2):210–213, 2013.

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

[5] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2020.

[6] Marco Lerro, Giuseppe Marotta, and Concetta Nazzaro. Measuring consumers' preferences for craft beer attributes through best-worst scaling. Agricultural and Food Economics, 8(1):1–13, 2020.

[7] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[8] Tomas Mikolov, Kai Chen, G.s Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. Proceedings of Workshop at ICLR, 2013, 01 2013.

[9] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 26, 2013.

[10] BERT Open Sourcing. State-of-the-art pre-training for natural language processing. Google AI Blog URL: https://ai. googleblog. com/2018/11/open-sourcing-bert-state-of-art-pre. html (accessed: 15.12. 2021), 2019.

[11] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

[12] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. Journal of documentation, 28(1):11–21, 1972.

[13] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2020.

[14] Yuhui Zhang, Hao Ding, Zeren Shui, Yifei Ma, James Zou, Anoop Deoras, and Hao Wang. Language models as recommender systems: Evaluations and limitations. In NeurIPS 2021 Workshop on I (Still) Can't Believe It's Not Better, 2021.