Sensitive Keyword Detection on Textual Product Data: An Approximate Dictionary Matching and Context-score Approach

Duc-Hong Pham
Faculty of Information Technology, Electric Power University, Hanoi, Vietnam
hongpd@epu.edu.vn

Abstract - In this paper, we define and study a new problem in the field of natural language processing and data science called Sensitive Keyword Detection, which aims at detecting variants of keyword in each textual product. We propose a method using approximate dictionary matching algorithm and context-score to solve this new text mining problem in a general way. Given a product data set, for each document will be dectected keyword-variant pairs, then to determine if they are similar in semantic, we compare the context score between them. We conduct experiments on 1.189.690 textual sentences extracted from descriptions and titles of 300.000 products, each textual sentence contains an original keyword or a keyword variant. Experimental results show that important role of the context-score approach and the proposed method is effective when using context-score based on character and word embedding.

***Keywords*:** Approximate dictionary matching; Context-score; Sensitive keyword; Keyword detection; Keyword variant.

1. Introduction

E-commerce today is developing very fast, some typical e-commerce floors such as Amazon, Alibaba. Sellers can choose the trading floor that suits them to sell their products. Buyers can easily choose their favorite products to buy. However, to help buyers buy quality products, right types and brands. There are many problems that pose many challenges to e-commerce product managers, such as preventing the sale of counterfeit products, counterfeit products, and information about product titles and descriptions that are concise, informative. Many studies have done these problems, such as determine quality of Lazada's product titles [1,2,3], logo detection and recognition [4, 5, 6], detection of e-commerce anomalies [7, 8, 9]. Transaction fraud detection [10, 11, 12], query auto-completion [13, 14], generating e-commerce product titles and computing their score [15, 16], product duplicate detection [17, 18], a multimodal joint method for predicting attributes and extracting values from textual product data [19].

Approximate dictionary matching [25, 26] (keyword, string, and substring matching, matching in dictionaries) is the task of approximate computing to find all keywords in a keyword collection V such that they have similarity that is no smaller than a threshold α with a query keyword x. This task has a broad range of applications, such as spelling correction, flexible dictionary look-up, record linkage, and duplicate detection [20, 21], bioinformatics [22, 23], web crawling [24].

Word embeddings are also known as real-valued vectors, where each word is represented as a vector of low dimension. They are learned from linguistic neural network models [27, 28] to encode semantic information and relationships between words in a large unlabeled corpus. They can be used effectively to represent sentence level, text level, context level. Many problems have been applied successfully and achieved high results such as text classification, document classification, information retrieval, question answering, name entity recognition, sentiment analysis. Besides, character embbedings are generated from models [29, 30] that added missing information (e.g. the morphology of words) in word embeddings. Many studies use character embeddings that have achieved excellent results, such as detecting misogyny in Spanish tweets [31], detecting protein S-sulfenylation sites [32].

In this paper, we study the problem of identifying sensitive keywords on product data set. We apply approximate dictionary matching algorithm [25] to identify sensitive keyword variants. This algorithm is indexed a dictionary of keywords and searches for variations of the keyword in each textual product. In order to select the best keyword-variant pairs that are semantically similar, we propose using a context-score approach: similarity between the context of the keyword and its variation, high context-score that means that the variant and keyword are not only similarity in word form but also similarity in semantic space. Context-score is calculated based on characters or word embeddings.

The rest of this paper is organized as follows: Section 2 presents relevant concepts and problem definition; Section 3 presents the proposed method with the framework of sensitive keywords detection system,

each task and technique will be detailed; Section 4 describes our experiments and results. Some conclusions are presented in the last section.

2. Problem definition

In this section, we formally define some concepts and the problem of sensitive keywords detection. As a problem of analyzing and detecting sensitive keywords, working with the input is a set of product documents of a particular trading floor, these documents are posted by sellers. Note that we consider description and title of each product on the floor as a document.

