Recommender Systems Driven by NLP and Transformers Models

Angel Vossough[1] and Arsalan Vossough[1]

University of California, Berkeley (Masters in Data Science program)
{avoss, vossough}@Berkeley.edu

Abstract. We implemented, experimented with, and sought to improve the machine learning model presented in "RecoBERT: A Catalog Language Model for Text-Based Recommendations" [12]. RecoBERT is a leading edge application of BERT, aiming to produce meaningful related recommendations within a catalog, driven only by NLP analysis. We include our open source implementation written in PyTorch. We have also prepared a supplementary data set of reviews, to facilitate exact matching between data sets. The original model scaled poorly with catalog size. To address this, we experimented with lightweight BERT models, notably DistilBERT, and evaluated strategies to improve inference time. Finally, we added a knowledge distillation approach. That system demonstrates results comparable with original RecoBERT, but with improved computation times.

Keywords: Natural Language Processing · Recommender Systems · BERT · RecoBERT · Knowledge distillation · DistilBERT .

1 Introduction

Among recent developments in the AI field, BERT [6] is among the most exciting and promising. At the time of writing, the paper has accumulated nearly 30k citations and enabled an entirely new family of transfer learning models. As postgraduate students nearing the end of a Masters in Data Science program, we both considered that developing aptitude with BERT would be of benefit to any career in Data Science.

In appraising RecoBERT [12], we appreciated the flexibility and potential of a system that can train from any unlabelled corpus of reviews, by permuting features to synthesize its own labelled data for fine-tuning. The publicly available wine reviews dataset also seemed interesting to use.

Recommender systems are very often informed by user activity (eg. click history from users collected as analytics). Exploring the design of an exclusively NLP-driven solution was an interesting challenge.

We therefore selected this theme as the most suitable idea for a semester project.

2 Related Work

Text-based recommendation Text-based recommender systems rely strictly on textual features to calculate a similarity score between any two items. The simplest approach for this problem is keyword based, wherein keywords are extracted from the text to represent the item [10,2]. Semantic features may also be considered as an essential factor to understand the item. [5] adopts the advantages of WordNet for including semantic features, while an ontology is used by [14] and [7]. These word based methods are limited when the represented text becomes longer and more complex.

Recent works apply the text embedding technique to capture the meaning of an item's text features. The embedding method can be as simple as Word2Vec [15], or produced via a more complicated neural network such as a transformer or CNN [8]. In this work, we follow the method introduced by [12] which utilizes the popular BERT model [6] to produce embedding vectors for the title and description of an item. Furthermore, we implement other pretrained BERT variant language models [11] and [1].

Knowledge distillation Inspired by [16], we further investigate a knowledge distillation approach to reduce the complexity of the model and therefore improve the inference time. Knowledge distillation was first introduced by [9] and has become a common technique to simplify a deep network. In this work, we follow [3] to create a distilled model for sentence embedding.

3 Recobert Design

3.1 Introduction

Figure 1 we include from [12], showing a useful overview of RecoBERT's components. A brief conceptual description is presented here, followed by a more complete formal presentation.

