A Best Match KNN-based Approach for Large-scale Product Categorization

Haohao Hu
School of Information Technology
York University
Toronto, ON, Canada
haohaohu@yorku.ca

Runjie Zhu
Department of Electrical Engineering and Computer Science, York University
Toronto, ON, Canada
sherryzh@yorku.ca

Yuqi Wang
Department of Mathematics, Trent University
Peterborough, ON, Canada
yuqiwang@trentu.ca

Wenying Feng
Department of Mathematics, Trent University
Peterborough, ON, Canada
wfeng@trentu.ca

Xing Tan
School of Information Technology
York University
Toronto, ON, Canada
xtan@yorku.ca

Jimmy Xiangji Huang
School of Information Technology
York University
Toronto, ON, Canada
jhuang@yorku.ca

ABSTRACT

We use K Nearest Neighbors (KNN) classic classification model and the Best Match (BM)25 probabilistic information retrieval model to assess how efficiently the classic KNN model could be modified to solve the real-life product categorizing problem. This paper gives a system description of the KNN-based algorithm for solving the product classification problem. Our submissions experimented are based on the Rakuten 1M product listings datasets in tsv format provided by the Rakuten Institute of Technology Boston. The classification of our KNN algorithm was based on the product title similarity scores generated from the BM25 Information Retrieval Model. With the setting of k=3 in KNN, our proposed program achieved 0.7809, 0.7821, 0.7790 in weighted-{precision, recall and F1 score} respectively in the test dataset.

CCS CONCEPTS

• **Information systems → Probabilistic retrieval models**; **Clustering and classification**; • **Computing methodologies → Instance-based learning**; • **Applied computing → Enterprise ontologies, taxonomies and vocabularies**;

KEYWORDS

SIGIReCom' 18; Large-scale taxonomy classification

ACM Reference Format:
Haohao Hu, Runjie Zhu, Yuqi Wang, Wenying Feng, Xing Tan, and Jimmy Xiangji Huang. 2018. A Best Match KNN-based Approach for Large-scale Product Categorization. In *Proceedings of ACM SIGIR Workshop on eCommerce (SIGIR 2018 eCom Data Challenge).* ACM, New York, NY, USA, Article 4, 6 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

As the fast-paced development of the internet, there has been a huge rise of the e-commerce market. Online shopping platforms such as Amazon and Alibaba provide not only goods meeting consumers' specific needs, but also products that are basically everyone's daily consumption in life. Almost all the e-commerce platforms aim for updating their shopping lists and inventories at their fastest speed to target certain consumers in order to win a bigger proportion of the market. Therefore, the technologies adopted to efficiently and effectively recognizing product categories become more important. This would, on one hand, help the system operators to add in or delete certain items from consumers' shopping list. On the other hand, it would also be easier for system operators or managers to deal with data analysis and data management in future. This specific data challenge belongs to the large-scale taxonomy classification domain and focuses on the fundamental problem of predicting product's category in the taxonomy tree with given product's title.

2 RELATED WORK AND MOTIVATION

Properly categorizing a new product as a dynamically updated category in the form of a taxonomy tree is of critical importance for e-commerce. Algorithms in support automated process for categorizing need to be straightforwardly simple for its scalability, flexible to allow labeling errors and noises, distributive over tree branches and paths hence the taxonomy trees they create are largely balanced. Leading approaches for measuring path similarities in a taxonomy tree make use of Wordnet [12] and address the problem in terms of product taxonomy alignment [1, 13]. A most recent effort turns to graphical models enriched with semantics, using frameworks such as Markov Logic Networks [14], or Probabilistic Soft Logic [2]. The taxonomy can actually be flatted for the purpose of categorizing. For example, in [17], a two-level classification, first on discovering latent groups through clustering the target classes, on training to classify items into those groups. The approach calls for additional parameter tuning.

Nearest-neighbor for classification can be traced back to as early as 1950s [4, 9]. We chose KNN for this task, because: 1) KNN has been used in text classification, which is similar to this taxonomy

classification task. Although it is simple, it was shown to perform as well as SVM in text classification [9]. 2) according to our analysis, the training dataset contains 3008 distinct category id paths, which is computationally expensive in general and particularly for more complicated algorithms such SVM.

