






Transformer-based Language Models for Semantic Search and Mobile Applications Retrieval

João Coelho^{1,4}, António Neto^{1,2}, Miguel Tavares^{1,5} ^a, Carlos Coutinho^{1,2,6} ^b, João Oliveira^{2,6,7} ^c,
Ricardo Ribeiro^{2,3} ^d and Fernando Batista^{2,3} ^e

¹*Caixa Mágica Software, Lisboa, Portugal*

²*ISCTE - Instituto Universitário de Lisboa, Av. das Forças Armadas, Portugal*

³*INESC-ID Lisboa, Portugal*

⁴*Instituto Superior Técnico, Lisboa, Portugal*

⁵*ULHT - Universidade Lusófona de Humanidades e Tecnologias, Portugal*

⁶*Information Sciences, Technologies and Architecture Research Center (ISTAR-IUL), Lisboa, Portugal*

⁷*Instituto de Telecomunicações (IT), Lisboa, Portugal*

Keywords: Semantic Search, Word Embeddings, ElasticSearch, Mobile Applications, Transformer-based Models.


Abstract: Search engines are being extensively used by Mobile App Stores, where millions of users world-wide use them every day. However, some stores still resort to simple lexical-based search engines, despite the recent advances in Machine Learning, Information Retrieval, and Natural Language Processing, which allow for richer semantic strategies. This work proposes an approach for semantic search of mobile applications that relies on transformer-based language models, fine-tuned with the existing textual information about known mobile applications. Our approach relies solely on the application name and on the unstructured textual information contained in its description. A dataset of about 500 thousand mobile apps was extended in the scope of this work with a test set, and all the available textual data was used to fine-tune our neural language models. We have evaluated our models using a public dataset that includes information about 43 thousand applications, and 56 manually annotated non-exact queries. The results show that our model surpasses the performance of all the other retrieval strategies reported in the literature. Tests with users have confirmed the performance of our semantic search approach, when compared with an existing deployed solution.


1 INTRODUCTION


The penetration of mobile devices in society has led most companies to see them as indispensable means for being in close contact with their customers. According to (Iqbal, 2020), Google Play Store has 2.6 million mobile apps available, Apple's iOS App Store has 1.85 million, and Aptoide has about 1 million¹, which creates an extremely tough competition between apps. In terms of app downloads, in 2019 there were 204 billion app downloads, a number that has


been increasing over the years. However, many of these downloads consist of multiple attempts to find the right app. In fact, many downloaded apps are never used, and in 77% of cases, apps are not used again within 72 hours of installation. This shows a big misalignment between the supply of apps by the app stores (distribution services) and the demand for them by the consumers (discovery). Furthermore, around 2019, 52% of apps were discovered by word-of-mouth between acquaintances, friends or family, and only 40% were discovered by searching app stores. These inefficiencies make app discovery and distribution a considerable and extremely relevant challenge, since they take place in a market of massive penetration in societies and seriously affect the relationship between companies and consumers.


The Aptoide's current search engine solution is

^a  <https://orcid.org/0000-0001-6346-0248>

^b  <https://orcid.org/0000-0001-8065-1898>

^c  <https://orcid.org/0000-0003-4654-0881>

^d  <https://orcid.org/0000-0002-2058-693X>

^e  <https://orcid.org/0000-0002-1075-0177>

¹ <https://pt.aptoide.com/company/about-us>

purely lexical, considering only applications names and descriptions as bags-of-words when searching. As such, the objective of this work is to assess whether or not a semantic search engine improves the results achieved by such a model, in this specific scenario. Hence, we describe a dataset containing information about mobile applications, which was previously built by this research team from scratch, and was extended with a test set in the scope of the work here presented. The textual data contained in this dataset was used to fine-tune a neural language model, which is then used in a bi-encoder architecture to independently generate representations for applications and user queries. We compare our model to an adaptation of Aptoide's current solution, by using the public dataset crawled by (Park et al., 2015), which includes information about 43,041 mobile applications and 56 non-exact queries previously annotated. Then, we have built a test platform on top of Elasticsearch, where users could rate which model returned the most relevant applications. From these tests, we were also able to extract relevancy judgments for approximately 150 unique queries, which were used to extend our dataset with a test set, to be used in future evaluations. Both the automatic and the user evaluations show that our semantic approach largely surpasses the existing lexical model.

