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 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 KNN algorithm was calculated based on the item relevance scores generated from the BM25 Information Retrieval Model. With the setting of k=1 or 3 in KNN, our proposed program achieved 0.78, 0.78, 0.78 in weighted-precision, recall and F1 score respectively in a subset of 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 Huang. 2018. A Best Match KNN-based Approach for Large-scale Product Categorization. In *Proceedings of ACM SIGIR Workshop On eCommerce (SIGIReCom'18)*, Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, Article 4, 4 pages. https://doi.org/10.475/123_4

SIGIReCom'18, July 2018, Ann Arbor, Michigan USA

ACM ISBN 123-4567-24-567/08/06…$15.00
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 remains in the large-scale taxonomy classification domain and focuses on the fundamental problem of predicting product's category with given product's title in the taxonomy tree.

2 RELATED WORK AND MOTIVATION

Properly cataloging new products into a dynamically updated category in the form of a taxonomy tree is of critical impermanence for e-commerce. Algorithms in support automated process for cataloging 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 for example Markov Logic Networks [15], or Probabilistic Soft Logic [2]. The taxonomy can actually be flatted for the purpose of cataloging. For example, in [18], 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]. Since this current research uses BM25 to measure the distance between two specific neighbor category, 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 [19]. 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, with an example provided. The framework in support of the classifier is also presented (illustration in Figure 1).

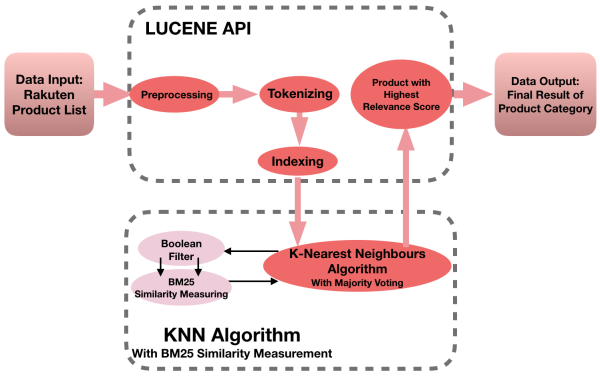

Figure 1: A KNN+BM25 Classifier System

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. According to our analysis, the training dataset contains 3008 unique categories, which is computationally expensive in general and particularly for more complicated algorithms such SVM.

BM25 (Best Match) [5, 16, 17] 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:

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

where $K = k_1 * ((1 - b) + b * dl/avdl)$, dl is the length of D, and $avdl$ is the average document length, k_1, k_3, b are parameters, $IDF(q_i) = \log \frac{N - DF(q_i) + 0.5}{DF(q_i) + 0.5}$, 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, $QTF(q_i)$ 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:

$$BM25(Q, D) = \sum_{i=1}^{|Q|} w(q_i, D), \quad (2)$$

where w is the term weight obtained from the above Equation (1), $|Q|$ 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 Boolean filter and then BM25 model will return at most k top matches. And these top matched products' categories are the input of our KNN classifier. An input of our KNN algorithm consists of categories of k closest training titles given a test title. And the output of our KNN algorithm is the major category among categories 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.

Given *tf* and *idf*, and the query term frequency (*qtf*) that is obtained from a title in test set, a BM25 similarity score is calculated for each title in the training set and that title in test set. In KNN paradigm, the distance metric 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. When k=1, we assign the category of the top matched training title (document), i.e. the title with highest BM25 similarity score given that test title (query), as the predicted category. When k is set greater than 1, the category with the highest occurrence among the categories of the top matched training titles given a test title is deemed the predicted category. If the categories have same number of occurrence in the top matched training titles, we assign the category of the higher ranked matched training title(s) as predicted category. If no match is found, we assign "2296>3597>689" as predicted category, which corresponds to "book" (manually judging from training data).

Here is an example to show a typical product listing in our dataset. Given this item (query) in test dataset: "Morris Costumes SS88229 Rat Pumpkin Push In"

Category	Occurrence
1608>1150>1244>615	2
1608>1150>4810>4336	2
1608>1150>1244>615	1

Table 1: Category and Occurrence within top 5 documents' categories

Let's set k=5 and the searcher returns the item's top 5 matches in training set according to BM25 similarity score of a document and the query in descending order:

```
(1) "Morris Costumes SS87899 Talking Skeleton",
1608>1150>1244>615
(2)"Morris Costumes Wolfhound Latex Mask",
1608>1150>4810>4336
(3) "Morris Costumes SS09681 Witch Broom",
1608>1150>1244>615
(4) "Morris Costumes UR28998LG Immortal Large",
1608>4269>3031>1579
(5) "Morris Costumes Multi Princess Glove",
1608>1150>4810>4336
```

The KNN algorithm (majority voting) then counts the occurrences of the categories within these matches. As shown in Table 1 above, '1608>1150>1244>615' and '1608>1150>4810>4336' have the same number of occurrences among the top 5 matches' categories. Because '1608>1150>1244>615' is the category of matches with higher BM25 similarity score, the category '1608>1150>1244>615' is assigned as the predicted category.

