A Framework for a Large-Scale B2B Recommender System

Azadeh Ghafari Nia, Jie Lu, Qian Zhang
Decision Systems and e-Service Intelligence Laboratory,
Centre for Artificial Intelligence
University of Technology Sydney, Australia
Email: azadeh.ghafarinia@student.uts.edu.au,
Jie.Lu@uts.edu.au, Qian.Zhang-1@uts.edu.au

Mario Ribeiro
Sydney, Australia
Email: info@marioribeiro.com.au

***Abstract*—The aim of a recommender system is to suggest relevant items in order to improve purchasing experience and minimise information overload. Despite extensive research in the area of B2C recommender systems, business-to-business distributors can have a complexity of the items and customers to deal with. A unique approach to recommendation with an emphasis on knowledge components is needed for such businesses. In a B2B distribution scenario one can not just rely on purchase history of customers to use collaborative filtering or content-based methods. A number of use cases have to deal with a list of queries to match against a large data set of items. Furthermore, the data can be very sparse due to the large number of items and the demographics of the customers. It is critical that category specific features be carefully analyzed for any recommendation. In this paper, we propose a large scale industrial B2B recommender system framework to deal with the above questions. Our proposed framework has the ability to deal with the huge number of items and make use of the side information accumulated in the real-world industry at the same time.**

***Index Terms*—recommender systems, business to business services, hybrid recommender systems, knowledge engineering**

I. Introduction

A Business-to-Business (B2B) distributor might have hundreds of thousands of products across tens of categories. This is known as an 'information overload' challenge for businesses [1]. It is a difficult task to identify the right product for a customer considering the diversity of the range and subtle differences between items when it comes to specialty categories. The challenge is more obvious when there is a list of thousands of products as a text input from a Business-to-Business (B2B) customer. There are a lot of abbreviations and variations of naming in the industry which makes it difficult to find the right item without the expert domain knowledge. An expert employee can receive a short text as the input to find the item and identify the category and narrow down to one or a few products. It is usually very costly and timely for a company to get a new employee to that level of expertise; In such a scenario technology can come very handy.

Recommender systems are primarily designed to assist individuals who are short on experience or knowledge deal with the vast number of choices in relation to items [1]. The explosive information on the web and the rapid increase of products or services has given users a huge amount of choice, which make users suffer when making decisions. Recommender systems take advantage of various sources of information including feedbacks of users to predict preferences of users in relation to different items [10]. This area of research has been the focus of great interest for the past twenty years from both academia and industry. Research in this field is motivated by the potential profit recommender systems have generated for businesses such as Amazon [3]. p_0^0

Recommender system studies typically target individual consumers in online shopping platforms but there seems to be a lot of room for improvements in the B2B space. Whilst the principles remain quite the same, the nature of B2B environments and large scale retailers and their business customers are very unique. First, the number of items for such retailer to choose from is huge. Given on a request from the user, it is very difficult to recognize which item matches to the current user's demand. Second, to discriminate the difference between some similar items/categories, we need to integrate their related attributes or descriptions into the recommender system. It is vital to deal with the vague and uncertain information in the text description. Third, the users in B2B are not like one customer in B2C recommender systems, but can vary from individuals to large enterprise. How to model the preference of those users are not properly handled by any existing recommender systems.

While having an efficient recommender system can improve business processes and facilitate new sales opportunities, it should rely on accurate knowledge about the business and detailed information on the products and their features. This paper elaborates the efforts to design a large-scale recommender system for a B2B industrial retailer to recommend and propose items based on customer input across many categories. Machine learning techniques (supervised and unsupervised) are used to develop the engine and the results will be compared against the input acquired from the category experts. The proposed framework includes a classifier component, a feature extraction component and a hybrid recommender. The success of item classification as the step one will directly suit

This research is supported by an Australian Government Research Training Program Scholarship.

the B2B retailer's requirements, however the full framework implementation will bring a clear saving in the labor costs including the number of employees and the amount of time to train an employee across many product categories.

