# Improve retrieval of regional titles in streaming services with dense retrieval

Bhargav Upadhyay
Amazon
India
upbharga@amazon.com

Tejas Khairnar*
Amazon
India
tejaskhairnar7@gmail.com

Anup Kotalwar
Amazon
India
kanup@amazon.com

## ABSTRACT

Customers search for movie and series titles released across the world on streaming services like primevideo.com (PV), netflix.com (Netflix). In non-English speaking countries like India, Nepal and many others, the regional titles are transliterated from native language to English and are being searched in English. Given that there can be multiple transliterations possible for almost all the titles, searching for a regional title can be a very frustrating customer experience if these nuances are not handled correctly by the search system. Typing errors make the problem even more challenging. Streaming services uses spell correction and auto-suggestions/auto-complete features to address this issue up to certain extent. Auto-suggest fails when user searches keywords not in scope of the auto-suggest. Spell correction is effective at correcting common typing errors but as these titles doesn't follow strict grammar rules and new titles constantly added to the catalog, spell correction have limited success.

With recent progress in deep learning (DL), embedding vectors based dense retrieval is being used extensively to retrieve semantically relevant documents for a given query. In this work, we have used dense retrieval to address the noise introduced by transliteration variations and typing errors to improve retrieval of regional media titles. In the absent of any relevant dataset to test our hypothesis, we created a new dataset of 40K query title pairs from PV search logs. We also created a baseline by bench-marking PV's performance on test data. We present an extensive study on the impact of 1. pre-training, 2. data augmentation, 3. positive to negative sample ratio, and 4. choice of loss function on retrieval performance. Our best model has shown **51.24%** improvement in Recall@16 over PV baseline.

## CCS CONCEPTS

• **Information retrieval** → **Dense retrieval**; **Title retrieval**; semantic matching; • **Artificial Intelligence** → Deep neural networks.

## KEYWORDS

Title retrieval, dense retrieval, transliterations, semantic matching

*Work done while author's internship at Amazon

**ACM Reference Format:**
Bhargav Upadhyay, Tejas Khairnar, and Anup Kotalwar. 2023. Improve retrieval of regional titles in streaming services with dense retrieval. In *Companion Proceedings of the ACM Web Conference 2023 (WWW '23 Companion), April 30-May 4, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3543873.3587619

## 1 INTRODUCTION

Streaming services like PV, Netflix etc. have huge collection of movies and series titles released across the world. The primary interface for such platform is in English. For many regions where English is not the primary language, name of the native titles released in these regions are transliterated to English. Interestingly there are multiple transliterations possible for almost all the titles, which makes searching these titles by their title name an interesting and important problem. Along with transliteration variation, common typing errors by customers makes it even more challenging. These streaming services use auto-suggest or auto-complete feature to guide the customer searches on their platform by providing the possible titles based on the prefix, which helps to avoid the noise because of the transliteration variation and/or typing mistakes. But it fails when customer searches for a title with some transliteration variation and/or typing error which is not part of the auto-suggest.

Figure 1 shows a basic search system architecture used by the streaming services. A customer query passes through 3 stages, pre-processing, matching, and ranking. Pre-processing step includes spell correction, stemming, lemmatization etc. Matching step retrieves titles from the search index by means of lexical matching, semantic matching and behavioral matching. Lexical matching retrieves titles based on text matching. It also uses term-based retrieval models such as BM25 [20], TFIDF [22]. Semantic matching retrieves titles which are semantically similar to the query. Behavioral matching retrieves titles based on past customer engagement with titles for the given query. All these retrieved titles are ranked in the ranking stage and produces the final list of titles in the ranked order. One important observation here is that if matching stage fails to retrieve a title from the search index then it won't be part of the final ranked output list. Semantic matching is traditionally used to retrieve semantically similar data [16, 18], for e.g. in case of media, 'bollywood movie' and 'hindi movies' are semantically similar and semantic matching can help to retrieve bollywood movie titles for query 'hindi movies'. In our work we are expanding the scope of the semantic matching to retrieve titles for queries with transliteration variations and typing errors. For e.g. it should retrieve title 'Ramprasad Ki Tehrvi' for query 'raamprsaad ki tervee'. We accomplish it by retrieving the most similar titles to the query based on the embedding vector similarity between query embedding vector and title embedding vector.