Orginal keyword: Is the exact keyword about the brand (e.g. adidas, iphone, abbott) of the product or banned keywords (e.g., cannabis, shisha, smoking cannabis) that associate related to acts that are not allowed by law to be written in product descriptions by sellers.

Keyword variant: Is the keyword that the seller intentionally incorrectly spells or lacks or adds some characters to the original keyword in the process of product description for sale on e-commerce floors, in order to deceive buyers understand the semantics of those keyword variations as the original keyword. For example: the orginal keyword 'adidas' has variants as 'adjdas','aadidas', and 'addidass'.

Sensitive keywords: Include orginal keywords and variants of orginal keywords.

Similarity score of two keywords: Let X and Y denote the character sets of the keywords x and y, respectively. According to research [25], the cosine similarity between the two keywords x and y is.

$$\mathrm{Cos}\,ine(X,Y)=\frac{|X\cap Y|}{\sqrt{|X||Y|}} \qquad (1)$$

Similarity score of two contexts (context-score): Let c_x and c_y denote the contexts of the keywords x and y, respectively. The context score between the two contexts of x and y is.

$$context-score(c_x,c_y)=\frac{v(c_x).v(c_y)}{\|v(c_x)\|\|v(c_y)\|} \qquad (2)$$

In which $v(c_x)$ is a representation vector of c_x and $v(c_y)$ is a representation vector of c_y. These vectors can be computed base on charaters, word embeddings.

Sensitive Keywords Detection: Given an original keyword set $K=\{\mathrm{k}_1,\mathrm{k}_2,...,\mathrm{k}_{|K|}\}$, and a document set $D=\{\mathrm{d}_1,\mathrm{d}_2,...,\mathrm{d}_{|D|}\}$, we need to detect keyword-variant pairs (including orginal keywords and variants) in each document $d\in D$.

Note that one keyword-variant pair in this paper is understood as the following cases: (1) an original keyword in the set K and this keyword is also detected in the input document; (2) an original keyword in K and its variation is detected in the input document.

In fact, it's easy to determine if the original keyword appears in the input or not. And when we determine the variation of the keyword appears in the input or not is often more difficult. To determine whether the original keyword and its variant are really similar in form and semantics, we use two information to confirm: similarity score and context-score.

3. The proposed method

Derived from a product data set is posted by sellers on an e-commerce platform, for example: sendo.vn, tiki.vn, amazon.com, alibaba.com. We need to identify sensitive keywords in each document. Figure 1 depicts a framework in our proposed method. For a document, we perform following steps: (1) Re-processing; (2) Using an approximate algorithm to identify the set of candidates that are variations of keywords; (3) Select potential variation candidates; (4) Calculate the context-score and choose keyword-variant pairs that satisfy a score threshold. In the next section we will detail these steps.