The remainder of the paper is as follows. Section II reviews related literature on recommender systems. The proposed framework and related components are demonstrated in Section III. The case study has been elaborated in section IV. Finally, conclusion and further study are given in Section V.

II. Related Work

Related work on the most relevant topics to this research has been provided in this section. First we review recommender systems, then...

A. Recommender Systems

In 1998 [2] introduced Assistant Agents as software systems which are created to perform tasks on behalf of the user with easy and efficient interactions. Since then the filed which is known as recommender systems now had a lot of interest and hundreds of articles have been published around it.

The significant recommendation approaches are Stereotyping, content-based (CB) filtering, collaborative filtering (CF), knowledge-Based (KB), co-occurrence recommendations, graph-based, global relevance and hybrid approaches.

Stereotyping or demographic approach groups the users into fixed classes, e.g. male/female and recommends different items based on the class [6] and in some cases personalization of the recommendations are demanded [12]. An example on demographic approach is to classify the users based on income level or annual spend and provide specific recommendations for each class. CF uses the rating profiles of the users with a similar history of ratings to suggest items such as movies. It mainly employs nearest neighbour techniques to identify the closest match [7], [9]. Global relevance approach introduced by [5] is defined as "a non-parametric probabilistic model which can measure the context-based relevance between a citation context and a document.", this method is often used as a secondary ranking mechanism along with another approach. CB approach is the most common recommendation approach [10] and it consider the features of the items and associates them with user ratings, then it extracts the common features that users have rated the highest and produces the recommendations subsequently. Knowledge-Based systems use the preferences and inferred requirements of the users to suggest an item [9]. Co-occurrence recommendations are based on the number of times two items are appeared in the same basket or context. It was introduced first by Small [16]. It focuses on relatedness rather than similarity. As an example two types of papers might be very similar in features but pen and paper are very related and both are required for writing [4]. Graph-based methods use the inherent connections between entities; each item is a node and the edges are created based on relations or similarity of features of two nodes [11], [17].

Here we name some drawbacks of individual recommender system approaches. Content-based filtering models the users and extracts the features for items and uses similarity measures to find similarities. If the number of users and items is large, it will need significant computing resources. Another downside of CB could be overspecialized recommendations which lead to recommended items being too similar to the items the user already has [10].

Sparsity is a problem associated with collaborative filtering approach. For CF it is desired for a system to have a lot of users and relatively smaller set of items.In an opposite situation with a large set of items and smaller set of users there are less data points such as user ratings to rely on. Also the explain-ability of CF is limited to the fact that the item is liked by other users [10]. Furthermore CF needs more computing time comparing to CB. Another known drawback for CF is the cold start issue, i.e. the lack of ratings for a new user, item or system [14]. One way to overcome cold start problem is to infer implicit ratings based on user interaction with items [15] but [4] believes this voids the real advantage of CF which is having the quality user ratings.

Hybrid recommender systems have been introduced to combine some of these approaches and deal with some inefficiencies of one recommendation approach. The hybrid approaches such as content boosted collaborative filtering [3] try to combine some of these techniques to eliminate the effects of sparsity and other issues with any one of the techniques. Usually global ranking and graph methods are used along with CF or CB to improve the results. [8] is another sample of a cross-domain hybrid recommender system with kernel-induced knowledge.

According to [4] a vast majority of recommender systems researches do not translate to practice and only a handful seem to be publicly available and online.

B. Short Text Classification

Short-text classification is a crucial task in many natural language processing (NLP) areas. There has been a variety of approaches to address this task, such as: building dependency trees with conditional random fields [18], support vector machines (SVMs) with rule-based features [19],recursive neural networks [34], combining SVMs with naive Bayes [20], convolutional neural networks (CNNs) [29]. This study focuses on CNN for the classification of short text. The input to the CNN is the word embedding vectors with more details in the subsequent sections.

C. Unsupervised Learning

Duda et al. describes five main reasons for choosing an unsupervised method in [23]:

- The cost of collection of sample patterns and labeling them can be very high.
- In some cases, the reverse direction of supervised learning might be desirable, i.e. unsupervised clustering of unlabeled data first, and then supervised labeling of the clusters.