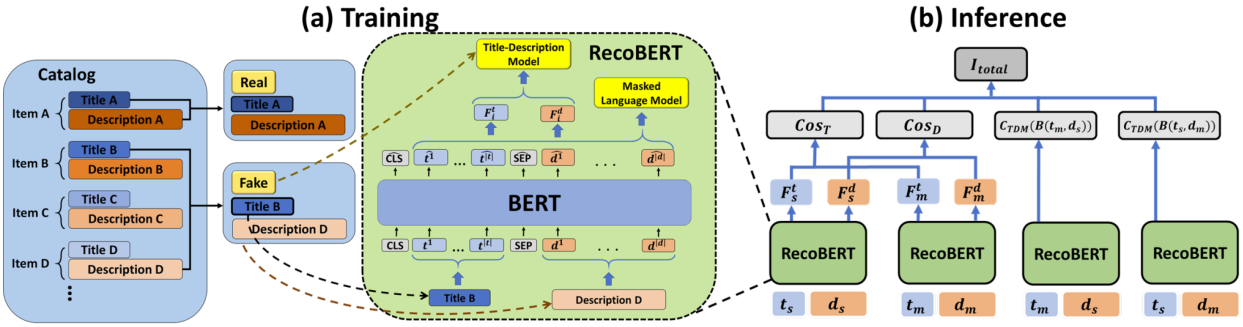

Fig. 1. RecoBERT: A Catalog Language Model for Text-Based Recommendations, (p. 3)[12]

Training From the source data set of wine reviews, we take two features: title, a string which includes the name, variety, producer and the wine's year of production; and description, a brief review.

From these records, we generate our training data, where a label of 1 (Real) indicates that title and description are both from the same review, and a label of 0 (Fake) indicates that we have sourced the title and description from different reviews.

Input textual features are tokenized for the BERT Masked Language Model, randomly masked, and then propagated through it. Word embeddings taken from the final hidden layer are then reduced through mean-pooling to give one embedding vector for the title and another for the description.

Our overall loss function consists of two parts: first we compute a TDM loss (Title-Description Model) based on cosine similarity of the title and description embeddings and considering the label (ie. if a title-description pair is real or fake). To this is summed the MLM loss (Masked Language Model) determined by BERT, giving an overall loss. Training continues so long as the overall loss is meaningfully reduced.

Inference Inference consists of evaluating a seed item against all candidates (ie. each member of the catalog). Four terms are calculated, the first two reflecting cosine similarity between seed title - candidate title and seed description - candidate description; and the last two reflecting a cross comparison: seed title - candidate description and seed description - candidate title. Scores for each term are then normalized across the dimension of candidates to give zero mean and unit variance. All four scores are summed together to give a final recommender score, and this score is then used to rank all candidates (higher is better / more similar).

3.2 Problem formulation

We follow the formulation described by [12]. The ith item of the dataset is represented by a tuple (t_i, d_i) where t_i and d_i are the title and the description of the item respectively. The title $t_i = \{t_1^i, t_2^i, ..., t_n^i\}$ and description $d_i = \{d_1^i, d_2^i, ..., d_m^i\}$ are sequences in which t_j^i and d_k^i are tokens. Let $\mathcal{C} = \{(t_i, d_i), ..., (t_c, d_c)\}$ be the catalog, with a given item (t_j, d_j), the task is to calculate the similarity score between the given item and all other items of the catalog, which can be represented by a similarity function $F((t_j, d_j), \mathcal{C}|\Theta)$. The parameter Θ is learned through training Masked Language Model and Title-Description relation model which is described in more detail below.

3.3 RecoBERT detail

The main target of RecoBERT [12] is to learn the relationship between title and description, therefore the input is the concatenation of title and description text, which is represented as:

$$I(t_i, d_i) = \{[CLS], t_1^i, t_2^i, ..., t_n^i, [SEP], d_1^i, d_2^i, ..., d_m^i, [SEP]\} \tag{1}$$

with the first part is a sequence of title tokens and the second part is description. This formulation is identical to the Next Sentence Prediction task in [6], therefore it can better utilize the capability of the pretrained model.

RecoBERT produces two embedding vectors one for the title and one for the description accordingly:

$$\mathcal{B}((t_i, d_i)) = (F_i^t, F_i^d) \tag{2}$$

with $F_i^t = \frac{1}{n}\sum_{j=1}^{n} h_{t_j}^i$, $F_i^d = \frac{1}{m}\sum_{k=1}^{m} h_{d_k}^i$ are title and description embedding vectors, $h_{t_j}^i$ and $h_{d_k}^i$ are the encoded vectors of token t_j and d_k accordingly.

The relationship between title and description of the item can be naively understood as how the title vector is similar to the description vector. If the title and the description belong to same item, then the cosine similarity between them should be closer to 1; and closer to 0 if the title and the description belong to two different items.