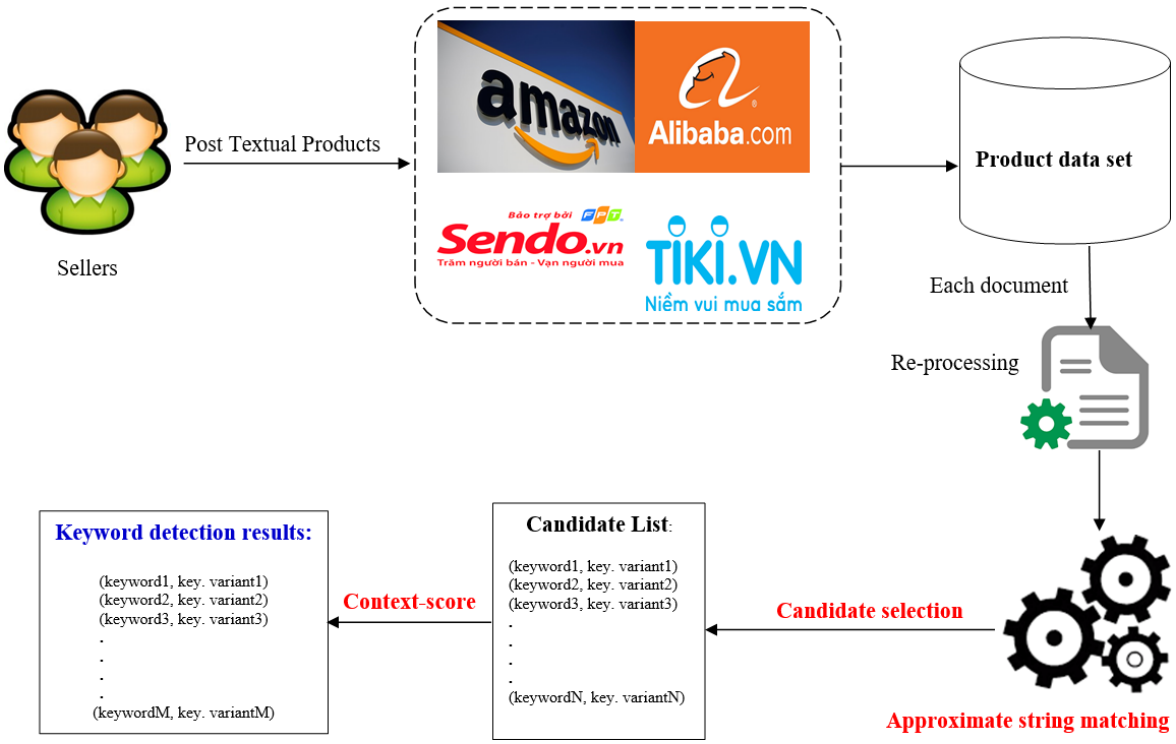

Figure 1. A framework of our proposed method

.

3.1. *Re-processing*

The mission of this task is to process documents, standardize, and word segmentation. Specifically, each document is stripped of html tags, meaningless icons. Each paragraph is marked as a corresponding sentence. We then standardize spaces, converting words in each document into lowercase words. We then use the long-matching algorithm and a dictionary of bi-gram and tri-gram keywords to tokenize the input document. After tokenizing is finished we use a dictionary of stop-words to remove stop words.

3.2. *Detecting sensitive keyword candidates*

The simplest way to identify sensitive keyword candidates is to use formula (1) to match with the original keywords in the set $K = \{k_1, k_2, ..., k_{|K|}\}$ and in the process of matching if similarity score between the original keyword and variant hits the threshold, these candidates will be selected. However, this way is impractical when the number of keywords $|K|$ is huge (e.g., more than one million). So, in order to identify variations of keywords quickly, we apply the approximate string matching algorithm [25], and denote it as SimString.

Algorithm 1: Sensitive keyword candidate detection

Input: $d = \{s_1, s_2, ..., s_l\}$, α - threshold for the similarity, n - grams of features
Output: R: list of keyword variant pairs

```
R = []
for each s ∈ d do
    Strs=n_gram(s)
        for each str ∈ strs do
            List_cadidate=SimString(str, α ,n)
            if len(List_cadidate)>0 then
                R = R ∪ List _ Cadidate
            end
        end
end
return R
```

Algorithm 2: Potential sensitive keyword candidate selection

```
Input: R: list of keyword variant pairs
Output: C: list of keyword variants
1  C = []
2  for each keyword v in list_keyword(R) do
3      list_v=get_variant(v,R)
4      r_remove=[]
5      list_v=sorted(list_v)
6      if len(list_v)> 1 then
7          top=1
8          for v1 in list_v do
               v_top=''
9              if top==1 then
10                 variant_top=1
                   v_top=v1
11                 top=top+1
12             for v2 in list_v do
13                 if len(v2)>len(v1) and v1 in v2 and cosinee(v1, keyword)==1 then
14                     r_remove.append(variant2)
15                 else
16                     if consin(v1, keyword)> consin(v2, keyword) then
17                         if (id of v1 in id of v2) and len(v2)>len(v1) and v1 belongs to v2 then
18                             r_remove.append(v2)
19                     else if (id of v2 in id of v1) and len(v2)<len(v1) and v2 belongs to v1 then
20                             r_remove.append(v2)
21                 else if len(v2)>len(v_top) and variant_top in v2 then
22                     r_remove.append(variant2)
23             for variant in list_variant do
24                 if v not in r_remove then
25                     C.append(v)
26     else
27         for variant in list_variant do
28             C.append(variant)
29 return C
```