- In case of a system dealing with changing patterns over time, unsupervised approach would improve the performance.
- Unsupervised methods are useful for identifying features for categorizations.
- It can be very helpful in the early stages of data exploration to examine the similarities and patterns.

Named entity recognittion (NER) is one of the NLP disciplines with a lot of different approaches including of unsupervised learning. [21] uses a bi-directional long-short term memory (BLSTM) network for NER. NER will take word embeddings as an input.

Word embeddings are the methods that one uses to represent a word. There has been numerous ways to represent a word, such as symbols, one-hot vectors or the more advanced real-valued vectors [24] that represent the relationship of a word in conjunction with other words in a numeric vector form. Some of the most recent method of extracting word vectors are Word2vec [25], GloVe (global vectors) which has been developed at Stanford University in 2014 [24], Poincare Embedding [26] in 2017 and ELMO for deep contextual word representations [22] in 2018.

D. Similarity Measures

To perform operations such as data mining tasks of clustering, classification, and information retrieval on any set, it is crucial to define a notion of distance or similarity between two entities [28]. The most commonly used similarity measure in text data mining and information retrieval is the cosine of the angle between vectors which represents the *text*. Once one has converted the corpus to an inner point distance matrix then one can apply simple nearest neighbor classifiers to the data. Where the inherent high dimensionality of the *text* features precludes a straightforward application of feature-based classification, strategies such as linear/quadratic classifiers, mixture models, and classification trees must be coupled with dimensionality reduction strategies. Deep learning approaches for text classification are increasingly discussed in the literature [27], [30].

The core functionality of many of the modern information systems is the ability to detect similarities between different segments of data while surveys, literature reviews, and experimental evaluations of these systems show that the simplistic use of similarity detection in such globally authored linked systems is not enough [32].

In IR and knowledge discovery systems must be able to justify the similarity or distance between information entities [33]. A similarity measure “is an algorithm that determines the degree of agreement between entities” [32]. “Similarity-based classifiers estimate the class label of a test sample based on the similarities between the test sample and a set of labeled training samples, and the pairwise similarities between the training samples.” [35]

“The core to measure similarity or distance between two information entities is required for all information discovery tasks (whether IR or data mining). It is crucial to use an appropriate measure both to improve the quality of information selection and to reduce the time and processing costs.” [32]

Three intuition have been provided by Lin [36] to define similarity:

- The commonality between A and B defines their similarity. The more they have in common, the more similar they are.
- The similarity is also related to the differences between two items. The more differences they have, the less similar they are.
- The maximum similarity between A and B is reached when A and B are identical, regardless of how much they have in common.

This is how it is formulated based on these assumptions: I(common(A, B)) is the commonality between A and B and the differences between A and B is measured by I(description(A, B)) - I (common(A, B). The similarity between A and B, $sim(A, B)$, is a function of their commonalities and differences; sim(A, B) = f (I(common(A, B)); I(description(A, B) and the domain of f is $(x, y)|x >= 0, y > 0, y >= x$ The similarity between a pair of identical objects is 1 and for any $y > 0, f(0, y) = 0$

Based on previous equations and assuming that the overall similarity of the two documents is a weighted average of their similarities computed from different perspectives. The weights are the amounts of information in the descriptions. This leads to the last assumption: for any $x1 <= y1, x2 <= y2$,

$$f(x1+x2, y1+y2) = (\frac{y1}{(y1+y2))}*f(x1, y1)+(\frac{y2}{(y1+y2)})*f(x2, y2)$$