Cosine similarity (or Vector Space Model (VSM)) is often used to measure similarity between two text documents[9]. We chose BM25 model, as it is often considered better than VSM. Since this current research uses BM25 to measure the similarity between two specific product titles, we give some brief review on BM25 and its predecessor Okapi here [5]. The leader of our team Prof. Huang was instrumental in the research reported in [5], and has continued to work and contribute consistently on the subject for two decades to follow, in theory and in application. More specifically, as recorded in [8], an enhanced version BM25 and Okapi system win Huang and his team the first place in the Genomics/biomedical track among all 135 entrants from around the world in international TREC competitions organized by National Institute of Standards & Technology. Term proximity for enhancement of BM25 were proposed in [6], with solidly verified improvement on effectiveness. Pseudo term (a.k.a., Cross Term) to model term proximity for boosting retrieval performance and thus the bigram CRoss TErm Retrieval (CRTER) retrieval model for searching were proposed in [20]. Meanwhile, an integrated sampling technique incorporating both over-sampling and under-sampling, with an ensemble of SVMs to improve the prediction performance is considered in [10]; A novel machine-learning-based data classification algorithm applied to network intrusion detection is reported in [3]; In [7], data mining to Pseudo-Relevance Feedback for High Performance Text Retrieval is investigated.

3 METHODOLOGY

In this section we first give brief introduction to technical preliminaries, BM25 the ranking function in particular. Categorization through classification in terms of KNN+BM25 is explained next (pseudo code in the upper part of Table 1), with an example provided. The framework in support of the classifier is also presented (illustration in Figure 1 and pseudo code in the lower part of Table 1).

3.1 Preliminaries

We chose the K nearest neighbors (KNN), which is a classic classification algorithm, as our major classification approach. KNN is traditionally a simple algorithm that stores all the available candidates for classification, and it classifies each new candidate based on the similarity measure.

Definition 1: KNN classification:

K-nearest neighbors algorithm is structured on the basis of feature similarity measurement. In other words, the degree of how closely the sparse sample features resemble the training dataset determines how we classify a given data point.

The most intuitive K-nearest neighbor classifier is to set the $k = 1$, or the one nearest neighbor classifier which assigns point a to the class of its closest neighbor in the feature space,

$$C_n^{1nn}(a) = Y(1) \tag{1}$$

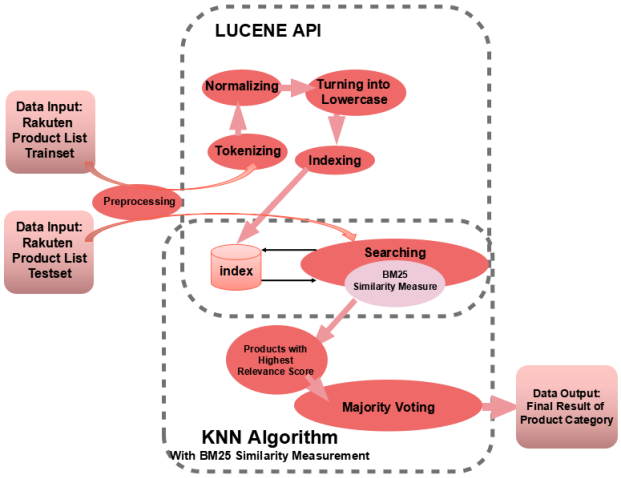

Figure 1: A KNN+BM25 Classifier System

Thus, k-nearest neighbor classifier could be considered as a generalized weighted nearest neighbor classifier where the assignment of k nearest neighbors is a weight of $\frac{1}{k}$ and all others weigh zero. Specifically, $\sum_{i=1}^{n} w_{ni} = 1$ represents the ith nearest neighbor is assigned with a weight of w_{ni}. Therefore, with the weighted score of the nearest neighbor classifier, the class of its closest neighbor in a feature space will denote as C_n^{wnn} with weights $\{w_{ni}\}_{i=1}^{n}$.