From the original catalog, the negative samples are sampled by swapping the description of another item for each considering item. The final training set includes $|\mathcal{C}|$ examples, each is a tuple (t_j, d_j) and a label:

$$y_j = \begin{cases} 1 & \text{if title } t_j \text{ and description } d_j \text{ correspond to the same item} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Trying to approximate the similarity between title and description to 0 or 1 can ignore the characteristics of similar items which usually have the similar title and description. but we hypothesize that the percentage of similar items is very small, which means that if the dataset is very large, by randomly picking two items, the probability that the two items are similar is approximately 0.

With these hypotheses, firstly the similarity between title and description is defined as :

$$C_{TDM}(F_i^t, F_i^d) = \frac{1 + cos(F_i^t, F_i^d)}{2} \tag{4}$$

and the Title-Description matching loss (TDM) is defined as:

$$\mathcal{L}_{TDM} = -\frac{1}{n}\sum_{i=1}^{n}[y_i log(C_{TDM}(F_i^t, F_i^d) + (1 - y_i) log(1 - C_{TDM}(F_i^t, F_i^d)] \tag{5}$$

The original paper argues that the Masked Language Model (MLM) loss is important, therefore we utilized the MLM loss directly from [6]. The final loss is the linear combination of TDM and MLM loss $\mathcal{L} = \mathcal{L}_{MLM} + \mathcal{L}_{TDM}$

During inference, the similarity between a seed item s and another item m includes 4 factors. The first two factors stand for the similarity between title and the description of the two items, which is calculated by *cosine* similarity. Specifically, the similarity between titles is $Cos_T(s, m) = cosine(F_s^t, F_m^t)$ and $Cos_D(s, m) = cosine(F_s^d, F_m^d)$ is the similarity between descriptions. The other two factors are the relationship between title of seed item s and the description of the considering item m and vice versa. This factor is calculated by 4. The final score is the linear combination of all 4 factors $Score(s, m) = \lambda_1 Cos_D(s, m) + \lambda_2 Cos_T(s, m) + \lambda_3 C_{TDM}(\mathcal{B}(t_m, d_s) + \lambda_4 C_{TDM}(\mathcal{B}(t_s, d_m)$.

The λ_i is predefined. Where lambda constants are mentioned in results, they are either set at 1 (this term is calculated and used) or 0 (not used).

3.4 Knowledge Distillation

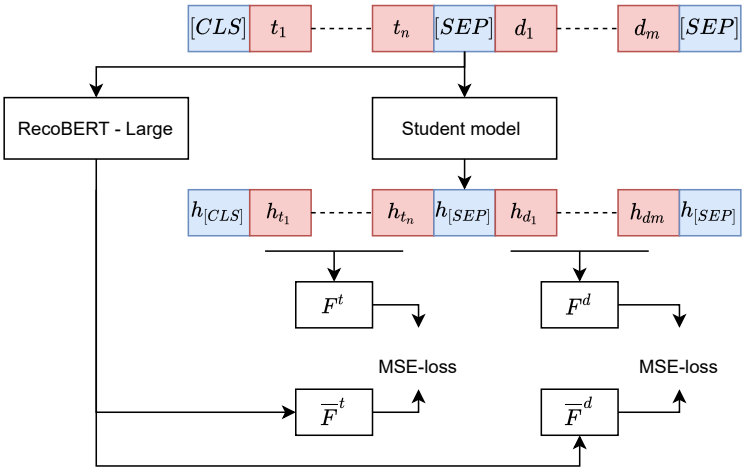

Fig. 2. Knowledge distillation for RecoBERT

The knowledge distillation technique applied for RecoBERT is illustrated in Figure 2. Our distilled student model utilizes the backbone of a BERT-large model, comprising a hidden layer size (embedding dimension)

of 1024 and the number of heads being 16. The model contains 6 layers with the initialized weights extracted from 6 layers $(0th, 4th, 8th, 15th, 19th, 23rd)$ of the teacher which is the trained 24 layer RecoBERT-large model. In addition to $\mathcal{L}_{MLM}$ and $\mathcal{L}_{TDM}$, we included the distillation loss between the output of student and teacher models.