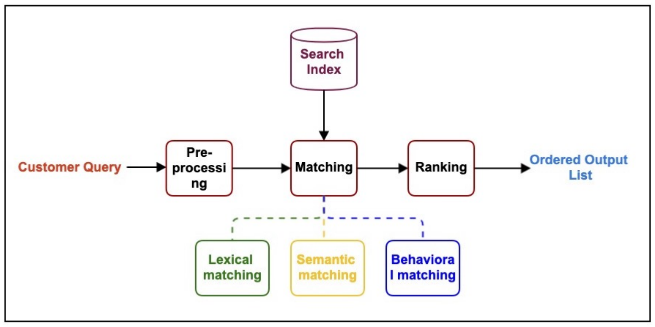

**Figure 1: Basic search system architecture**

One can argue to tackle the problem of transliteration variation by means of spell correction as most of the search systems uses spell correction to correct spelling errors in an input query. Traditionally spell correction is treated as a sequence to sequence (seq2seq) machine learning task [9]. While research has shown that these spell correction system works well in correcting the common typing mistakes [9] but we argue that these spell correction systems performs poorly in streaming services because many movie and series titles have proper nouns and doesn't follow strict grammar rules while transliterating titles from native language to English. Another limitation of existing spell correction system is that we need to retrain the model once vocabulary or possible correct words are updated. Given the fact that there are many new titles are being added to the catalog constantly, it requires frequent re-training where as semantic matching based solution can work out of the box without any re-training.

One major challenge in applying semantic matching is the lack of labelled data. To the best of our knowledge there isn't any existing dataset with the map of corrupted media titles with the correct title. In this work, we propose a novel approach to prepare the data from PV titles and search logs. We built a dataset of 40K query title pairs. Our other contributions are, to derive the impact of data augmentation, ratio of positive to negative samples, and loss function on model's ability to generate sentence embeddings with high similarity between corrupt query and relevant titles and high dissimilarity with irrelevant titles.

This paper is structured in following way, In section 2 we summarise related research. In section 3 we discuss dataset preparation strategy, model architecture, loss functions in detail. Section 4 have detail about the set of experiments we performed and section 5 summarises the experiment results. In section 6 we derive conclusion based on our experiments and discuss future work.

## 2 RELATED WORK

Early research in dense retrieval [3] field used different word embedding techniques [17] to create the dense vector representation of query and documents. With recent progress in representation learning, most of the new research [5] in the dense retrieval, focuses on optimum ways to capture the important details of the documents in the dense vector representation [10]. It is achieved by using different neural network architecture [10] or a novel training strategy [7].

Dense retrieval has shown significant performance gain in several fields like e-commerce [16], social media [8] etc. As per best of our knowledge, all the research in the field utilises existing monolingual [2, 13, 15, 23] or multilingual [19] datasets to evaluate new ideas. But as discussed by Priyanka et al. [16], collecting parallel corpus of data manually can be a very time consuming, expensive and nonscalable. This represents friction in using any state of the art research for practical purpose. Priyanka et al. [16] has some similarity to our work as they have tried to use customer logs to generate training data.

Since our work focuses on utilising dense retrieval to address transliteration noise and typing errors for media titles, we don't find previous research with high similarity. Though our experiment design is influenced by research related to zero-shot transfer [26], negative sample sampling strategies [7], loss function selection for fine-tuning [6, 18].

## 3 PROPOSED SOLUTION

In this section, we formally define the task of dense retrieval. We discuss the details of neural network architecture, model training procedure, and different loss functions used in the experiments. We also provide information about dataset preparation which one of the important contribution of this work.

### 3.1 Problem definition

Given a set of movie or series titles $T = \{t_1, t_2, ..., t_k\}$, build a vector space $S = \{E_{t_1}, E_{t_2}, ..., E_{t_k}\}$. Where $E_{t_n} \in \mathbb{R}^m$, is a dense vector representation of title $t_n$, constructed by passing the title $t_n$ through title encoder neural network (NN) as shown in Figure 4. For a customer query $Q$, get the dense vector embedding $E_Q \in \mathbb{R}^m$ by passing it to the same NN query encoder. Retrieve most relevant titles for input query $Q$ by retrieving the most similar title vectors from $S$. We have used cosine similarity to measure the vector similarity. For our experiments, we have used Pythorch's cosine similarity function.