At the end Lin defines the Similarity Theorem as: The similarity between A and B is measured by the ratio between the amount of information needed to state the commonality of A and B and the information needed to fully describe what A and B are:

$$sim(A, B) = \frac{logP(common(A, B))}{logP(description(A, B))}$$

In a word processing scenario, after defining the set of features for each word in a corpus, Lin defines below measure for similarity of two words: Let $F(w)$ be the set of features possessed by w. $F(w)$ can be viewed as a description of the word w. The commonalities between two words $w1$ and $w2$ is then

$$F(w1) \cap F(w2)$$

Subsequently, the similarity between the two words can be defined as:

$$sim(w1, w2) = \frac{F(w1) \cap F(w2)}{I(F(w1)) + I(F(w2))}$$

Text similarity measurement approaches have been categorized to three categories by [31]: string-based, corpus-based and knowledge-based. Also the steps on text similarity matching begin with word similarity and then summarize to sentence and paragraph similarity. Words can be lexically or semantically similar. Lexical similarity which is the shape and character sequence similarity and string-based algorithms are used for this measurement, but semantic similarity is focused on the context of the words, e.g. synonym and opposite. To measure the semantic similarity corpus-based and knowledge-based algorithms are used. String-based measures are categorized to character-based and term-based measures [31]; some of the more famous character-based measures are:

- Longest character substring (LCS) which is based on the longest continuous character string in common between the two string inputs.
- Damerau-Levenshtein is based on the number of operations needed to transform one string to the other.
- Jaro counts the number of common characters and order of them.
- JaroWinkler has a prefix scale to favour the similarities at the beginning of the text.
- Needleman-Wunsch [38] is a dynamic approach which divides the problem to smaller sets and combines the results back to the final measure result.
- Smith-Waterman [37] is another dynamic solution which tries to find partial similarities.
- N-gram is a sub-sequence of N characters or items from a list and the distance is computed by dividing the number of similar N-grams between the two strings by the maximal number of N-grams.

Here some of the most used term-based similarity measures are briefly explained:

- Block distance (Manhattan distance, absolute value distance, boxcar distance, L1 distance and city block distance are other names for it). In an assumed grid-like path, how much travel is required to get from one point to another. To measure this distance between two list of items or components this will be calculated per component and then summarized.
- Cosine similarity is used to measure the similarity between two vectors of an inner product space that measures the cosine of the angle between them.
- Dices coefficient is defined as twice the number of common terms in the compared strings divided by the total number of terms in both strings.
- Euclidean distance or L2 distance is the square root of the sum of squared differences between corresponding elements of the two vectors.
- Jaccard similarity is computed as the number of shared terms over the number of all unique terms in both strings.
- Matching Coefficient is a very simple vector-based approach which simply counts the number of similar terms, (dimensions), on which both vectors are non zero.
- Overlap coefficient is similar to the Dice's coefficient, but considers two strings a full match if one is a subset of the other.

Corpus-based similarity uses the information which is extracted from a large corpora to determine the similarity degree between words [31]. Knowledge-based similarity measures determine the scale of similarity between two words by driving information from semantic networks [31] and also Hybrid similarity measures defined by the same source.

III. Framework for a Large-scale B2B Recommender System

A variety of components are required in the proposed framework. We propose a framework that will break down the complexity by classification of the input by mapping it to the right category as shown in Fig. **??** and implement a cascade of category specific recommenders using mainly unsupervised methods followed by an involvement of human category expert to validate the results which are a set of features produced by clustering of products labeled data for each category.

There are 2 main stages for this approach: 1- Classification of the input, 2- CB/KB/Demographics on the input item and the user and a generic stage for pre-processing of the input to make it ready to be passed to the main stages.

A. Pre-processing

As the input to this system is text and we use some NLP techniques such as NER, the input must go through some initial steps to be prepared for the subsequent processes. Generally NLP based solutions have these steps at the beginning: tokenization, stemming, tagging and lemmatization. In tokenization the text is broken to smaller elements (words). A lot of NLP systems remove the digits to reduce the size of the corpus. Stemming aims to normalize how the words appear in a text so it will reform all the forms of a word to a common shape which is the stem, e.g. navigator, navigation, navigate would all convert to 'navigat'.

Since the input to this system is not exactly complying with natural language structure but more of an industrial form with abbreviations and style names and codes, we can not apply all these steps in pre-processing. In our case the digits can not be dismissed because they play an important role in identifying the relevant product. Moreover if we apply stemming on the input it will heavily reduce the accuracy of the recommendations. In safety product categories for example, normalizing abbreviations and style names could lead to choosing a totally wrong product. To summarize, in this work the main pre-processing step is tokenization with keeping the digits and no stemming, tagging or lemmatization.