For each document $d \in D$ that is splitted into sentences $\{s_1, s_2, ..., s_l\}$, each sentence $s \in d$ is made up of strings with 1 gram, 2 grams, and 3 grams. These strings will be used as inputs to the approximate algorithm.

Algorithm 1 is proposed for this task, where α - is threshold for the similarity, n is the number of ngrams to extract the feature for each input string of the string, the output is a list R of the keyword variant pairs. The function n_gram creates strings with size is 1, 2, 3, 4 gram. Assume the dictionary $K = \{k_1, k_2, ..., k_{|K|}\}$ has been indexed into SimString. In line 5, SimString return variants of the input string (str) and they are saved into *List_cadidate*. In line 7, the *List_cadidate* is union into *R*.

3.3. Potential sensitive keyword candidate selection

To avoid omitting variations of keyword appearing on documents, the SimString algorithm uses a low threshold to find all possible variations, however there will be many variations of the keyword identified. So we propose an algorithm to re-select the most potential candidates. Our idea is based on the position of words appearing on the input sentence, because the n-gram formation of strings are generated words, in which a specific word will be considered in three cases: (i) a word as a string (1-gram); (ii) a word combines with a word immediately after it as a string (2-grams); (iii) a word combines with a preceding word and a word after as a string (3 grams). Using an original keyword matching approach to these three cases, we select only the one most potential string, and we consider it as a potential sensitive keyword candidate.

A keyword can have many variants, let *list_v* denote a list of variations of the keyword *v* and we need to determine among those variations which is really a good variation of the keyword *v*. We first sort the variations in descending order of similarity between the keyword v and its variations. We then consider the following criteria: (a) If the variant *v1* belongs to *v2* and *Cosine(v1,v)* = 1 then remove *v2* from the variant candidate list of keyword *v*; (b) Between two variants *v1* and *v2*, is there *v1* belongs to *v2* or *v2* belongs to *v1* ? If so, if the

Cosine(v1,v)> Cosine(v2,v) then we remove *v2* from the list of candidate variations of keyword *v*. (c) If keyword *v* has only 1 candidate in *list_v*, this candidate will obviously be selected as a potential candidate.

We propose the Algorithm 2, which includes the necessary steps to identify potential candidates. In line 3, the function *get_variant (v, R)* is responsible for getting the candidate list of keyword *v* in *R*, line 4 initializes the r_remove list which will save the variants removed from *list_v*. In lines 15–16 which eliminates variants and save them to r_remove, pairs of keyword potential candidates, variants are included in the set *C* (lines 23-25). Lines 27-28 puts keywords with only one variation in the set *C*.

3.4. Context-score

This is the final task to be done in our proposed method. As mentioned, the above tasks help us detect keyword variations based on shape (form) and not considered the level of semantic similarity between them. For a semantic evaluation, the simplest and most effective way, we rely on the context of the original keyword and the variation keyword.

After identifying the keyword - potential variation pairs C, to re-evaluate whether the variant is semantically similar or not. We use the context-score method. This method uses a context set of keywords and the context set of keywords' variation keywords to semantically determine whether or not the context of the keywords is similar or not. The variants whose context-score score compared to its original keyword reaches a given threshold will be selected as the actual result to be determined for the sensitive keywords detection problem.