### 3.2 dataset preparation

We explored the existing datasets [2, 13, 15, 23] used for dense retrieval research, but didn't find a dataset which can be used to evaluate our hypothesis. Acquiring labelled data for such use-case is often expensive and time consuming. Historically many researchers [4, 12, 14, 21] have generated synthetic data by applying transformations to the customer queries. Researchers have also used customer behavioral data like clicks, purchases to generate datasets [16]. We collaborated with PV to get catalog and search logs data, because of restrictions on the fields that can be shared in search logs, we couldn't use the customer behavioral data like clicks, streams, purchases, etc to generate labelled data.

To generate labelled data, we took the route of applying different transformation on the movie and series titles. As shown in figure 2, we take movie and series titles and pass it through transformation module. The details about transformations is discussed in section 3.2.1. We then use the actual customer queries corpus, to keep only the transformations searched by the customers.

We collected 42K movie and series titles from PV. After transformations and filter step we left with nearly 40K samples of searched queries with transliteration variations and typing errors and their corresponding correct titles. We used 16 days worth of customer search data to prepare this data.

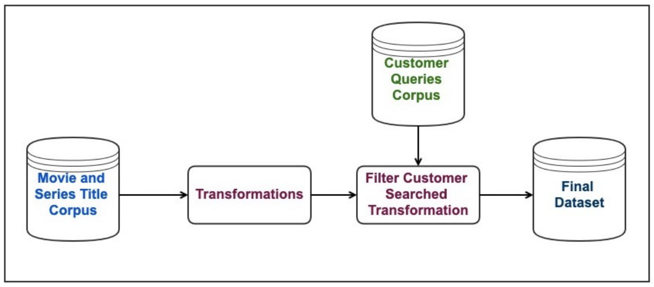
Customer Queries Corpus
Movie and Series Title Corpus
Transformations
Filter Customer Searched Transformation
Final Dataset

**Figure 2: dataset preparation procedure**

*3.2.1 Transformations.* We considered 7 kind of errors to generate title variations.

*Common keyboard errors.* This error happens because of the proximity of the keys on the keyboard. For e.g. it is very common to type 'a' instead of 's', type 'r' instead of 'e'. Though the probability to mistakenly type any other character is not same for all characters. For e.g. it is very unlikely to type 'z' instead of 'p'. To calculate this probability distribution for each character, we referred to this article [1] to get the data for each character. For each character, we calculated the probability of mistakenly typing another character. For e.g. $P(a|s), P(d|s)$ etc. The exact mapping for all characters is present at table 1 in Appendix 1.

*Missing character.* This error happens when user misses to type one or few characters in the query. For e.g. typing 'hary potter' or 'hary poter' instead of 'harry potter'.

*Transliteration.* This error happens because customer can use a transliteration variation different than the movie or series title. For e.g. 'puspa', 'pooshpaa', 'puspaa' are the possible variations for title 'pushpa'. Appendix 1 contains all the transliteration variations considered for preparing the dataset.

*Transpose adjacent characters.* This error happens when two adjacent characters are swapped with each other. For e.g. 'saegmaker' instead of 'sagemaker', 'wehn' instead of 'when' etc.

*Dropping space character.* This error happens when space is missed between two words. For e.g., 'spiderman' instead of 'spider man', 'harrypotter' instead of 'harry potter'.

*Numbers to words.* This variation happens when numeric value is typed in word. For e.g., 'harry potter two' instead of 'harry potter 2', 'fast and furious five' instead of 'fast and furious 5'.

*Suffix to the titles.* This happens when customer types extra suffixes like movie, series after the title name. For e.g. 'braking bad series' instead of 'breaking bad', 'pushpa movie' instead of 'pushpa'.

*3.2.2 Transformations algorithm.* We described 7 different kind of transformation we considered in data preparation. We generate the data by applying these transformation independently. We also generate data by combining, common keyboard errors, missing character and transliteration based transformations. We apply number to word and suffix to the titles transformation directly to the titles as well as the data generated from the other transformations. Here is the procedure we use to combine common keyboard errors, missing character and transliteration based transformations.