The outputs of this work, i.e. the dataset and the model, will be made available². This was developed in the scope of the AppRecommender project, which has as its strategic objective to investigate and develop technologies capable of offering the right app, to the right customer, at the right time. Both the dataset and the model will be publicly available for reproducible purposes.

This paper is organized as follows. Section 2 presents an overview of the related literature, focusing on existing work on semantic retrieval. Section 3 presents the data used in our experiments, which consists of the data that we have collected to train or fine-tune our models, and also the dataset that we have used for evaluation. Section 4 describes our set of approaches. Section 5 describes the conducted experiments and the achieved results. Finally, Section 6 summarizes the most relevant contributions, and presents the future work.

2 RELATED WORK

Commercial app stores, like Aptoide, are, in general, based on lexical similarity and use search engines

²<https://apprecommender.caixamagica.pt/resources/>

built on top of Lucene³, such as Solr⁴, or Elasticsearch⁵.

Classic approaches to improve search results has focused on well-know natural language pre-processing steps (stemming and/or lemmatization) followed by a semantic layer that ranged from the use of synonyms or hyponyms (Datta et al., 2012; Datta et al., 2013) to topic models (Blei, 2012) based on the information available (e.g., titles and descriptions of the applications, reviews, etc.) (Zhuo et al., 2015; Park et al., 2015; Park et al., 2016; Ribeiro et al., 2020).

Neural-based word representations have achieved a considerable success in spoken and written language processing (Ghannay et al., 2016), as well as in information retrieval (Yao et al., 2020; Samarawickrama et al., 2017). (Yates et al., 2021) survey methods for text ranking (i.e., score a collection of textual documents with respect to a query) leveraging neural language models. They distinguish between dual-encoders and cross-encoders. The former encode queries and documents independently, performing better temporally, while the latter encode concatenations of queries and documents, generally obtaining better results, but not suitable to search over large collections, given its computational cost. Multiple models can be used as the encoders, for instance InferSent (Conneau et al., 2017) leverages LSTM-based models, whilst, more recently, transformer-based models have been adopted (Reimers and Gurevych, 2019).

3 MOBILE APPLICATIONS DATA

In this section we characterize the data we used in this work, which consists in two datasets with information about mobile applications. The first one was previously built by us, which we use to fine-tune our language models. The other is a public dataset which we use for evaluation.

3.1 Dataset for Fine-tuning Neural Language Models

The dataset used in the scope of this work was built from scratch, by scrapping the Aptoide's API. A first endpoint was used to extract general data about applications, including the title, Aptoide identifier, added date, update date, and a set of statistics. The Aptoide

³<https://lucene.apache.org/>

⁴<https://lucene.apache.org/solr/>

⁵<https://www.elastic.co/>

identifier was then used to query two other endpoints for application-specific information. The second endpoint contains information regarding the developer, required permissions and the description. The third endpoint contains information about the categories associated with the application.

The first endpoint returned information about 499 thousand applications. For those, the second and third endpoints were queried, returning information for 436,969 of them. The observed discrepancy in values occurred, not only due to missing information on the API, but also due to privatization and/or discontinuation of applications.

The relevancy statistics in our dataset include the number of downloads, the average rating, and the total number of ratings of each of application.

This dataset contains information about applications, of which the textual data will be used to fine-tune neural language models. However, we don't have access to any user queries and/or associated relevant applications. As such, we used the dataset characterized in the next subsection for our evaluations. Besides, during user tests described in Section 5.2, we were able to further extend our dataset with 147 queries, each associated with 5.2 relevant applications, in average.

While all applications in our dataset provide useful textual information to fine-tune our language models, after preliminary analysis we came to the conclusion that the majority of the applications in the dataset were not of much value in terms of relevancy. This is supported by the high number of applications with very few downloads, high number of applications without recent updates, and high number of applications with few and low ratings.

As such, when considering which applications to index for search, we consider a subset built based on four heuristics:

1. The top-5000 downloaded applications which were updated in the last 2 years;
2. The top-5000 downloaded applications which were updated in the last 6 months;
3. The top-1000 rated applications, with at least 200 rates and 1000 downloads;
4. The top-750 with more rates, added in the last 3 months, with at least 1000 downloads.