Because the target of knowledge distillation in this case is to make the student model produce embedded vectors that are similar to those produced by the teacher model, inspired by [3], we utilized the mean squared loss as the following:

$$\mathcal{L}_{dist} = \|F^t - \overline{F}^t\|_2^2 + \|F^d - \overline{F}^d\|_2^2 \tag{6}$$

with F and $\overline{F}$ are the embedding vectors of student and teacher models respectively. The final loss is $\mathcal{L} = \mathcal{L}_{MLM} + \mathcal{L}_{TDM} + \mathcal{L}_{dist}$

4 Method

4.1 Implementation

The original paper carefully states most model design and hyperparameter settings. Where these are mentioned, we chose to follow them exactly. This includes a training mini-batch size of 16; Adam optimizer; a 50% probability of a true title-description pair versus a switched title-description pair; a 15% probability of masking any input token; model choice of pretrained BERT-Large; and 1.5 million training steps.

To prepare training and validation sets, we used a 90% and 10% split.

We coded a PyTorch implementation from scratch, having not found any associated code for RecoBERT released by the original authors. At the Hugging Face model repository, we also found no published fine-tuned RecoBERT models.

Our BERT model was configured with a fixed width of 160 tokens. This allowed for faster training as compared to the standard 512-width model, given that the reduced model contains less parameters. Fixed width size was determined by measuring the amount of truncation observed in the tokenized inputs, and ensuring that it happened very infrequently or never. Due to RecoBERT's constant permutation of titles and descriptions during training, and the randomness of lengths that this produces, we accept the occasional truncation to achieve a lean model and better training time.

4.2 Choice of learning rate

Noticing that learning rate schedules were not described in [12], we looked further into this question.

In a successful training, we observed that the MLM loss should very quickly approach 0 and then remain negligible. TDM loss will gradually reduce as training progresses and eventually plateau, at which point training can conclude.

On occasion, we observed that while training BERT-large (and ALBERT-large) with peak learning rates above $2e^-5$, the MLM loss could sharply increase, effectively ruining the model tuning. As mentioned by [13], the model might be affected by catastrophic forgetting. To address this problem, we avoid any learning rates above that level. Our experiments were run (similarly to [13]) with 10k warmup steps to the $2e^-5$ level and a linear decay thereafter.

4.3 Platform and computation times

The platform for all experiments was a Linux instance, using the Google Cloud Deep Learning CUDA PyTorch build for Debian. A single Nvidia A100 GPU was used. As well as time, compute cost was a factor in constraining the number and duration of model trainings that we could prepare.

Our training rate when propagated through BERT-Large of ≈7 it/s exceeded the rate of ≈3.5 it/s (1.5M steps over 5 days) reported in the original paper. This could be due to our use of a newer generation of GPU, or possibly from implementation differences.

Inference times are not performant as catalog size increases. As reported in [12], 9.5 hours to run inference from 100 seed items averages to 6 minutes per item. We have included some experiments toward improving this. Notably, the original authors have also addressed the knowledge distillation path in detail in [3].

4.4 Training duration and early stopping

The original authors of RecoBERT in [12] trained their best model to a duration of 1.5M training steps, that being approximately 200 epochs. Early stopping was used by them during certain trainings, with their exact conditions not specified. In our experiments, we observed that the BERT-Large model had already converged at 100 epochs.

4.5 Evaluation metrics

The original paper provides a small data set of "A is similar to B" recommendations by wine sommeliers. For each seed A, there are about 10 similar wines B listed. Whereas the quality of results produced by a recommender system are often subjective, the expert dataset allows us to quantify performance of the system.