(1) Update the map prepared for common keyword errors by adding empty string for missing character and possible transliteration variation for each character. This map is present in Appendix 1. For e.g. map for 'a' becomes {'q' : 594, 's': 42401, 'w' : 10853, 'x' : 3822, 'z' : 3062, '' : 12164 , 'aa': 12164 }. The count for empty string and transliteration variation is the mean of keyboard error counts.
(2) Decide weight for each transformation. For e.g. 1:2:1, 2:2:5. We empirically use 3 different combinations, 1:2:1, 1:3:1, 2:2:5, to generate data. We kept the data generated from each combination but weight 1:3:1 has highest number of samples matched with customer query logs.
(3) Multiply the weight of each class to the replace count map produced in step1. For e.g. for weight 1:2:1 the map for 'a' becomes, {'q' : 594, 's': 42401, 'w' : 10853, 'x' : 3822, 'z' : 3062, '' : 24328 , 'aa': 12164 }.
(4) Normalise the map such that total probability for possible replacements for each character becomes 1. For e.g. map for 'a' becomes, {'q' : 0.006, 's': 0.436, 'w' : 0.111, 'x' : 0.039, 'z' : 0.031, '' : 0.250 , 'aa': 0.125 }.
(5) Randomly choose characters to be replaced in the original title. We randomly choose, number of replacements based on the length of the title. Empirically, we choose a number from $(1, max(len(title)/5, 1))$.
(6) For each character sampled in step 5, randomly sample replacement character based on the probability distribution computed in step 4.

While performing individual transformation for common keyboard errors, missing character and transliteration based transformations, we follow this same procedure, but only keep the weight of the remaining 2 types as 0.

For Dropping space character, Numbers to words, Suffix to the titles variations, we selected original as well as the titles transformed by other transformation and applied these transformation independently.

## 3.3 Model

For our experiments we have used distilled pre-trained model proposed by Wang et al. in [25]. Given the scale of the streaming services, using BERT-base model is more expensive and leads to higher latency, so we decided to used distilled pre-trained model which has shown comparable performance to BERT-base on GLUE benchmark [24] NLP tasks [25]. We have experimented with 2 different

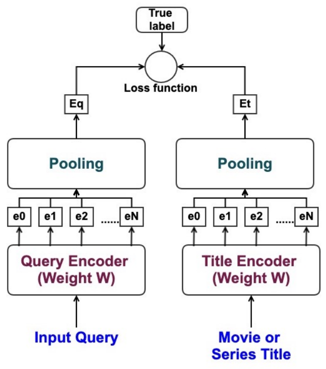

**Figure 3: Model fine tuning procedure**

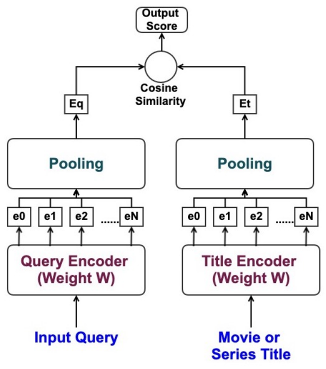

**Figure 4: Model inference procedure**

versions of the distilled model, **1. multi-qa-MiniLM-L6-cos-v1** [1] , **2. all-MiniLM-L6-v2** [2] .

Model **multi-qa-MiniLM-L6-cos-v1** (referred as **Model 1** from now on) is pre-trained 215M question, answer pairs data. It can support maximum sentence length of 512 tokens each with dimension of 384. Model **all-MiniLM-L6-v2** (referred as **Model 2** from now on) is pre-trained using large and diverse dataset of over 1 billion data pairs. It can support maximum sentence length of 256 tokens each with dimension of 384. Both of these models are consist of 6 layers of multi-head attention.

3.3.1 *Model training.* Figure 3 shows model training procedure. We use siamese [11] strategy to fine-tune the pre-trained NN. To measure the impact of the loss function of sentence embeddings, we experimented with 2 loss functions, cosine similarity based Mean Squared Error (MSE) loss and contrastive loss.

*MSE loss.* is represented as

$$L_{MSE}(\hat{y}, E_q, E_t) = (\hat{y} - \cos(E_q, E_t))^2 \tag{1}$$

*Contrastive loss.* is represented as

$$L_{Contrastive}(\hat{y}, E_q, E_t) = \hat{y}\frac{{D_w}^2}{2} + (1-\hat{y})\frac{(\max(0, m - D_w))^2}{2} \tag{2}$$

Here $\hat{y}$ is the true label, $\cos(E_q, E_t)$ is cosine similarity score between query sentence embedding $E_q \in \mathbb{R}^m$ and title embedding $E_t \in \mathbb{R}^m$.

$D_w(cosinedistance) = 1 - \cos(E_q, E_t)$, $m(margin)$ = Negative samples (label == 0) should have a distance of at least the margin value. For training We used default parameters as, $m = 0.5$, batch-size = 16, Adam optimizer with learning rate $\alpha = 2 * e^{-5}$, and a linear learning rate warm-up over 10% of the training data. We used mean pooling strategy to combine input token embeddings into single embedding vector.