Definition 2: BM25:

BM25 (Best Match) [5, 15, 16] is a probabilistic ranking function which ranks the matching documents based on their degree of relevance to the given user queries.

To get a document D's BM25 score given a query Q, a weighting function for each query term $q_i \in Q$ and the document D is first calculated as follows:

$$w(q_i, D) = \frac{(k_1 + 1) \times TF(q_i, D)}{K + TF(q_i, D)} \times \frac{(k_3 + 1) \times QTF(q_i)}{k_3 + QTF(q_i)} \times IDF(q_i) \tag{2}$$

where $K = k_1 \times [(1 - b) + b \times dl/avdl]$, dl is the length of D, $avdl$ is the average document length. k_1, k_3, b are parameters. $IDF(q_i) = \log(1 + \frac{N - DF(q_i) + 0.5}{DF(q_i) + 0.5})$. N is the number of indexed documents in the collection. $DF(q_i)$ is the number of documents containing q_i. $TF(q_i, D)$ is the number of occurrence of q_i in D, and $QTF(q_i)$ is the number of occurrence of q_i in Q. A document D's BM25 similarity score given a query Q is calculated as the sum of D's weight for each Q's term:

$$BM25(Q, D) = \sum_{i=1}^{|Q|} w(q_i, D), \tag{3}$$

where w is the term weight obtained from the above Equation (2), $|Q|$ is the number of terms in Q.

3.2 A KNN+BM25 Classifier for Categorization

In our program, k of our KNN classifier is set as a parameter with respect to a given query. The output of searcher using BM25 model will return at most k top matches. And these top matched products'

categoy id paths are the input of our KNN classifier. In other words, an input of our KNN algorithm consists of category id paths of k closest training titles given a test title. And the output of our KNN algorithm is the majority category id path among category id paths of those training titles, i.e. the category with highest occurrence. We wanted to examine whether it was effective to use a flat classification structure to solve the given problem instead of a hierarchical one. Thus, all items in the product list are classified in one shot.

A BM25 similarity score is calculated for each title in the training set and a title in test set. In KNN paradigm, the similarity function of our approach is the BM25 similarity score between an item title in test set and an item title in training set. The higher the BM25 score, the more similar a training title and a test title. We tried setting different values of k in KNN to see whether or not predicting based on individual match is better than on multiple matches, since the individual match may be an outlier. Our KNN+BM25 algorithm is shown in the upper part of Table 1.

Suppose p_i is a product in the training dataset TR. p_i contains product title pt_i and product category id path c_i. t_j is a product title in the test dataset TE. N is the number of products in TR. When $k = 1$, we assign the category id path c_1 of the top 1 matched training title (document), i.e. the title pt_1 with highest BM25 similarity score given that test title (query) t, as the predicted category id path pc:

$$pc = c_1$$

where $BM25(t, pt_1) = \max_{m \in \{1,2,\ldots,N\}} BM25(t, pt_m)$

Generally, when $k > 1$, we assign the majority category id path of returned top n ($n \leq k$, since it is possible that the number of matches is less than k) products' category id paths $\{c_1, c_2, ..., c_n\}$. Specifically, the algorithm finds the distinct category id paths $\{dc_1, dc_2, ..., dc_i\} \subset \{c_1, c_2, ..., c_n\} (i \leq n)$ and their number of occurrences $\{Occur(dc_1), Occur(dc_2), ..., Occur(dc_i)\}$ $(\sum_{m=1}^{i} Occur(dc_m) = n)$. The distinct category id path dc_j with the highest number of occurrences among the category id paths of the top k matched training titles given a test title is deemed the predicted category id path pc:

$$pc = dc_j \quad where \quad Occur(dc_j) = \max_{q \in \{1,2,\ldots,i\}} Occur(dc_q) \tag{4}$$

If the category id paths have same number of occurrence within the top matched training titles, we assign the category id path of the higher ranked matched training title(s) as predicted category id path. For example, if

$$Occur(dc_1) = Occur(dc_2) = \max_{q \in \{1,2,\ldots,i\}} Occur(dc_q) \tag{5}$$

, then dc_1 is the predicted category id path. If no match is found ($n = 0$), we assign "2296>3597>689" as predicted category id path, which corresponds to "Media>Music>Pop" (manually judging from training data and Rakuten website[1]).