We propose the Algorithm 3 below which is the pseudo code needed to do the context-score calculation, for each keyword v in set C, in line 3 the *getContext* function is used to get the list of contexts for the keyword v, line 4 using *getContext* function to get list of contexts for variation of keyword v. The context-score between the two contexts we calculate by equation (2). The keyword-variant pair is selected to save the list X if context-score reaches the similarity score of context β.

Note that the context set of keywords P is identified from sentence set containing orginal keywords, these sentences are extracted from a product docment data. Context set of keyword variants Q is identified from sentence set containing keyword variants which extracted from the input document. The size of context in P and Q is equal. Figure 2, for an example about context: Given a sentence '*TPU Silicone Back Cover Samsum A01'*, we have '*Samsum*' which is a variant of keyword '*Samsung*', if the context window of size is 4 then variation '*Samsum*' has the contexts as '*Silicone Back Cove'*, and '*Back Cover A01'*.

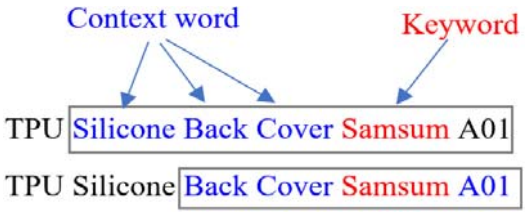

Figure 2. An example about context

Algorithm 3: Sensitive keyword detection

Input: C: list of keyword variants, P: context set of keywords, Q: context set of keyword variants, β - threshold for the similarity of contexts
Output: X: list of keyword variants

```
1  X = []
2  for each pair (keyword, variant) in C do
3      Contexts=getContext(keyword, P)
4      Context-variants=getContext(variant, Q)
5   for Context in Contexts do
6      for Context-variant in Context-variants do
7          if Cosine(Context, Context-variant)> β then
8              X.append(keyword, variant)
9              break
10 return X
```

4. Experiments

4.1. Data set

We use the data including 1.189.690 sentences on descriptions and titles of 300,000 products, which are collected from website sendo.vn, in which each sentence contains an original keyword or a keyword variant, and it is labeled in 0 or 1 by at least four annotators. We perform pre-processing on this data set as follows: use a standard stop word list to remove stop words and the words convert into lower cases. We then use the long-matching algorithm and a dictionary of 40,659 bi-grams and 22,327 tri-grams words to tokenize for each sentence. Table 1 show statistical data in our experiments.

Table 1. Evaluation data statistic

Number of sentences	1.189.690
Number of products	300.000
Number of keyword pairs (including original keywords and keywords variants)	280.219
Number of keyword variant pairs (Variant only)	2.3137
Number of keywords	753

4.2. Charater embeddings, word embeddings, and context set

Both Charater and word embedding play an important role in context representation. So to learn quality charater and word embeddings we use a data set of 30M products, which also collected from website sendo.vn and use continuous bag-of-words architecture Word2Vec, Fastext model of Gensim tool[*]. Both the Word2Vec and the Fastext models use the window size of context as 8, the word frequency threshold as 15 and the size of word vector is 50.

Also from this data set we construct a context set for 753 keywords. To do this, for each keyword we collect a set of text sentences containing it, then we use a context window of size 5 to define the contexts around the keyword. per sentence of text that contains it.

4.3. Experimental result

In order to apply the approximate string matching algorithm [25] to identify keyword variation candidates, we use the SimString[†] tool with a threshold of 0.65, the number of ngram to extract the feature is 2, and a dictionary is indexed by 753 keywords. The method of calculating the context-score is based on word embedding, the representing vector of the context is calculated by the average of the vectors representing of words that appearing on the context. The context-score threshold is 0.60. The sensitive keyword detection system performs through 4 main steps, which are detailed in section 3. Table 2 below shows the 15 keyword-variant pairs detected by the system.