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, and rounded rectangles are data inputs/outputs in interaction with Lucene API. Specifically, input product to 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, major model components of the product categorizing system (Figure 1) include:

- The input of Training Dataset as documents
- The tokenization of the document collection
- The indexing of the document collection
- The input of Test Dataset as queries
- The process of searching the index with test title with BM25 similarity to get top k most relevant training titles
- KNN algorithm: majority voting of category of top k matched training title(s) given a test title
- The final result of the product category generated from KNN classification algorithm

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 for its 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 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 is better than on several matches, since the individual match may be an outlier. In particular, k was set to 1, 3, 5, 7, 50 and 100. The Rakuten training dataset (800,000 product titles with categories) in tsv format is set as input for the data preprocessing and tokenization first. Specifically, we did formatting to the training data, where "w/out" was replaced by "without", "w/" was replaced by "with". With the application of the Lucene standard analyzer, which consists of a standard unigram tokenizer, the analyzer was able to break sentences into words discarding whitespace and punctuations. Furthermore, we continued to apply the analyzer to exercise the standard filter, the lowercase filter and then the stopword filter that uses a list of English stop words. All the item titles and categories in the product list are indexed in text field and string field respectively. The item titles in text field are tokenized for getting the document matrix of term frequency (TF) and inverse document frequency (IDF) while the item titles in string fields are not, since those are our target categories for later product list classification.

We considered the test data (200,000 product titles without categories) as query inputs. Same as the training data, we did formatting on test dataset, where "w/out" was replaced by "without", and "w/" was replaced by "with". Also, we removed the Lucene reserved word "OR", "AND", "NOT" at the end of a product title, since Lucene first filters items with Boolean model and these reserved words appearing in the end of an item title caused errors.

4.2 Results and Analysis

The results in the table above (Table 2) show the official results of our primary submissions. In this classification problem, the experimental results are evaluated with weighted-precision, recall 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=1 or 3 in KNN classification algorithm, our program

model	Precision	Recall	F1 score
kNN (k=1)	0.78	0.78	0.78
kNN (k=3)	0.78	0.78	0.78
kNN (k=5)	0.78	0.78	0.77
kNN (k=7)	0.78	0.78	0.77
kNN (k=50)	0.78	0.78	0.77
kNN (k=100)	0.76	0.77	0.76

Table 2: Performance on a subset of the test set.

achieved 0.78, 0.78 and 0.78 for the weighted-precision, recall and F1 score respectively in a subset of the test dataset. The results of k=1 and 3 are 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=5/7/50/100 are slightly lower than those of k=1 or 3. 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.

Aside from the Lucene indexing, we have tried to tune the parameters of the BM25 information retrieval (IR) model to get higher classification accuracy. We found a slight difference in between the tuning of the parameters. Specifically, by setting $k_1 = 1.2$, b=0.35, we achieved slightly lower results of (0.78, 0.77, 0.77) for weighted-precision, recall 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-precision, recall and F1 score respectively.

Apart from the BM25 model, we also used TF-IDF model to conduct searching on the product list. The results (0.77, 0.77, 0.77) are slightly lower than those of generated from BM25 IR model. This is because compared to TF-IDF that only incorporates term frequency (TF) and inverse document frequency (IDF), BM25 also takes the query term frequency (*QTF*) into account , which leads to higher accuracy rate as results.

5 BRIEF SUMMARY

This specific data challenge remains in the large-scale taxonomy classification domain and focuses on the fundamental problem of predicting product's category with given product's title in the taxonomy tree. In this paper, we described our classification experiments and results of the Rakuten 1M product listings in tsv format and exercised it on KNN-based approach with Lucene BM25 score.

The insights from participating this competition are as follows:

Lucene's Boolean filter really saves a lot of memory consumption and time. We actually tried a traditional Euclidean distance based KNN in Scikit-learn [14] on a server, and the algorithm required dimension reduction and still consumed much more memory and took more time to run.

We believe the results of (0.78, 0.78, 0.78) could be further improved in future. Specifically, 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) [19] can be combined with BM25 model.

ACKNOWLEDGMENTS

This work was supported by a discovery grant of Natural Sciences and Engineering Research Council of Canada (NSERC), a NSERC CREATE Program and ORF-RE Grant.

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. 2006. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and Applications)*. Springer-Verlag, Berlin, Heidelberg.

[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] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research* 12, Oct (2011), 2825–2830.

[15] 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

[16] Stephen E Robertson. 1997. Overview of the okapi projects. *Journal of Documentation* 53, 1 (1997), 3–7.

[17] 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.

[18] 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

[19] 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