Here is an example to show a typical product listing in our dataset. We set $k_1 = 1.2$, $b = 0.92$ in BM25. As shown in Fig.2, given this item (query) in test dataset:

```
"Sterling Silver Dangle Ball Earrings w/ Brilliant Cut
 CZ Stones & Yellow Topaz-colored Crystal Balls, 1""
(26 mm) tall"
```

[1]https://www.rakuten.com

function KNN_BM25(q, DC, k) **returns** predicted category id path
  inputs: q: the query (test title)
    DC: the document collection of product titles and corresponding category id paths
    k: the k value in KNN algorithm
  local variables: p_j: the jth matched training product containing pt_j (product title) and c_j (its corresponding category id path)
    pc: the predicted category id path

  search q in DC with BM25 IR model
    get top n ($n \leq k$) matches $\{pt_1, pt_2, ..., pt_n\} \subset DC$ and corresponding $\{c_1, c_2, ..., c_n\}$
  if n equals 0 **then**
    set pc as "2296>3597>689"
  else
    set pc as the majority category id path of $\{c_1, c_2, ..., c_n\}$
  end if
  return pc

procedure Main_program(TR, TE, k) **returns** prediction file
  inputs: TR: the training dataset
    TE: the test dataset
    k: the k value for KNN algorithm
  local variables: p_j: the jth training product containing pt_j (product title) and c_j (its corresponding category id path)
    t_j: the jth test product title
    pc_j: the predicted category id path of t_j

for each $p_j \in TR$ **do**
  preprocess pt_j
  tokenize pt_j
  normalize pt_j
  lowercase pt_j
  index pt_j
  store (pt_j, c_j) in $DocumentCollection$
end for
for each $t_j \in TE$ **do**
  preprocess t_j and store it in $temp\_t$
  get pc_j through KNN_BM25($temp\_t, DocumentCollection, k$)
  write t_j and pc_j in prediction file
end for

Table 1: pseudo code of the classifier in action

If we set $k = 10$, then the searcher will return the item's top 10 matches in training set according to BM25 similarity score of a document and the query in descending order as shown in Table 2 below.

As an illustration, the term weight for the matched term "sterling" (q_1) in the top 1 document (D_1) is calculated as follows:

Ranking	Product Title	Category Id Path	BM25 score
1	"Sterling Silver Marquise Shape Dangle Earrings with Brilliant Cut CZ Stones, 1 1/16 in. (26 mm) tall"	1608>2320>2173>2878	56.89168
2	"Sterling Silver Floral Dangle Chandelier Earrings with Brilliant Cut CZ Stones, 1 1/4 in. (31 mm) tall"	1608>2320>2173>2878	51.909283
3	"Sterling Silver Curvy Hoop Earrings with Brilliant Cut CZ Stones, 13/16 in. (21 mm)"	1608>2320>2173>2878	42.515083
4	"Sterling Silver French Clip Black Onyx Bar Earrings with Brilliant Cut CZ Stones, 5/8 in. (16 mm) tall"	1608>2320>2173>2878	41.787933
5	"Sterling Silver Square-shaped Stud Earrings (7 mm) & Pendant (12mm tall) Set, with Princess Cut Blue Sapphire-colored CZ Stones"	1608>2320>2173>3881	40.36754
6	"Sterling Silver Double Wire Knot Lace Post Earrings with Brilliant Cut CZ Stones, 1 7/16 in. (36 mm)"	1608>2320>2173>2878	39.01934
7	"High Polished Sterling Silver 3/4"" (18 mm) tall Heart Cut Out Pendant, with Brilliant Cut CZ Stones, with 18"" Thin Box Chain"	1608>2320>498>1546	37.43821
8	"High Polished Sterling Silver 7/16"" (11 mm) tall Bead Charm, with Brilliant Cut CZ Stones, with 18"" Thin Box Chain"	1608>2320>2495>3682	37.43036
9	"Sterling Silver Black Onyx Ring with Brilliant Cut CZ Stones, 1/4 in. (6 mm) wide, size 7"	1608>2320>3648	36.71535
10	"High Polished Sterling Silver 11/16"" (17 mm) tall Flower Cut Out Pendant, 1.5mm Brilliant Cut CZ Stones, with 18"" Thin Box Chain"	1608>2320>498>1546	36.443573