B. The Classifier

The title or description for an industrial product is often a very short text, between 10 to 60 characters. As depicted in the proposed model in Fig. **??**, to be able to accurately identify an item we propose to predict the category first based on the provided input. Thus this is a short-text classification problem. Out of many different techniques for text classification, CNN, initially designed for computer vision, have become increasingly popular and proven outstanding performance in NLP. "A CNN considers feature extraction and classification as one joint task." [13]

We use a CNN to identify the category of the input query. The CNN with be trained based on the labeled data set for product categories. According to Fig. **??** The categories of products are $\{C_1, C_2, ...C_n\}$, so for any provided input, the classifier will provide a probability $\{p_1, p_2, ...p_n\}$, p_1 being the probability of the input belonging to C_1 and so on. A ranking would pick the category with the most likelihood, being C_k. This provides a context to the next step which extracts the features and having the context improves the quality of the feature extraction.

C. The Feature-based Recommendation

Having identified the category in previous step as C_k, the input query will be passed through an LTSM model for named entity recognition [21] against the feature set of the category C_k, the extracted features are $f_1, f_2, ...f_m$ and this provides the input to the main recommender components (CB, KB, Demographics). In terms of word embeddings, we have done experiments on GloVe and Word2vec embeddings as a part of this research. Our early results show that Word2vec suits our models best. Also since Word2vec can be implemented as continuous bag of words (CBOW) and skip-gram, we have ran tests using both methods and due to interchangeable location of term in an industrial product description, it seems that CBOW is a better representation because it is not sensitive to the order of the words and in many cases the words in an item description can appear in different locations.

The features extracted from the input will be represented as a vector and matched against the pre-extracted and stored feature vectors of all the items in the database. Cosine-similarity will be applied on the vectors to identify the closest match. Any preferences set based on user demographics will be applied on the result set to achieve the final ranking of recommended items. For some product categories there are a fixed set of rules dictated by the human expert which are included as a KB component in the solution. Fig. **??** depicts the components of this model and their interactions. A calculated accuracy measure is required which is capable of reflecting the role of each component. A test automation platform has been developed to allow for the many changes in hyper parameters and measure the results based on the labeled data in a timely fashion. This also allows to isolate the impact of any change in one of the system components in the overall outcome. The overall accuracy of the model is a product of classification model and the feature recognition (NER).

IV. Case Study

Due to the scope of this work, the framework will be designed according to an existing B2B industrial retailer requirements. There is a data set of 100,000 lines of text available from the industrial retailer for the development and improvement of this framework. All the text lines are labeled with the category of the products and the actual industrial product code. This provides sufficient input to measure the accuracy of the models proposed and developed in this research. In this work an item is equivalent to a product.

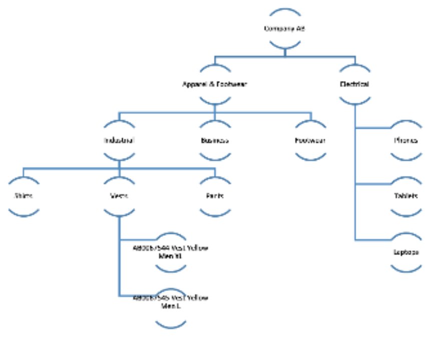

Fig. 1. A sample product hierarchy

Each product has a life-cycle, from getting introduced through an on-boarding process to becoming no-longer-available or removed from the range. Each product has a unique code to refer to and a description which sometimes has an imposed length limit by enterprise resource planning (ERP) systems which leads to creation of many abbreviations and shortenings on the words. During its lifetime, a product will be searched and found and get ordered and shipped and stocked and so on by the unique product code. A product can also be classified in a hierarchy and there could be more than one product hierarchy in use depending on the industry requirements. Fig. 1 shows how a sample set of products could be classified. In this figure we see how two sample products are classified under category and subcategories. The two sample products have a unique product code (e.g. AB000566) followed by the short text that describes the products.