We have used the same metrics as in the original paper: Hit Rate @ k (HR@k); Mean Reciprocal Rank (MRR); and Mean Percentile Rank (MPR).

Hit Rate @ k denotes the frequency where a recommendation in the expert dataset is found in the top k items scored by the model. We use a variety of thresholds [5, 10, 50, 100] in results.

4.6 Datasets

In communications with the authors of the original paper, we have established that the criteria for matching labeled recommendations with the reviews corpus used inexact matching. A match with edit distance of up to 3 or 4 could have been accepted by them. Which would mean that, for example, a 2008 and 2017 vintage of the same vineyard and product could be considered equivalent, even though reviews would necessarily be different.

We completed an additional targeted scrape of published reviews, fetching exact reviews from the original source for all items in the expert labeled set. Having prepared this, we use exact matching to identify the correct review for each wine in the expert labeled set.

Due to the divergence in data sets and matching techniques, we expected our results to differ from those reported in the original paper.

5 Results

We use TF-IDF cosine-based similarity [4] as a simple baseline. The represented text of each wine is the concatenation of its title and description. The similarity score between seed and candidate items is the cosine similarity between their TF-IDF vectors. We also compare the RecoBERT model with the original pretrained models. Detailed results are shown in Table 1.

Table 1. Results on expert dataset. **Bold** is the best score and underline is the second best. *RecoBERT-X ensemble* is the combination of first two terms (λ_1 and λ_2) from the pretrained BERT-X and the last two terms (λ_3 and λ_4) from the corresponding RecoBERT-X.

Model	MPR	MRR	HR@100	HR@50	HR@10	HR@5	Infer-Time
TF-IDF	95.6	71.3	89.4	82.5	53.2	33.2	1s
BERT-base	88.3	52.3	70.5	62.9	36.6	24.5	
BERT-large	88.8	55	73.4	65.5	40.5	26.5	
BERT-large$_{\lambda_1 \leftarrow 0, \lambda_2 \leftarrow 0}$	61.1	12.9	22.6	14.4	5.5	3.6	
BERT-large$_{\lambda_3 \leftarrow 0, \lambda_4 \leftarrow 0}$	92.2	67.3	80.8	74.4	53.7	34.5	
RecoBERT-base	95.4	56.2	86.6	71.8	35.6	21.9	5m02s
RecoBERT-large	95.4	55.9	87.1	71	35.5	22.3	8m50s
DistilRecoBERT	95.5	56.3	87.2	72	35.3	21.8	4m32s
RecoRoberta	94.4	52.5	83.8	66.5	32.3	19.3	5m04
RecoBERT-large$_{\lambda_1 \leftarrow 0, \lambda_2 \leftarrow 0}$	94.8	50.2	85.2	67.3	28.4	16.9	
RecoBERT-large$_{\lambda_3 \leftarrow 0, \lambda_4 \leftarrow 0}$	95.3	54.5	86.2	70.3	34.5	22.9	
RecoBERT-base ensemble	96.4	73.2	89.3	82.2	**57.3**	36.3	5m02s
RecoBERT-large ensemble	**96.5**	**73.4**	**89.5**	**82.9**	56.9	**36.6**	8m50s
RecoBERT-large ensemble fast 25%	93.3	72.7	87.4	81.9	57.2	**36.6**	**3m38s**

It should be noted that the challenge of producing meaningful rankings on the expert labelled dataset is less complex than it may first appear. In the universe of wines, similarity is often drawn by region and tradition of production, or grape variety. Even a less sophisticated machine learning system, such as the TF-IDF cosine similarity system we implemented, could quickly learn to populate top ranks somewhat well by considering, for example, a *Gamay* as similar to other *Gamays*. Indeed, TF-IDF out-performs most of the vanilla BERT models in most metrics, and nearly matches the best model at HR@100 and HR@50. All this while returning its predictions in less than 100th the time of our fastest model.