Table 2: Top 10 matches obtained given the query

Figure 2: An earring item need to be categorized. Picture source: https://www.rakuten.com/shop/sabrina-silver/product/TE3873/

$$
\begin{aligned}
w(q_1, D_1) &= \frac{(k_1+1)\times TF(q_1,D_1)}{k_1\times[(1-b)+b\times dl/avdl]+TF(q_1,D_1)}\times \\
&\quad \frac{(k_3+1)\times QTF(q_1)}{k_3+QTF(q_1)}\times\log\left(1+\frac{N-DF(q_1)+0.5}{DF(q_1)+0.5}\right) \\
&= \frac{(1.2+1)\times 1}{1.2\times[(1-0.92)+0.92\times 18/11.566492]+1}\times \\
&\quad \frac{(8+1)\times 1}{8+1}\times\log\left(1+\frac{800000-16528+0.5}{16528+0.5}\right) \\
&= 3.0329628
\end{aligned}
\tag{6}
$$

The KNN algorithm (majority voting) then counts the occurrences of the categories within these matches. As shown in Table 3 below, "1608>2320>2173>2878" (corresponding to "Clothing, Shoes & Accessories>Jewelry & Watches>Earrings>Stud Earrings") has the highest number of occurrences among the top 1/3/5/7/10 matches' categories. Thus, the category '1608>2320>2173>2878' is assigned as the predicted category id path when our KNN algorithm's k is set to 1/3/5/7/10. We have mannually verified on the Rakuten website[2] that this prediction is correct.

3.3 The Classifier in Action

Based on the classifier as above, we actually implement a system for the classifier in action. Figure 1 is a pictorial description of the system, where ovals are functional components, cylinder is index and rounded rectangles are data inputs/outputs in interaction with the classifier system. Specifically, input product can be effectively categorized through searching the index of product titles in training dataset. Training data are preprocessed and tokenized to get to a word-based index pool. Given a query, the system calculates the BM25 relevance score of the given product title and training titles, and categorizes it into the category of its most relevant product title(s) in training set. The implementation is in JAVA. We explored different approaches and strategies for minimizing the classification error and matching the product categories with high accuracy.

More precisely, as shown in Figure 1 and lower part of Table 1, major flow of the product categorizing system is as follows:

First, the Rakuten Training Dataset TR (800,000 product titles with category id paths) in tsv format is read line by line as document inputs. Each document $p_j \in TR$ has two fields, one for product title

[2]https://www.rakuten.com

Candidate Category Id Path	Candidate Category	k values				
		1	3	5	7	10
1608>2320>2173>2878	Clothing, Shoes & Accessories>Jewelry & Watches>Earrings>Stud Earrings	1	3	4	5	5
1608>2320>498>1546	Clothing, Shoes & Accessories>Jewelry & Watches>Pendants & Necklaces>Pendants	0	0	0	1	2
1608>2320>2173>3881	Clothing, Shoes & Accessories>Jewelry & Watches>Earrings>Earring Sets	0	0	1	1	1
1608>2320>3648	Clothing, Shoes & Accessories>Jewelry & Watches>Rings	0	0	0	0	1
1608>2320>2495>3682	Clothing, Shoes & Accessories>Jewelry & Watches>Accessories>Individual Charms	0	0	0	0	1