The objective was to consider applications that are widely used, not leaving out recent ones that are being updated constantly. The applications were also filtered to consider only those with descriptions in English. Since the information about the language is not included in our dataset, we used PyCLD3

(Python bindings for Google's CLD3 language detection model) to automatically detect it.

Overall, 5,819 relevant applications were considered for indexation, which are the applications upon which user test searches will be conducted on.

3.2 Dataset for Evaluation

While the dataset described in the previous section contains substantial information on a considerable number of mobile applications, there are no labeled user queries which would allow us for an evaluation. As such, we resort to a public dataset scrapped in the scope of other work in the same subject (Park et al., 2015).

This dataset includes information about 43,041 mobile applications, including relevant textual information such as name, description and category. The dataset also features 56 non-exact queries (i.e., queries with a meaningful semantic context, instead of an application's name). For each one of the queries, 81 applications are labelled with a relevancy score of 0 (not relevant), 1 (somewhat relevant), or 2 (very relevant). These scores were manually annotated.

The authors of the dataset evaluated their approaches by re-ranking the 81 labeled applications for each query.

4 MODELS FOR MOBILE APPS SEARCH ENGINES

As previously stated, one of the objectives of this work is to compare lexical to semantic search in the scope of mobile applications. Hence, in this section we start by describing the lexical and semantic models used for retrieval, also presenting the retrieval setup, which was built on top of Elasticsearch for both models.

4.1 Lexical Model

The lexical model is an adaptation of Aptoidé's current solution, considering the name and description of an application as bags-of-words, and using Elasticsearch for indexing such information. Upon a query q , all indexed applications are scored, and the top- N are retrieved. The Lucene Scoring Function is used to compute the scores over the chosen fields (name and/or description) for an application a , combining them if more than one:

$$\text{score}(q, a) = \sum_{f \in a} \text{LSF}(q, f), \quad (1)$$

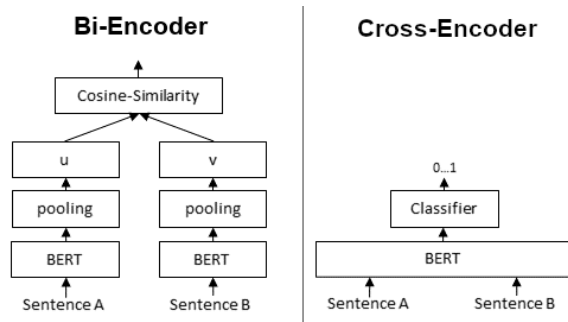


Figure 1: General architecture of a Bi-Encoder (left) and a Cross-Encoder (right), using BERT (Devlin et al., 2019) as the language model.

$$\text{LSF}(q, f) = \frac{1}{\sqrt{\sum_{t \in q} \text{idf}(t)^2}} \times \frac{|q \cap f|}{|q|} \times \sum_{t \in q} \left(\frac{\text{tf}(t, f) \cdot \text{idf}(t)^2 \cdot w_f}{\sqrt{|f|}} \right), \quad (2)$$

where f are application a 's textual fields (in our case, name and/or description), w_f is a boost factor for field f , t are the query's tokens, tf is the term-frequency, and idf is the inverse document frequency.

4.2 Semantic Model

Our semantic model is built on top of RoBERTa (Liu et al., 2019). The fine-tuning procedure is similar to the one we used previously for this task (Coelho et al., 2021), but the access to better hardware (namely GPUs) made it possible to train with larger batch sizes. In this subsection we address the general architecture of the model, the training procedure, and how it was used to generate representations.

4.2.1 The Bi-encoder Architecture

Current state-of-the-art neural models for retrieval leverage transformer-based encoders as either cross-encoders or the bi-encoders (Yates et al., 2021). Cross-encoders (Figure 1⁶, right) receive as input a concatenation of sentences, generating a representation for both through the [CLS] token, for example. This representation can be used to compute a score, for instance by feeding it to a linear layer with sigmoid activation. Conversely, bi-encoders (Figure 1, left) encode sentences separately, which allows the indexation of individual representations, through methods that support fast execution of maximum inner product searches.