## 4 EXPERIMENTS

### 4.1 Experiment dataset

We divide the **40K** positive samples generated from data preparation into 2 parts, **33K** for training and **7K** for test. We add **5K** titles with no variations to the train dataset, meaning 5K samples where query and title are exact match. This makes total **38K** positive samples in the train data. For all our experiments, these 38K samples are used in the training data. For data augmentation experiments, we generate more positive samples and append it with these 38K samples.

### 4.2 Negative samples

To generate negative samples, we consider two types of sampling strategies. **Type 1** sampling, randomly samples titles with maximum length difference of 1 from the target title. **Type 2** sampling samples randomly from the corpus of 42K titles. In our experiments, we avoided edit distance based strategy to sample negative samples mainly because, for queries with spell errors, even a possible strong candidate title can be labelled as 0. It can create a confusing dataset, where different labels are assign for similar examples. For e.g. for a query 'despicable me 2' it will mark 'despicable me' as label 0 and we don't want to learn embedding where similar titles are placed at a distance. Though sampling strategies we used are not 100% full proof from such errors but probability of getting very closely related titles reduces significantly.

### 4.3 Model evaluation

To evaluate the model, we measure Recall@1, Recall@5, and Recall@16 for the 7K test data. We use cosine similarity to retrieve the closest titles from the corpus of 42K titles.

### 4.4 Baseline model

To build a baseline model, we fine-tuned Model 1, keeping positive to negative samples ratio 1:1 and no data augmentation. We used the 38K positive samples, and generated 38K negative samples using the strategy described in 4.2.

### 4.5 Label ratio

We experimented with different the positive to negative sample ratio. We tried 1:1, 1:5 and 1:10 ratios for Model 1 to compare the retrieval performance.

[1]Model details and weights available at: https://huggingface.co/sentence-transformers/multi-qa-MiniLM-L6-cos-v1
[2]Model details and weights available at: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

### 4.6 Data augmentation

As mentioned in 2, after performing transformations we filter all the queries based on the actual customer search logs. To measure the impact of data augmentation, we removed the filter step and kept all the variation generated by the transformation steps. We trained model with 228K, 1.2M, 3M, 4M samples, in each case the positive to negative sample ratio is 1:5.

## 5 RESULTS

Table 1 shows the impact of data augmentation. We trained Model 1 with different training samples using MSE loss. We observed consistent improvement in retrieval performance till 3M samples. After that the performance started to saturate. It is interesting to see that as we go for data augmentation, we included title variations which are not present in the customer search logs, but still it improved Recall@16 by **13.1%** compared to model trained with 228K samples. For this experiment, we use positive to negative sample ratio as 1:5.

**Table 1: Impact of data augmentation, computed with Model 1, MSE loss**

| Train samples | Recall@1 | Recall@5 | Recall@16 |
|---|---|---|---|
| 228K | 42.38% | 57.65% | 68.1% |
| 1.2M | 49.27% | 64.53% | 73.97% |
| 3M | 55.92% | 72.55% | 81.13% |
| 4M | 56.04% | 72.8% | 81.2% |

Table 2 shows the impact of positive to negative sample ratio. Increasing the ratio from 1:1 to 1:5 shows **11.49%** Recall@16 improvement. But further increase from 1:5 to 1:10 shows Recall@16 improvement of only **1%**. This experiment is performed by fine-tuning Model 1.

Table 3 shows impact of loss function. We trained Model 2 with MSE and contrastive loss functions and observed performance improvement of **4.26%**. This shows that training loss function have huge impact on retrieval performance.

**Table 3: Impact of loss function, computed with Model 2 with 4M samples**

| Loss function | Recall@1 | Recall@5 | Recall@16 |
|---|---|---|---|
| Mean Squared Error (MSE) | 56.25% | 72.18% | 81.16% |
| Contrastive Loss | 60.68% | 77.47% | 85.38% |

Model 1 and Model 2 are pre-trained different datasets. After fine tuning both on 4M samples with MSE loss, Model 1's Recall@16 is 81.2% and Model 2's Recall@16 is 81.16%. It shows that different pre-trainng strategy doesn't have noticeable impact on final performance.