A larger expert labelled dataset would be necessary to properly discriminate between a good model and a great one. Ideally, we would seek to prove empirically that beyond the headline categories, a review's nuances

describing flavors and characteristics are flowing through to the sentence-level embeddings and consequently the similarity score.

The final line of the table "RecoBERT-large ensemble fast 25%" is an experiment for an abbreviated inference process. The candidate space is first ranked by the sum of λ_1 and λ_2 terms, which are quick to look up from embeddings that can be precalculated without regard to seed item. The candidate space is then pruned to keep only the top $x\%$, and ranking then proceeds using λ_3 and λ_4 terms. At the threshold of $x = 25$, a comparable quality of recommendations was produced with faster inference time.

The row of the table titled "DistilRecoBERT" is an implemention of the knowledge distillation technique. As a student model, DistilRecoBERT's results were similar to the teacher (RecoBERT-Large) model's, while inference time is 60% faster.

6 Conclusion

Our results agreed with the original paper, in that each of the four inference terms contributed to an improvement in the quality of recommendations produced. As would be expected, moving to more lightweight models (BERT-Base, DistilBERT) produced a marginally quicker recommender system at the cost of reduced quality of results.

The original evaluation task lacks sufficient complexity. Evaluation using a candidate space of at least 10k items would give more potential for models' full quality to be revealed.

We found that the design in [12] grows to be heavy on computation due to the need to calculate word embeddings in so many cascades of iterations and then mean-pool them. It would not be suitable for catalogs whose size exceeds ≈10k items.

Ultimately, a knowledge distillation approach, where a student-teacher model leads to directly producing sentence-level embeddings such as in [3], seems to be a more versatile and universally useful design.

References

1. Albert: A lite bert for self-supervised learning of language representations. ArXiv **abs/1909.11942** (2020)
2. wook Ahn, J., Brusilovsky, P., Grady, J., He, D., Syn, S.Y.: Open user profiles for adaptive news systems: help or harm? In: WWW '07 (2007)
3. Barkan, O., Razin, N., Malkiel, I., Katz, O., Caciularu, A., Koenigstein, N.: Scalable attentive sentence-pair modeling via distilled sentence embedding. In: AAAI (2020)
4. Cantador, I., Bellogín, A., Vallet, D.: Content-based recommendation in social tagging systems. In: RecSys '10 (2010)
5. Capelle, M., Moerland, M., Hogenboom, F., Frasincar, F., Vandic, D.: Bing-sf-idf+: a hybrid semantics-driven news recommender. Proceedings of the 30th Annual ACM Symposium on Applied Computing (2015)
6. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: NAACL (2019)
7. García-Sánchez, F., García-Díaz, J.A., Gómez-Berbís, J.M., Valencia-García, R.: Ontology-based advertisement recommendation in social networks. In: DCAI (2018)
8. Gong, Y.H., Zhang, Q.: Hashtag recommendation using attention-based convolutional neural network. In: IJCAI (2016)
9. Hinton, G.E., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. ArXiv **abs/1503.02531** (2015)
10. Jain, S., Khangarot, H., Singh, S.: Journal recommendation system using content-based filtering. Advances in Intelligent Systems and Computing (2018)
11. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. ArXiv **abs/1907.11692** (2019)
12. Malkiel, I., Barkan, O., Caciularu, A., Razin, N., Katz, O., Koenigstein, N.: Recobert: A catalog language model for text-based recommendations. ArXiv **abs/2009.13292** (2020)
13. Mosbach, M., Andriushchenko, M., Klakow, D.: On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. ArXiv **abs/2006.04884** (2021)
14. Obeid, C., Lahoud, I., Khoury, H.E., Champin, P.A.: Ontology-based recommender system in higher education. Companion Proceedings of the The Web Conference 2018 (2018)
15. Özsoy, M.G.: From word embeddings to item recommendation. ArXiv **abs/1601.01356** (2016)
16. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. ArXiv **abs/1910.01108** (2019)