⁶Image from: <https://www.sbert.net/examples/applications/cross-encoder/README.html>

Cross-encoders usually achieve higher performances, due to better capturing two-sentence interactions. However, the superior computational efficiency of bi-encoders make them more suitable for a mobile app search engine, as response time is a critical factor in this specific context.

As such, we explore the usage of a fine-tuned neural language model as a bi-encoder in the context of a mobile application search engine, since good results have been reported in other retrieval contexts (Khat-tab and Zaharia, 2020; Qu et al., 2020; Reimers and Gurevych, 2019), and bi-encoders make it possible to pre-compute and index representations for the application's textual data.

4.2.2 Fine-tuning Models

To fine-tune the model, we leverage the textual data in our scrapped dataset, which includes the name and description for approximately 499 thousand applications. We start by fine-tuning RoBERTa_{base} (Liu et al., 2019) on a masked language modelling task, using the Huggingface Transformers library (Wolf et al., 2020). We split our textual data into train and test sets (90% and 10%, respectively). The text in the train set is processed into sets of 512 tokens by RoBERTa's original tokenizer, with a masking probability of 15%. The model was trained during 1 epoch with a batch size of 4, using a cross-entropy loss function. The test set was used to evaluate the fine-tuned model, which achieved a perplexity of 4.54 on a mask prediction task.

Other authors have noted that using out-of-the-box BERT-like models (or even fine-tuned versions on masked language model tasks) is not optimal for downstream similarity tasks, often achieving subpar results (Reimers and Gurevych, 2019). As such, we consider further fine-tuning the model on a semantic similarity task. However, our dataset does not contain any labeled queries, which would be useful for this sort of training. Hence, we consider synthesizing queries by concatenating an application's name with its category. This also extends the applications with semantic context (e.g., from "facebook" to "facebook social"), which is useful for the task in hand.

For training, each query is associated with the description of the application. This way, we build a collection of (query, relevant description) pairs.

Each training batch contained 10 pairs. Batch-wise negative pairing is applied, by using the relevant description of a given query as negatives for the others, increasing the effective training data.

The model is used to generate representations for the query and the descriptions, and scores are obtained from the cosine similarity between the repre-

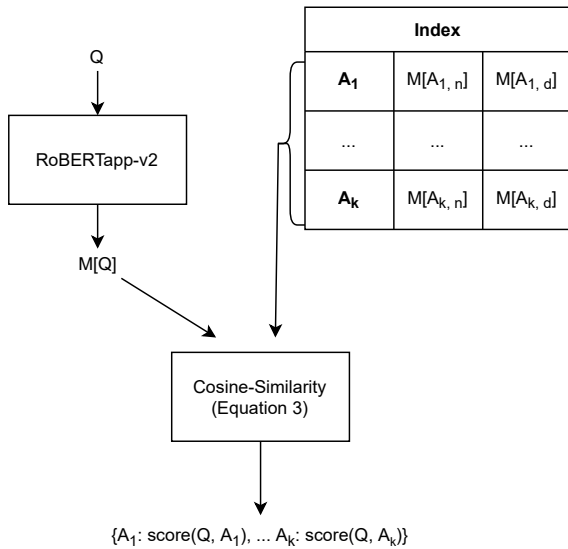


Figure 2: General overview of the search procedure. RoBERTapp-v2 is used to generate representations for the query (Q), and for the applications name and description ($A_{i,n}$, $A_{i,d}$). All applications are scored using Equation 3.

sentations of the query and the descriptions. The objective is to minimize the negative log-likelihood of softmaxed scores. The whole textual dataset is used to train, except for 500 random applications which were used to evaluate. For those applications, queries were built as previously described, and a description corpus was built from their descriptions. The model was used as a dual-encoder to rank the 500 descriptions for each query. Given that each query only has one relevant description, we computed the Precision@1, the Recall@10, and the MRR@10. The results were 0.8929, 0.9809, and 0.9222, respectively.

The final model will be henceforth referred to as RoBERTapp-v2. The major difference in between this model and our previous version, RoBERTapp-v1 (Coelho et al., 2021) is the usage of larger batch sizes.

4.2.3 Generation of Representations and Retrieval

Given a sentence as input, the RoBERTapp-v2 is used to generate representations through mean pooling of the word token embeddings of the last hidden-layer.

For all the applications in the subset to be indexed of our dataset (see Section 3.1), we generate representations for their name and description, indexing them with Elasticsearch.