Finally table 4 shows the comparison of PV production system, pre-trained Model 2 and fine tuned model. Result shows, that PV is highly sensitive to transliteration variations and typing errors. Pre-trained model trained on different tasks shows reasonable performance but after fine-tuning we observe performance improvement of **34.34%** on Recall@16. Compared to PV baseline, fine-tumed model improved Recall@16 by **51.25%**.

## 6 CONCLUSION

In this paper, we discuss the importance of handling the transliteration variations and typing errors for title search on streaming services. We evaluate the performance of PV on queries with such noise and found that for only 34.13% of the queries have desired title present in the results. We propose to address it by means of dense retrieval. In the absent of relevant dataset, we prepared a dataset of around 40K samples using query transformations and customer search logs. We extensively study the impact of data augmentation, negative to positive sample ratio, loss function used for fine-tuning, and pre-training strategy on retrieval performance. Our best model showed the performance improvement of **51.25%** for Recall@16 compared to PV.

## REFERENCES

[1] Nick Berry. 2013. *Sloppy Typing.* https://datagenetics.com/blog/november42012/index.html

[2] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 deep learning track. *CoRR* abs/2003.07820 (2020). arXiv:2003.07820 https://arxiv.org/abs/2003.07820

[3] Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth J.F. Jones. 2015. Word Embedding Based Generalized Language Model for Information Retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Santiago, Chile) *(SIGIR '15)*. Association for Computing Machinery, New York, NY, USA, 795–798. https://doi.org/10.1145/2766462.2767780

[4] Shaona Ghosh and Per Ola Kristensson. 2017. Neural Networks for Text Correction and Completion in Keyboard Decoding. *CoRR* abs/1709.06429 (2017). arXiv:1709.06429 http://arxiv.org/abs/1709.06429

[5] Jiafeng Guo, Yinqiong Cai, Yixing Fan, Fei Sun, Ruqing Zhang, and Xueqi Cheng. 2022. Semantic Models for the First-Stage Retrieval: A Comprehensive Review. *ACM Trans. Inf. Syst.* 40, 4, Article 66 (2022), 42 pages. https://doi.org/10.1145/3486250

[6] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality Reduction by Learning an Invariant Mapping. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* 2 (2006), 1735–1742.

[7] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. *CoRR* abs/2104.06967 (2021). arXiv:2104.06967 https://arxiv.org/abs/2104.06967

[8] Ben Ltaifa Ibtihel, Hlaoua Lobna, and Ben Romdhane Lotfi. 2019. A Deep Learning-based Ranking Approach for Microblog Retrieval. *Procedia Computer Science* 159 (2019), 352–362. https://doi.org/10.1016/j.procs.2019.09.190 Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 23rd International Conference KES2019.

[9] Sai Muralidhar Jayanthi, Danish Pruthi, and Graham Neubig. 2020. NeuSpell: A Neural Spelling Correction Toolkit. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations.* Association for Computational Linguistics, Online, 158–164. https://doi.org/10.18653/v1/2020.emnlp-demos.21

[10] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 39–48. https://doi.org/10.1145/3397271.3401075

[11] Gregory R. Koch. 2015. Siamese Neural Networks for One-Shot Image Recognition.

[12] Hao Li, Yang Wang, Xinyu Liu, Zhichao Sheng, and Si Wei. 2018. Spelling Error Correction Using a Nested RNN Model and Pseudo Training Data. *CoRR* abs/1811.00238 (2018). arXiv:1811.00238 http://arxiv.org/abs/1811.00238

[13] Federico Nanni, Bhaskar Mitra, Matt Magnusson, and Laura Dietz. 2017. Benchmark for Complex Answer Retrieval. *CoRR* abs/1705.04803 (2017). arXiv:1705.04803 http://arxiv.org/abs/1705.04803

[14] Markus Näther. 2020. An In-Depth Comparison of 14 Spelling Correction Tools on a Common Benchmark. In *Proceedings of the 12th Language Resources and Evaluation Conference.* European Language Resources Association, Marseille, France, 1849–1857. https://aclanthology.org/2020.lrec-1.228

**Table 2: Impact of negative samples, computed with Model 1, MSE loss**

| Ratio | Train samples | Positive samples | Negative Sample | Recall@1 | Recall@5 | Recall@16 |
|---|---|---|---|---|---|---|
| 1:1 | 76K | 38K | 38K | 37.53% | 48.8% | 56.6% |
| 1:5 | 228K | 38K | 190K | 42.38% | 57.65% | 68.1% |
| 1:10 | 418K | 38K | 380K | 43% | 58.71% | 69.1% |