Table 2. Top 15 pair keyword-variants of 5 keywords

Original keyword	**Keyword variant**	**Similarity score**	**Context-score**
adidas	adjdas	0.83	0.68
adidas	aadidas	0.92	0.63
adidas	addidass	0.77	0.75
balenciaga	baleciaga	0.947	0.66
balenciaga	alenciagà	0.842	0.83
balenciaga	balencia	0.889	0.72
camera_mini	camera_c2c_mini	0.846	0.88
camera_mini	camera_ezviz_mini	0.786	0.83
camera_mini	cameramin	0.94	0.69
samsung	sam_sung	0.933	0.62
samsung	samsum	0.769	0.89
samsung	sang_sung	0.75	0.91
abbott	abbot	0.909	0.63
abbott	abott	0.909	0.73
abbott	abot	0.80	0.89

[*] https://pypi.org/project/gensim/
[†] https://pypi.org/project/simstring-pure/

Through Table 2 we see that the variation keywords detected by the system all satisfy context-score > 0.60 and Similarity score between keyword and variation is greater than 0.65. The brand tag ‘adidas’ has the variations ‘adjdas’, ‘aadidas’, and ‘addidass’. The original keyword 'balenciaga' has the variations ‘baleciaga’, ‘alenciagà’, and ‘balencia’. Meanwhile the original keyword ‘camera_mini’ has the variations ‘camera_c2c_mini’, ‘camera_ezviz_mini’, and ‘cameramin’. ‘sam_sung’, ‘samsum’, and ‘sang_sung’ are variants of keyword ‘samsung’. ‘abbot’, ‘abott’, and ‘abot’ are variants of the dairy brand keyword ‘abbott’.

In order to evaluate the proposed method, we use three methods of context-score calculation: (i) base on word embedding; (ii) base on character embedding; (iii) character similarity. For method (i) and (ii) we use the formula (2), and method (iii) uses the formula (1). Two evaluation cases: (a) ***Overall*** means we evaluate on sentences contaning both original keywords and keyword variants; (b) ***Variant only*** means we only evaluate on sentences contaning keyword variants. The common metrics used for evaluation are Precision, Recall, and F1-Score, the results achieved by each method are shown in Table 3.

Table 3. Compare the results of determining keywords between different context-score methods

Context-score method	Keyword Detection Case	Precision	Recall	F1-Score
Character similarity	Overall	88.55	87.90	88.22
	Variant only	78.37	75.18	76.74
Character embedding	Overall	98.14	97.19	97.66
	Variant only	85.06	79.01	81.93
Word embedding	Overall	98.16	97.59	97.87
	Variant only	85.44	82.13	83.75

We found that the method using character similarity gave worse results than the method based on character embedding and word embedding. This proves that the context-score calculated based on the semantic information of the word or character gives better results. Also, when using word embedding to calculate the context-score for better results when using character embedding, this proves that character level information does not affect context-score as much as the detect keyword results.

The proposed method is heavily influenced by the return results of SimString, although the very low detect keywords threshold of 0.65 is chosen, there are still some cases where SimString can not dectect. In Table 4 we show 10 cases that SimString did not detect.

Table 4. Show 10 cases that SimString did not detect

Original keyword	Keyword variant	Similarity score
adidas	adidasvietnam	0.63
Apple	Applewatchband	0.53
bioderma	biodermanướctẩytrang	0.57
yanhee	bộttắmtrắngyanhee	0.52
zara	khăntắmzara	0.53
blackberry	sảnphẩmblackberrybold	0.64
zippo	hopquetzippo	0.59
xiaomi	kichsongwifixiaomi2	0.48
bosch	Boschaquatak	0.59
chanel	chanelclassic	0.63

Through Table 4, we see that brand keywords like 'adidas', 'Apple', 'bioderma' fall into the 'Adidasvietnam', 'Applewatchband', 'bioderma' variant keywords, all of which are very important variants, but SimString was undetectable. In a general observation, most of the cases that SimString is not detected are relatively low scores compared to the low keyword, and lower than the SimString search threshold.