Upon a query, all the applications are scored and the top- N scoring applications are retrieved. The scoring procedure is depicted in Figure 2. Computing the scores leverages Elasticsearch’s built-in cosine similarity function. Representations for queries are gen-

erated at run-time, using the chosen model. Given a query q and an application a , the score is computed as follows:

$$\text{score}(q, a) = \alpha \text{sim}(M(q), M(a_n)) + \beta \text{sim}(M(q), M(a_d)), \quad (3)$$

where M is the model encoding function, a_n is an application’s name, a_d is an application’s description, sim is the cosine similarity function, and α , β are combination weights. Note that $M(a_n)$ and $M(a_d)$ are already indexed, only $M(q)$ is generated at run-time.

5 EVALUATION AND RESULTS

In this section, we present the results for the base evaluation, which was conducted on top of the dataset scrapped by Park et al., described in Section 3.2. Then, we introduce our platform for user tests, built on top of Elasticsearch, considering the subset of applications for indexation of our dataset (Section 3.1).

5.1 Automatic Evaluation

As previously mentioned, we use the evaluation dataset described in Section 3.2 before analysing the results of the models on our user tests. We use the Normalized Discounted Cumulative Gain as the evaluation metric, which takes the full order of the item list and graded relevance scores into account:

$$\text{DCG}@k = R(1) + \sum_{i=2}^k \frac{R(i)}{\log_2(i)}, \quad (4)$$

$$\text{NDCG}@k = \frac{\text{DCG}@k}{\text{IDCG}@k}, \quad (5)$$

where $R(i)$ is a function that returns the relevance value of the passage at rank i . The index of the passage up to which the ranking is considered is represented by k . The DCG is normalized with the ideal DCG (IDCG), i.e., the DCG of a perfectly sorted result. Table 1 presents the comparison between our previous models and our new fine-tuned RoBERTapp-v2.

Previous work by (Park et al., 2015) had evaluated the results of Google Play and LBDM on this dataset. Also, lexical models similar to ours (extended with topic information) were also tested in this dataset (Ribeiro et al., 2020), using a solution based on BM25F and another with Elasticsearch.

Our previous work (Coelho et al., 2021) had shown that both our first versions of semantic models surpassed previous approaches. From those models, FT2 is a fastText (Bojanowski et al., 2017) model

Table 1: NDCG@{3,5,10,25} for multiple models.

	Model	NDCG@3	NDCG@5	NDCG@10	NDCG@25
(Park et al., 2015)	LBDM	0.584	0.563	0.543	0.565
(Park et al., 2015)	Google Play	0.589	0.575	0.568	0.566
(Ribeiro et al., 2020)	BM25F	0.574	0.542	0.527	0.544
(Ribeiro et al., 2020)	ElasticSearch	0.552	0.532	0.504	0.519
(Coelho et al., 2021)	FT2	0.587	0.589	0.582	0.600
(Coelho et al., 2021)	RoBERTapp-V1	0.616	0.587	0.580	0.605
	RoBERTapp-V2	0.633	0.610	0.601	0.620

fine-tuned on our textual data, and RoBERTapp-v1 is the first version of the semantic model described in this document, trained in a similar fashion but with smaller batches.

The semantic model here discussed, RoBERTapp-V2, achieved the best results so far for NDCG@{3,5,10,25} surpassing both RoBERTapp and FT2.

5.2 User Tests

The results shown in the previous section are already encouraging for the performance of semantic models in the scope of mobile applications retrieval. Nonetheless, we argue that testing with users is an important aspect for this type of system, since, ultimately, user satisfaction dictates the success of mobile application stores.

As such, we conduct user tests with two objectives in mind. First, to assess whether users prefer a lexical model (alike Aptoide's current solution), or a semantic model. Then, we want to be able to extract queries and relevancy judgements from users, so as to extend our dataset with a test set. Hence, a platform was developed on top of ElasticSearch, serving a lexical and a semantic model.

For the tests, we gathered 48 users from diverse backgrounds, all comfortable with writing and reading English text. The tests were performed by the users autonomously without any help or supervision from us, so as to minimize any bias.

The retrieval setup of the tests is as follows: given a query, up to 10 applications are retrieved by each model. The applications are then shuffled in a random order. The users have a checkbox where they can mark which applications are, in their personal opinion, relevant to the query.