**Table 4: Experiments Result Summary**

| Configuration | Recall@1 | Recall@5 | Recall@16 |
|---|---|---|---|
| Prime Video | 0% | 0.1% | 34.13% |
| Pre-train Model 2 | 34.58% | 45.2% | 51.04% |
| Model 2 trained with 4M Samples & Contrastive loss | 60.68% | 77.47% | 85.38% |

[15] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *CoRR* abs/1611.09268 (2016). arXiv:1611.09268 http://arxiv.org/abs/1611.09268

[16] Priyank Nigam, Yiwei Song, Vijai Mohan, Vihan Lakshman, Weitian Ding, Ankit Shingavi, Choon Hui Teo, Hao Gu, and Bing Yin. 2019. Semantic Product Search. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2019). https://arxiv.org/pdf/1907.00937.pdf

[17] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).* Association for Computational Linguistics, Doha, Qatar, 1532–1543. https://doi.org/10.3115/v1/D14-1162

[18] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics. http://arxiv.org/abs/1908.10084

[19] Nils Reimers and Iryna Gurevych. 2020. Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation. *arXiv preprint arXiv:2004.09813* (04 2020). http://arxiv.org/abs/2004.09813

[20] Stephen Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. 1995. Okapi at TREC-3. In *Overview of the Third Text REtrieval Conference (TREC-3)* (overview of the third text retrieval conference (trec–3) ed.). Gaithersburg, MD: NIST, 109–126. https://www.microsoft.com/en-us/research/publication/okapi-at-trec-3/

[21] Keisuke Sakaguchi, Kevin Duh, Matt Post, and Benjamin Van Durme. 2016. Robsut Wrod Reocginiton via semi-Character Recurrent Neural Network. *CoRR* abs/1608.02214 (2016). arXiv:1608.02214 http://arxiv.org/abs/1608.02214

[22] Gerard Salton and Michael J. McGill. 1986. *Introduction to Modern Information Retrieval.* McGraw-Hill, Inc., USA.

[23] Ellen Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2021. TREC-COVID: Constructing a Pandemic Information Retrieval Test Collection. *SIGIR Forum* 54, 1, Article 1 (2021), 12 pages. https://doi.org/10.1145/3451964.3451965

[24] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP.* Association for Computational Linguistics, Brussels, Belgium, 353–355. https://doi.org/10.18653/v1/W18-5446

[25] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. *CoRR* abs/2002.10957 (2020). arXiv:2002.10957 https://arxiv.org/abs/2002.10957

[26] Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. 2022. Laprador: Unsupervised pretrained dense retriever for zero-shot text retrieval. *arXiv preprint arXiv:2203.06169* (2022).

## Letter transposition map

This mapping shows possible replacements for each letter. This is based on the common keyword errors, missing character and transliteration variations.

Not all the keys in the adjacent have equal probability to be typed by mistake. To get a rough estimation for keyword based error, we referred to this article [1] which shows number of possible dictionary words possible by replacing each letter by its adjacent keyword keys. These represents a rough proxy as humans are most likely to type a correct dictionary word then a total random word. For missing space and transliteration variation error related replacements, we take the average value of keywords based error count. For numbers, we considered adjacent keys and put same score for both.

## Transliteration variation

We have considered these transliteration variations for our experiments.

**Table 2: Transliteration variations**

| Letter | Replacement |
|---|---|
| a | aa |
| e | i |
| i | ee |
| l | ll |
| s | z |
| u | oo |
| w | wh |
| z | s |

## Dataset samples summary

Table 3 shows number of samples from each type of transformation. We found total 63028 samples out of which 40531 are distinct.