5. Conclusion

In this paper, we have defined a new problem in the field of natural language processing and data science called Sensitive Keywords Detection, which aims at detecting variants of keyword in each textual product. This problem is meaningful in the process of supporting e-commerce floor managers to censor fake products, counterfeit products, and brand counterfeit products. A method uses approximate dictionary matching algorithm and context-score is proposed to solve the problem in a general way. Through experimental results, we show that the proposed method is effective when using context-score based on character and word embedding.

In the future, we aim at studying more rules to capture the cases which the SimString algorithm fails and deploying a sensitive keyword detection system to use in practice with a large number of keywords. We also aim to use this research results in other problems, such as automatic discovery of variations of brand keywords, spell checking for keywords in search engine.

References

[1] A. Bhonde, S. Agarwal, "Multi-layer stacking with diverse supervised models to determine quality of product titles," in *International Conference on Information and Knowledge Management*, *AnalytiCup*, 2017.
[2] M. Nicosia, A. Moschitti, "Lazada Product Title Quality Challenge: Constructing Features for a Diversified Ensemble of Classifiers," in *International Conference on Information and Knowledge Management*, *AnalytiCup*, 2017.
[3] K.Singh, V.Sunder, "Lazada Product Title Quality Challenge: An Ensemble of Deep and Shallow Learning to predict the Quality of Product Titles," in *International Conference on Information and Knowledge Management*, *AnalytiCup*, 2017.
[4] Y. Zhang, M. Zhu, D. Wang, and S.Feng, "Logo Detection and Recognition Based on Classification, International Conference on Web-Age Information Management," in *2014 WAIM Conference on Web-Age Information Management*, 2014, pp. 805-816
[5] T. Mudumbi, N. Bian, Y. Zhang and F. Hazoume, "An Approach Combined the Faster RCNN and Mobilenet for Logo Detection," *Journal of Physics Conference Series*, 2019
[6] V. Murugan, V.R. vijaykumar, and A. Nidhila, "Vehicle Logo Recognition using RCNN for Intelligent Transportation Systems," in *2019 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET),* 2019, pp. 107-111
[7] M. Bozbura, H. Tunc, M. Kusak, C. Sakar, "Detection of e-Commerce Anomalies using LSTM-recurrent Neural Networks," in *Proceedings of the 8th International Conference on Data Science, Technology and Applications*, 2019, pp. 217-224
[8] Z. Yang, S. Cao, and B. Yan, "Using linear discrim-inant analysis and data mining approaches to identifyE-commerce anomaly," in *Proceedings of 2011 7th Inter-national Conference on Natural Computation*, 2011, pp. 2406–2410.
[9] A. R. Yelundur, S. H. Sengamedu, and B. Mishra, "E-commerce Anomaly Detection: A Bayesian Semi-Supervised Tensor Decomposition Approach usingNatural Gradients," *arXiv e-prints. arXiv:1804.03836*, 2018
[10] L. Zheng, G. Liu, C. Yan; C. Jiang, "Transaction Fraud Detection Based on Total Order Relation and Behavior Diversity," *IEEE Transactions on Computational Social Systems*, vol. 5, 2018, pp. 796 - 806
[11] S. Cao, X. Yang, C. Chen, J. Zhou, X. Li, and Y. Qi, "TitAnt: Online Real-time Transaction Fraud Detection in Ant Financial," in *Proceedings of the VLDB Endowment*, 12(12), 2019, pp. 2082 - 2093
[12] Z. Zhang, X. Zhou, X. Zhang, L. Wang, and P. Wang, "A Model Based on Convolutional Neural Network for Online Transaction Fraud Detection," *Security and Communication Networks*, 2018
[13] J.Y. Jiang, Y.Y. Ke, P.Y. Chien, "Learning user reformulation behavior for query auto-completion," in: *Proceedings of the 37th international ACM SIGIR conference on research & development in information retrieval*, 2014, pp. 445–454