We divided the tests in two phases. In the first one, the users would be shown retrieved applications for fixed queries. The objective of this part was to identify if the level of discrepancy between user's answers. For the second part, the same principles apply but users could input free queries to the system. The

objective of this part was to gather as much queries as possible, so as to build the final test set.

5.2.1 Fixed Queries

In the first part, users were asked to deliver relevancy judgements for six fixed queries. We needed queries that could be useful for testing the semantic capabilities of a semantic search engine (hence, searching for explicit application names was considered not relevant). Also, although the platform allows the users to retrieve more information over each application, we wanted the results returned by those queries to be as widely-recognized as possible. As such, the following queries were chosen:

- social networks;
- guitar playing;
- fitness;
- read comics;
- brain challenge;
- food at home.

5.2.2 Free Queries

In the second part, the users could input queries of their choosing, evaluating the relevancy of the returned applications as previously. Here, the base number of free queries was 4, which yields a total of 10 labeled applications per user. However, in the end of the test, the users were prompted to answer more free queries, if they felt like doing so.

5.2.3 Results

After all the users answered, we proceeded to analyze the gathered data. The average time of completion was 10 minutes and 30 seconds. All users were from Portugal. 83% of the users fully completed the test (i.e., answered to at least 10 queries). From those, 20% did more free queries. Overall, the users answered to 471 queries in total.

As for the models, the major conclusion is that the users favored our semantic model. Table 2 presents the overall votes each model was given, i.e., the total number of relevant apps each model retrieved, and the approximate percentage this number represents, as deemed by the users:

Table 2: Results of the User Tests, comparing the total number of relevant apps retrieved for the semantic and the lexical mode, as well as the percentage value.

Model	Total Relevant Apps	Percentage
Lexical	1243	26%
Semantic	1828	39%

This roughly translates that, in average, for 10 retrieved apps, a lexical model yields 2.6 relevant ones, while our semantic model yields 3.9. Given that our models, besides being able to retrieve applications given a semantic context, can also retrieve applications by their exact names, we take the above result difference as a significant upgrade.

Analyzing the answers to the fixed queries, we were able to identify that user answers were very close to each other, with an average answer similarity of 75%. From this, we can more confidently assume the answers of users to their fixed queries as ground truth for future evaluations. For this, the extracted dataset contains 147 unique queries. Each query has, on average, 5.2 relevant labeled applications.

The majority of these queries were rated by only one user. Hence, we could only extract binary relevance (i.e., for a query, app A is relevant), instead of graded relevancy (i.e., for a query, app A is more relevant than app B). As such, we consider the Mean Reciprocal Rank as the evaluation metric for this dataset, given by:

$$\text{MRR}@k = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}(i,k)}, \quad (6)$$

where Q is the set of queries, and $\text{rank}(i,k)$ is a function that, given the top- k results for the i^{th} query, returns the position (rank) of the first relevant passage.

Table 3: MRR@10 of the lexical and semantic models in our test set.

Model	MRR@1	MRR@5	MRR@10
Lexical	0.394	0.466	0.471
Semantic	0.524	0.586	0.592

As expected from the previous analysis, the semantic model achieves the best results on this dataset in all MRR cuts, when compared to the lexical model.

However, it is worthy to note that since the applications to be labeled were retrieved from the top-10 yielded by these models, the results may be somewhat biased towards these models. Nonetheless, the comparison between the two still holds.

6 CONCLUSIONS AND FUTURE WORK

This work evaluates the performance of semantic search when applied to mobile applications. We have tested several models using a public dataset of about 43,000 applications and the results achieved have shown that the proposed model surpasses all the other tested models. We have also conducted a questionnaire about the results produced by each one of the tested models, using real users. The results achieved clearly show that the proposed model produces more relevant results than the lexical-based search currently being adopted by a number of mobile application stores, including Aptoide. We characterized a new dataset, previously built by (Coelho et al., 2021), and extended it in the scope of this work through the user tests. Moreover, we fully describe the training setup for our semantic model, which was inspired by recent developments in Machine Learning, Information Retrieval, and Natural Language Processing. Both the model and the dataset will be made publicly available for research purposes.