**Table 1: letter mapping**

| Letter | Possible replacements | Replacement counts |
|---|---|---|
| ’a’ | [’q’,’s’,’w’,’x’,’z’,”,’aa’] | [594,42401,10853,3822,3062,12146,12146] |
| ’b’ | [’f’,’g’,’h’,’n’,’v’,”] | [16112,21182,10826,19375,6146,14728] |
| ’c’ | [’d’,’f’,’s’,’v’,’x’,”] | [19151,15124,37974,7444,1854,16309] |
| ’d’ | [’c’,’e’,’f’,’r’,’s’,’v’,’w’,’x’,”] | [19151,39499,16091,64063,80813,7848,10614,2018,30012] |
| ’e’ | [’d’,’f’,’r’,’s’,’w’,”,’i’] | [39499,17080,76503,75665,13193,44388,44388] |
| ’f’ | [’b’,’c’,’d’,’e’,’g’,’r’,’t’,’v’,”] | [16112,15124,16091,17080,13344,18722,20980,5822,15409] |
| ’g’ | [’b’,’f’,’h’,’n’,’r’,’t’,’v’,’y’,”] | [21182,13344,10144,23414,22092,30296,5093,5295,16357] |
| ’h’ | [’b’,’g’,’j’,’m’,’n’,’t’,’u’,’y’,”] | [10826,10144,2663,11486,11859,23856,10462,5518,10851] |
| ’i’ | [’j’,’k’,’l’,’o’,’u’,”,’e’,’ee’] | [699,9983,40985,82987,63669, 39644, 39644,39644] |
| ’j’ | [’h’,’i’,’k’,’m’,’n’,’u’,’y’,”] | [2663,699,1248,3464,2011,568,672,1618] |
| ’k’ | [’i’,’j’,’l’,’m’,’o’,’u’,”] | [9983,1248,14651,8496,8366,5455,8033] |
| ’l’ | [’i’,’k’,’o’,’p’,”,’ll’] | [40985,14651,43713,30126,32368, 32368] |
| ’m’ | [’h’,’j’,’k’,’n’,”] | [11486,3464,8496,23433,11719] |
| ’n’ | [’b’,’g’,’h’,’j’,’m’,”] | [19375,23414,11859,2011,23433,16018] |
| ’o’ | [’i’,’k’,’l’,’p’,”] | [82987,8366,43713,18072,38248] |
| ’p’ | [’l’,’o’,”] | [30126,18072,24099] |
| ’q’ | [’a’,’s’,’w’,”] | [594,2041,728,1121] |
| ’r’ | [’d’,’e’,’f’,’g’,’t’,”] | [64063,76503,18722,22092,54571,47190] |
| ’s’ | [’a’,’c’,’d’,’e’,’w’,’x’,’z’,”,’z’] | [42401,37974,80813,75665,17079,3613,7300,37835,37835] |
| ’t’ | [’f’,’g’,’h’,’r’,’y’,”] | [20980,30296,23856,54571,13286,28598] |
| ’u’ | [’h’,’i’,’j’,’k’,’y’,”,’oo’] | [10462,63669,568,5455,6783,17387,17387] |
| ’v’ | [’b’,’c’,’d’,’f’,’g’,’y’,”] | [6146,7444,7848,5822,5093,6783,6523] |
| ’w’ | [’a’,’d’,’e’,’q’,’s’,”,’wh’] | [10853,10614,13193,728,17079,10493,10493] |
| ’x’ | [’a’,’c’,’d’,’s’,’z’,”] | [3882,1854,2018,3613,516,2377] |
| ’y’ | [’g’,’h’,’j’,’t’,’u’,”] | [5295,5518,672,13286,6783,6311] |
| ’z’ | [’a’,’s’,’x’,”,’s’] | [3062,7300,516,3626,3626] |
| ’ ’ | [”] | [1] |
| ’-’ | [”] | [1] |
| ’&’ | [”] | [1] |
| ’:’ | [”] | [1] |
| ’1’ | [’“’,’2’,”] | [2,2,2] |
| ’2’ | [’1’,’3’,”] | [2,2,2] |
| ’3’ | [’2’,’4’,”] | [2,2,2] |
| ’4’ | [’3’,’5’,”] | [2,2,2] |
| ’5’ | [’4’,’6’,”] | [2,2,2] |
| ’6’ | [’5’,’7’,”] | [2,2,2] |
| ’7’ | [’6’,’8’,”] | [2,2,2] |
| ’8’ | [’7’,’9’,”] | [2,2,2] |
| ’9’ | [’8’,’0’,”] | [2,2,2] |
| ’0’ | [’_’,’9’,”] | [2,2,2] |

**Table 3: Dataset samples summary**

| Transformation type | Number of samples |
|---|---|
| Keyword error | 3080 |
| Missing character | 19775 |
| Transliteration | 4216 |
| Transpose adjacent letters | 2729 |
| Missing space | 6167 |
| Numbers | 12 |
| Suffix | 55 |
| Total | 63028 |
| Total Distinct | 40531 |