The users (customers or accounts) of the subjected retail system are mainly other businesses. The size of the customers vary from individuals to large enterprises. There are multiple channels available to the users to place their transactions a.k.a omnichannel which includes physical stores, email, phone and e-commerce platform. An account might have several sub-accounts or customers and they can transact at any level. A large scale industrial retailer can trade from 100,000 items to over one million. The number of categories could be from 10 to over a hundred. Some of the categories are safety and personal protection equipment with a very specialized specification and targeting a specific usage. A lot of items in such categories have very similar features and subtle variations. Due to the scale of the customers, the number of items that just one customer needs to identify can be very huge at a given point of time. Without identifying the right item, an order can not be placed. So there is an extensive amount of labour involved in identifying the products which are mainly requested through trade stores, emails and phone calls and last but not the least bids and tenders including thousand of items to be quoted. The labour which supports these operations need long periods of training before getting involved in the tasks. The question is how such expertise in industrial products with such a vast range can be augmented in a recommender system.

V. Conclusion and Future Work

This research aims to cover an end-to-end large scale B2B recommender system. Having looked at all the different recommendation approaches and their characteristics, we have decided to design a hybrid system with CB, KB and demographic components to suit an industrial retailer. From a technical point of view, this work highly relies on word embeddings and NER using BLSTM for short text classification and feature extraction to build the recommender engine for CB and recognition of the features from the input.

More detailed research is to be done for modules in our framework. But first we have two major focus areas: item classification and feature extraction. For future work, we will amend and adjust our pre-processing part with NLP techniques so that we can map the input to an existing product catgory accuratly. Also, we will do in-depth analysis of deep learning classification methods with the application scenario described in our case study. In this process, the some new methods or algorithms that are suitable for our probelm will be developed.

References

[1] J. Lu and D. Wu and M. Mao and W. Wang and G. Zhang, "Recommender system application developments: a survey," Decision Support Systems, vol. 74, pp. 12-32, 2015.

[2] K.D. Bollacker and S. Lawrence and C.L. Gile, "CiteSeer: an autonomous web agent for automatic retrieval and identification of interesting publications," Proceedings of the 2nd international conference on Autonomous agents, pp. 116-123, 1998.

[3] P. Melville and R.J. Mooney and R. Nagarajan, "Content-boosted collaborative filtering for improved recommendations," Proceedings of the National Conference on Artificial Intelligence, pp. 187-192, 2002.

[4] J. Beel and B. Gipp and S. Langer and C. Breitinger, "Research-paper recommender systems: a literature survey," International Journal on Digital Libraries 17, pp. 305–338, 2016.

[5] Q. He and J. Pei and D. Kifer and P. Mitra and L. Giles, " Context-aware citation recommendation," Proceedings of the 19th international conference on World wide web, pp. 421-430, 2010.

[6] E. Rich, "User modeling via stereotypes," Cognitive Sciences 3(4), pp. 329-354, 1979.

[7] B. Yang and Y. Lei and J. Liu and W. Li, "Social collaborative filtering by trust," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 8, pp. 1633-1647, 2017.

[8] Q. Zhang and J. Lu and D. Wu, and G. Zhang, "A cross-domain recommender system with kernel-induced knowledge transfer for overlapping entities," IEEE Transactions on Neural Networks and Learning Systems, 2018.

[9] R. Bourke, "Hybrid Web Recommender Systems," Springer, The Adaptive Web: Methods and Strategies of Web Personalization, pp. 377–408, 2007.

[10] F. Ricci and L. Rokach and B. Shapira and P.B. Kantor, "Recommender Systems Handbook," Springer, Berlin, pp. 1–35, 2011.

[11] M. Mao and J. Lu and G. Zhang and J. Zhang, "Multirelational social recommendations via multigraph ranking," IEEE Transactions on Cybernetics, vol. 47, no. 12, pp. 4049-4061, 2016.

[12] Q. Shambour and J. Lu, "An effective recommender system by unifying user and item trust information for b2b applications," Journal of Computer and System Sciences, vol. 81, no. 7, pp. 1110-1126, 2015.

[13] A. Hassan and A. Mahmood, "Convolutional recurrent deep learning model for sentence classification," Ieee Access, 2018.