[14] L. Ramachandran, U. Murthy, "Ghosting: contextualized query auto-completion on Amazon search," in: *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, 2019, pp. 1377–1378
[15] Jose G. Camargo de Souza, M. Kozielski, P. Mathur, E. Chang, M. Guerini, M. Negri, M. Turchi, and E. Matusov, "Generating E-Commerce Product Titles and Predicting their Quality," in *Proceedings of the 11th International Conference on Natural Language Generation*, Association for Computational Linguistics, 2018, pp. 233–243
[16] M. R. Mane, S. Kedia, A. Mantha, S. Guo, and K. Achan, "Product Title Generation for Conversational Systems using BERT," *arXiv preprint arXiv:2007.11768*, 2020
[17] A. Hartveld, M. V. Keulen, D. Mathol, T.V. Noort, T. Plaatsman, F. FrasincarEmail, and K. Schouten, "An LSH-Based Model-Words-Driven Product Duplicate Detection Method," in *International Conference on Advanced Information Systems Engineering*, 2018, pp. 409-423
[18] B. West, K.A. Jadda, U. Ahsan, H. Qu, and X. Cui, "Interpretable Methods for Identifying Product Variants," in *WWW '20: Companion Proceedings of the Web Conference*, 2020 pp. 448–453
[19] T. Zhu, Y. Wang, H. Li, Y. Wu, X. He, and B. Zhou, "Multimodal Joint Attribute Prediction and Value Extraction for E-commerce Product," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2020, pp. 2129–2139
[20] Henzinger, Monika, "Finding near-duplicate web pages: a large-scale evaluation of algorithms," in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2006, pp. 284– 291
[21] G.S. Manku, G. Singh, A. Jain, and A.D. Sarma, "Detecting near-duplicates for web crawling," in *WWW '07: Proceedings of the 16th International Conference on World Wide Web*, 2007, pp. 141–150
[22] S. Kurtz, J. V. Choudhuri, E. Ohlebusch, C. Schleiermacher, J. Stoye, and R. Giegerich, "REPuter: the manifold applications of repeat analysis on a genomic scale," *Nucleic Acids Research*, 29(22): 4633–4642, 2001
[23] G. M. Landau, J. P. Schmidt, and D. Sokol. "An algorithm for approximate tandem repeats," *Journal of Computational Biology*, 8(1):1–18, 2001.
[24] G. S. Manku, A. Jain, and A. Das Sarma, "Detecting near-duplicates for web crawling," in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 141–150
[25] N. Okazaki, and J. Tsujii, "Simple and efficient algorithm for approximate dictionary matching," in *Proceedings of the 23rd International Conference on Computational Linguistics*, 2010, pp. 851–859
[26] A. Cislak, Sz. Grabowski, "A practical index for approximate dictionary matching with few mismatches", in *COMPUTING AND INFORMATICS*, VOL 36, NO 5, 2017, pp. 1088-1106
[27] Yoshua Bengio, R. Ducharme, Pascal Vincent, Christian Janvin, "A Neural Probabilistic Language Model," *Journal of Machine Learning Research*, 3(6):932-938, 2003
[28] T. Mikolov, G. Corrado, K. Chen, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," 2013
[29] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov ,"Enriching word vectors with subword information," *Trans. Assoc. Comput. Linguist*, 2017, 5, pp. 135–146.
[30] M.E. Peters, M. Neumann, M. Iyyer, and M. Gardner, "Deep contextualized word representations," in *Proceedings of the NAACL-HLT*, 2018, pp. 2227–2237
[31] José Antonio García-Díaz, Mar Cánovas-García, Ricardo Colomo-Palacios, Rafael Valencia-García, "Detecting misogyny in Spanish tweets. An approach based on linguistics features and word embeddings," *Future Generation Computer Systems*, vol. 114, 2021, pp. 506-518
[32] D.T. Do, T.Q.T. Le, N.Q.K. Le, "Using deep neural networks and biological subwords to detect protein S-sulfenylation sites," *Briefings in Bioinformatics*, 2020