Table 3: Distinct Category id paths, corresponding categories and numbers of occurrences within top k(k=1/3/5/7/10) documents' category id paths

pt_j and one for category id path c_j. Second, documents' product titles pt_j are preprocessed. Specifically, "w/out" was replaced by "without", "w/" was replaced by "with", "&" was replaced by "and", "'" was replaced by "feet" and """ was replaced by "inches". After that, documents' product titles pt_j and category id paths c_j are indexed in text field and string field respectively. Product titles pt_j in text fields are analyzed by Lucene Standard Analyzer. Specifically, they are tokenized by Lucene's standard tokenizer, before being normalized by Lucene's standard filter and turned into lowercase by lowercase filter. In contrast, category id paths c_j in string fields are not analyzed, since they are target categories for later classification. After that, Test Dataset TE (200,000 product titles without category id paths) is read line by line as query inputs. Same as the training data, test titles $t_j \in TE$ are preprocessed in the same way. Then, the searcher searches the index with test title t_j with BM25 similarity to get top n ($n \leq k$) most relevant training titles $\{pt_1, ..., pt_n\}$ and their corresponding category id paths $\{c_1, ..., c_n\}$. This is followed by our KNN algorithm (as shown in upper part of Table 1) that returns the most frequent category id path among top n matched training title(s)' category id paths $\{c_1, c_2, ..., c_n\}$ given a test title t_j as predicted category id path pc_j. Finally, test titles and their predicted category id paths are written in a tsv file.

4 EXPERIMENTAL ANALYSIS

In this section, we test our system on Rakuten data. Experimental set-ups are introduced first, results obtained are analyzed. We specifically tested on different k values in KNN and different values of k_1 and b in BM25 for their impacts on effectiveness of the classifier and the system.

4.1 Experimental Set-ups

Experiments were mainly done on a laptop with 4GB RAM. We trained the K nearest neighbors algorithm using the Rakuten 800,000 product listings in tsv format provided. And we used the Java program to exercise the experiments. Lucene is an open-source information retrieval (IR) software library which is empowered to do full text indexing and full text searching capability. This architecture is built on the idea of a document with fields of text. We exercise our experiments on top of the Lucene API in order to get a full product list search for the most accurate result of product categorization.

We tried setting different values of k in KNN to see whether or not predicting based on individual match ($k = 1$) is better than on several matches ($k > 1$), since the individual match may be an outlier. In particular, k was set to 1, 3, 5, 7, 50 and 100.

Model	Weighted-{precision	recall	F1 score}
kNN ($k = 1$)	0.78	0.78	0.78
kNN ($k = 3$)	0.79	0.78	0.78
kNN ($k = 5$)	0.78	0.78	0.78
kNN ($k = 7$)	0.78	0.78	0.77
kNN ($k = 50$)	0.71	0.73	0.71
kNN ($k = 100$)	0.67	0.70	0.67

Table 4: Performance on a subset of the test set.

4.2 Results and Analysis

The results in Table 4 above show the official results of our primary submissions. In this classification problem, the experimental results are evaluated with weighted-{precision (P), recall (R) and F1 score} respectively. Precision is an evaluation of the fraction of relevant items among all retrieved times; Recall is an evaluation of the fraction of relevant items that have been retrieved over the total amount of the relevant items. And the F1 Score is a measure of the accuracy of the test. In our experiment, with the setting of parameter $k = 3$ in KNN classification algorithm, our program achieved 0.79, 0.78 and 0.78 for the weighted-{P, R and F1 score} respectively in a subset of the test dataset. The results of k=1, 3 and 5 are roughly the same, because the top document matches of a query are highly similar to each other and thus have high probability of belonging to the same category. Also, the results for $k = 3$ rather than $k = 1$ is the best one among different settings of k, because the top documents have high probability to have the same BM25 similarity score and thus the top 1 document's category may be an outlier. Generally, we can see that the prediction result declines as k increases, since titles with lower similarity are less likely to belong to the same category, as shown in the Example in Section 3.2.