In the near future, we plan on building a multi-criteria retrieval system, where other information present in our dataset (such as relevancy statistics, for example) can be also used as a relevant criteria. Also, we plan on extending the test set with more queries and more relevant apps per query. Finally, we will be testing the importance of semantic similarity between the textual information of the applications, studied in this document, for recommendation systems being built in the scope of our project.

ACKNOWLEDGEMENTS

This work was supported by PT2020 project number 39703 (AppRecommender) and by national funds through FCT – Fundação para a Ciência e a Tecnologia with reference UIDB/50021/2020.

REFERENCES

- Blei, D. M. (2012). Probabilistic topic models. *Commun. ACM*, 55(4):77–84.
- Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of ACL*, 5:135–146.
- Coelho, J., Neto, A., Tavares, M., Coutinho, C., Oliveira, J., Ribeiro, R., and Batista, F. (2021). Semantic search of mobile applications using word embeddings. In *Proc. of SLATE 2021*.
- Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. In *Proc. of EMNLP 2017*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.
- Datta, A., Dutta, K., Kajanan, S., and Pervin, N. (2012). Mobilewalla: A mobile application search engine. In Zhang, J. Y., Wilkiewicz, J., and Nahapetian, A., editors, *Mobile Computing, Applications, and Services*, pages 172–187, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Datta, A., Kajanan, S., and Pervin, N. (2013). A Mobile App Search Engine. *Mobile Networks and Applications*, 18.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. of NAACL 2019*.
- Ghannay, S., Favre, B., Estève, Y., and Camelin, N. (2016). Word embedding evaluation and combination. In *Proc. of LREC 2016*, pages 300–305, Portorož, Slovenia. ELRA.
- Iqbal, M. (2020). App download and usage statistics (2020). web page.
- Khattab, O. and Zaharia, M. (2020). Colbert: Efficient and effective passage search via contextualized late interaction over BERT. In *Proc. of SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 39–48. ACM.
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized BERT pre-training approach. *CoRR*, abs/1907.11692.
- Park, D. H., Fang, Y., Liu, M., and Zhai, C. (2016). Mobile app retrieval for social media users via inference of implicit intent in social media text. In *Proc. of the 25th ACM Int. on Conf. on Information and Knowledge Management, CIKM '16*, page 959–968, New York, NY, USA. ACM.
- Park, D. H., Liu, M., Zhai, C., and Wang, H. (2015). Leveraging user reviews to improve accuracy for mobile app retrieval. In Baeza-Yates, R., Lalmas, M., Mofat, A., and Ribeiro-Neto, B. A., editors, *Proc. of the 38th Int. ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, pages 533–542. ACM.
- Qu, Y., Ding, Y., Liu, J., Liu, K., Ren, R., Zhao, X., Dong, D., Wu, H., and Wang, H. (2020). Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. *CoRR*, abs/2010.08191.
- Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proc. of EMNLP 2019*. Association for Computational Linguistics.
- Ribeiro, E., Ribeiro, R., Batista, F., and Oliveira, J. (2020). Using topic information to improve non-exact keyword-based search for mobile applications. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 373–386, Cham. Springer International Publishing.
- Samarawickrama, S., Karunasekera, S., Harwood, A., and Kotagiri, R. (2017). Search result personalization in twitter using neural word embeddings. In Bellatreche, L. and Chakravarthy, S., editors, *Big Data Analytics and Knowledge Discovery*, pages 244–258, Cham. Springer International Publishing.
- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. In *Proc. of EMNLP 2020: System Demonstrations*, pages 38–45. Assoc. for Computational Linguistics.
- Yao, J., Dou, Z., and Wen, J.-R. (2020). Employing personal word embeddings for personalized search. In *Proc. of SIGIR '20, SIGIR '20*, page 1359–1368, New York, NY, USA. ACM.
- Yates, A., Nogueira, R., and Lin, J. (2021). Pretrained transformers for text ranking: BERT and beyond. In *Proc. of WSDM '21, Virtual Event, Israel*, pages 1154–1156. ACM.
- Zhuo, J., Huang, Z., Liu, Y., Kang, Z., Cao, X., Li, M., and Jin, L. (2015). Semantic matching in app search. In *Proc. of WSDM '15, WSDM '15*, page 209–210, New York, NY, USA. ACM.