[14] J.B. Schafer and D. Frankowski and J. Herlocker and S. Sen, "Collaborative filtering recommender systems," Lecture Notes Computer Science 4321, 291, 2007.

[15] C. Yang and B. Wei and J. Wu and Y. Zhang and L. Zhang, "a ranking-oriented CADAL recommender system," Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries, pp. 203-212, 2009.

[16] H. Small, "Co-citation in the scientific literature: a new measure of the relationship between two documents," Journal of the American Society of Information Science. 24, 265269, 1973.

[17] Z. Huang and W. Chung and T.H. Ong and H. Chen, "A graph-based recommender system for digital library," Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries, pp. 65-73, 2002.

[18] T. Nakagawa and K. Inui and S. Kurohashi, "Dependency tree-based sentiment classification using CRFs with hidden variables in Human Language Technologies,"The Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 786-794, 2010.

[19] J. Silva and L. Coheur and A. C. Mendes and A. Wichert, "From symbolic to sub-symbolic information in question classification," Artificial Intelligence Review, Vol. 35 2, pp. 137-154, 2011.

[20] S. Wang and C. D. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics Short Papers, Vol. 2, pp. 90-94, 2012.

[21] J. P. C. Chiu and E. Nichols, "Named Entity Recognition with Bidirectional LSTM-CNNs," Transactions of the Association for Computational Linguistics, vol. 4, pp. 357-370, 2016

[22] M. E. Peters and M. Neumann and M. Iyyer and M. Gardner and C. Clark and K. Lee and L. Zettlemoyer, "Deep contextualized word representations," ArXiv, vol. abs/1802.05365,2018.

[23] R. O. Duda and P. E. Hart and D. G. Stork, "Pattern Classification and Scene Analysis," Wiley, 1996.

[24] J. Pennington and R. Socher and C. D. Manning, "GloVe: global vectors for word representation," Stanford University, 2014

[25] T. Mikolov and W. T. Yih and G. Zweig, "Linguistic regularities in continuous space word representations," HLTNAACL, 2013

[26] M. Nickel and D. Kiela, "Poincare Embeddings for Learning Hierarchical Representations," Advances in Neural Information Processing Systems 30, 2017.

[27] J. Bian and B. Gao and T. Y. Liu, "Knowledge-Powered Deep Learning for Word Embedding," ECML PKDD Part I, LNCS 8724, pp. 132-148, 2014

[28] C. D. Manning and H. Schutze, "Foundations of Statistical Natural Language Processing," MIT Press, 1999

[29] Y. Kim, "Convolutional neural networks for sentence classification," Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pp. 17461751, 2014.

[30] M. W. Berry and M. Browne, "Understanding Search Engines: Mathematical Modeling and Text Retrieval(Software, Environments, Tools)," Society for Industrial and Applied Mathematics, 2005.

[31] W. H. Gomaa and A.A. Fahmy, "A Survey of Text Similarity Approaches," International Journal of Computer Applications, V. 68,pp.09758887, 2013.

[32] B. Zaka, "Theory and Applications of Similarity Detection Techniques," Institute for Information Systems and Computer Media (IICM), Graz University of Technology,2009.

[33] I. F. Iatan, "Studies in Computational Intelligence," Springer, 2017.

[34] R. Socher and B. Huval and C.D. Manning and A.Y. Ng., "Semantic compositionality through recursive matrixvector spaces," Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1201-1211. 2012.

[35] Y. Chen and E.K. Garcia and M.Y. Gupta and A. Rahimi and A. Cazzanti, "Similarity-based Classification: Concepts and Algorithms," Journal of Machine Learning Research, Vol. 10, pp. 747-776, 2009.

[36] D. Lin, "An information-theoretic definition of similarity,",1998.

[37] F. T. Smith and S. M. Waterman, "Identification of Common Molecular Subsequences," Journal of Molecular Biology, Vol. 147, pp. 195-197,1981.

[38] B. S. Needleman and D. C. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," Journal of Molecular Biology, Vol. 48, 3, pp. 443–453, 1970.