Aside from tuning k in KNN, we have tried to tune the parameters of the BM25 IR model to get higher classification accuracy. We found a slight difference in between the tuning of the parameters. Specifically, with the same setting of $k = 1$ in KNN algorithm, by setting $k_1 = 1.2$, b=0.35, we achieved slightly lower results of (0.78,

0.77, 0.77) for weighted-{P, R and F1 score} respectively than those of the default parameters ($k_1 = 1.2$, $b = 0.75$) which get (0.78, 0.78, 0.78) for weighted-{P, R and F1 score} respectively. Also, we splited the training dataset into 2 parts, 1 as training set (1-in-2-TRAIN) and 1 as testing set (2-in-2-TEST)(category id paths are removed) and gold standard (2-in-2-GOLD). Then, we conducted parameter tuning by fixing k for KNN to 3, k_1 to 1.2 and changing the value of b. We found the optimal weighted-F1 score is obtained when $b = 0.92$ or 0.93.

We have also compared results of using the stopword filter in Lucene's standard analyzer to those of not using it. We found that using stopword filter would slightly reduce the results, since after stop word removal, documents (training titles) may have no terms.

Apart from the BM25 model, we also used Lucene's VSM (default setting, parameter-free) to conduct searching on the product list. When k is set to 1 in KNN, the results (0.77, 0.77, 0.77) are slightly lower than those of BM25 IR model on a subset of the test dataset. This is because compared to VSM that only incorporates term frequency (TF) and inverse document frequency (IDF), BM25 also takes the average document length (*avdl*) into account , which leads to higher accuracy rate as results. We also tried Lucene's implementation of Dirichlet Language Model [18] and Jelinek-Mercer Language Model[18] and tuned the parameter μ and λ respectively. We found by using the same training set (1-in-2-TRAIN), test set (2-in-2-TEST) and $k = 3$ for KNN, the results of the Dirichlet Language Model (μ is set to 0.1)(0.7492, 0.7563, 0.7499) and Jelinek-Mercer Language model ($\lambda = 0.25$)(0.7494, 0.7569, 0.7501) are slightly lower than those of BM25 (0.7539, 0.7573, 0.7534)($k_1 = 1.2$, $b = 0.92$)(all models' parameters are tuned to optimize weighted-F1 score) in terms of weighted-{P, R and F1 score} respectively.

5 BRIEF SUMMARY

In this paper, we described our taxonomy classification system based on KNN with Lucene BM25 similarity score.

The insights from participating this competition are as follows:

IR model can be used as similarity function in KNN to generate competitive prediction results.

For future work, we are going to incorporate word associations into our analysis by adding algorithms like bigram or Word2Vec skip-gram [11] into the Lucene search system to get more accurate match. In particular, CRTER (CRoss TERm) [20] can be combined with BM25 model. We also found that the most important part-of-speech (POS) for predicting product category is noun. Thus, it would be helpful to incorporate POS tagger into the search engine to give more weights to nouns. It is also interesting to use other more powerful IR models, such as Context-sensitive Proximity Model [19], as similarity function in KNN.

ACKNOWLEDGMENTS

We gratefully acknowledge the support by NSERC (Natural Sciences and Engineering Research Council of Canada) CREATE (Collaborative Research and Training Experience Program) award in ADERSIM (Advanced Disaster, Emergency and Rapid-response Simulation)[3], ORF-RE (Ontario Research Fund - Research excellence)[4] award in BRAIN (Big Data Research, Analytics, and Information Network) Alliance[5], and the Research Chair Program at York University, Canada.

REFERENCES

[1] Steven S. Aanen, Damir Vandic, and Flavius Frasincar. 2015. Automated Product Taxonomy Mapping in an E-commerce Environment. *Expert Systems with Applications* 42, 3 (2015), 1298 – 1313. https://doi.org/10.1016/j.eswa.2014.09.032

[2] Varun Embar, Golnoosh Farnadi, Jay Pujara, and Lise Getoor. 2018. Aligning Product Categories Using Anchor Products. In *First Workshop on Knowledge Base Construction, Reasoning and Mining*.

[3] Wenying Feng, Qinglei Zhang, Gongzhu Hu, and Jimmy Xiangji Huang. 2014. Mining Network Data for Intrusion Detection through Combining SVMs with Ant Colony Networks. *Future Generation Comp. Syst.* 37 (2014), 127–140. https://doi.org/10.1016/j.future.2013.06.027

[4] Jiawei Han, Micheline Kamber, and Jian Pei. 2011. *Data Mining: Concepts and Techniques* (3rd ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[5] Micheline Hancock-Beaulieu, Mike Gatford, Xiangji Huang, Stephen E. Robertson, Steve Walker, and P. W. Williams. 1996. Okapi at TREC-5. In *Proceedings of The Fifth Text REtrieval Conference, TREC 1996, Gaithersburg, Maryland, USA, November 20-22, 1996*. http://trec.nist.gov/pubs/trec5/papers/city.procpaper.ps.gz

[6] Ben He, Jimmy Xiangji Huang, and Xiaofeng Zhou. 2011. Modeling Term Proximity for Probabilistic Information Retrieval Models. *Inf. Sci.* 181, 14 (2011), 3017–3031. https://doi.org/10.1016/j.ins.2011.03.007

[7] Xiangji Huang, Yan Rui Huang, Miao Wen, Aijun An, Yang Liu, and Josiah Poon. 2006. Applying Data Mining to Pseudo-Relevance Feedback for High Performance Text Retrieval. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China*. 295–306. https://doi.org/10.1109/ICDM.2006.22

[8] Xiangji Huang, Ming Zhong, and Luo Si. 2005. York University at TREC 2005: Genomics Track. In *Proceedings of the Fourteenth Text REtrieval Conference, TREC 2005, Gaithersburg, Maryland, USA, November 15-18, 2005*. http://trec.nist.gov/pubs/trec14/papers/yorku-huang2.geo.pdf

[9] Bing Liu. 2007. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Springer Science & Business Media.

[10] Yang Liu, Xiaohui Yu, Jimmy Xiangji Huang, and Aijun An. 2011. Combining Integrated Sampling with SVM Ensembles for Learning from Imbalanced Datasets. *Inf. Process. Manage.* 47, 4 (2011), 617–631. https://doi.org/10.1016/j.ipm.2010.11.007

[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Advances in Neural Information Processing Systems*. 3111–3119.

[12] George A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM* 38, 11 (Nov. 1995), 39–41. https://doi.org/10.1145/219717.219748

[13] Sangun Park and Wooju Kim. 2007. Ontology Mapping between Heterogeneous Product Taxonomies in an Electronic Commerce Environment. *International Journal of Electronic Commerce* 12, 2 (2007), 69–87. http://www.jstor.org/stable/27751250

[14] Matthew Richardson and Pedro Domingos. 2006. Markov Logic Networks. *Mach. Learn.* 62, 1-2 (Feb. 2006), 107–136. https://doi.org/10.1007/s10994-006-5833-1

[15] Stephen E Robertson. 1997. Overview of the Okapi Projects. *Journal of Documentation* 53, 1 (1997), 3–7.

[16] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at TREC-3. *NIST Special Publication Sp* 109 (1995), 109.

[17] Dan Shen, Jean-David Ruvini, and Badrul Sarwar. 2012. Large-scale Item Categorization for E-Commerce. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM '12)*. ACM, New York, NY, USA, 595–604. https://doi.org/10.1145/2396761.2396838

[18] Chengxiang Zhai and John Lafferty. 2017. A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. In *ACM SIGIR Forum*, Vol. 51. ACM, 268–276.

[19] Jiashu Zhao and Jimmy Xiangji Huang. 2014. An Enhanced Context-sensitive Proximity Model for Probabilistic Information Retrieval. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 1131–1134.

[20] Jiashu Zhao, Jimmy Xiangji Huang, and Ben He. 2011. CRTER: Using Cross Terms to Enhance Probabilistic Information Retrieval. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*. 155–164. https://doi.org/10.1145/2009916.2009941

[3] http://www.yorku.ca/adersim/NSERC/

[4] https://www.ontario.ca/page/ontario-research-fund-research-excellence

[5] http://www.brainalliance.ca/