A Review of a Tree-based Unsupervised Keyphrase Extraction Technique

Hyoin An and Yongqi Liu

May 4, 2022

1 Introduction

Automatic keyphrase extraction techniques are used in various settings related to digital information processing applications such as natural language processing [7] and recommender systems [8]. Utilizing these techniques, however, could be challenging when a method is domain specific so that expert-level domain knowledge becomes necessary, a large amount of training data is needed, and/or a method is computationally too extensive [6].

Rabby et al. (2020) propose a new tree-based unsupervised keyphrase extraction technique (TeKET) that is domain- and language-independent and does not require training data [6]. A variant of tree, called Keyphrase Extraction (KePhEx) tree, is introduced which determines the final keyphrases from candidate keyphrases. In addition, Rabby et al. (2020) derive a new measure, called cohesiveness index (μ), which calculates the degree of cohesiveness of a word with respect to the root word.

The other sections of this report are organized as follows. In Section 2, we demonstrate the data preprocessing procedure, which is used to provide a list of candidate keyphrases that are the input of the proposed method. The details of TeKET, together with the ideas and implementation of the five main steps, are presented in Section 3. Section 4 shows the results of applying TeKET on a real-world corpus dataset. Finally, we summarize our findings, discuss the connection between the proposed method and the class, and suggest a few possible future directions in Section 5.

2 Data Preprocessing

In this section, we illustrate preprocessing procedure of a corpus. The terms that we use in preprocessing and algorithms are presented in Section 2.1. The Part of Speech Tagging process is introduced in Section 2.2. Then further refining process, i.e. cleaning and stemming, is described in Section 2.3.

2.1 Terms

- **Noun phrases** are any phrases of the form "noun + adjective" or "adjective + noun".
- **Candidate keyphrases** are noun phrases that are extracted from a document and used to construct KePhEx trees.
- **Root word** (γ) is a (noun) word that goes into the root node in a KePhEx tree.
- **Similar keyphrases** are a collection of candidate keyphrases that share the same root word. We write σ for a similar keyphrase.
- **TF-IDF** stands for term frequency - inverse document frequency, and this value is calculated for each word and reflects the importance of a keyphrase. Term frequency measures "aboutness" (popularity) and inverse document frequency measures "informativeness" of a word.
- **Mu** (μ) denotes the cohesiveness index (with respect to the root). This measures the "cohesiveness" by considering how often a word appears in similar keyphrases. We will go over how to calculate this value exactly in Section 3.2.
- **Mamu** stands for minimum allowable mu and it sets a threshold that is used to prune a tree in Section 3.3.

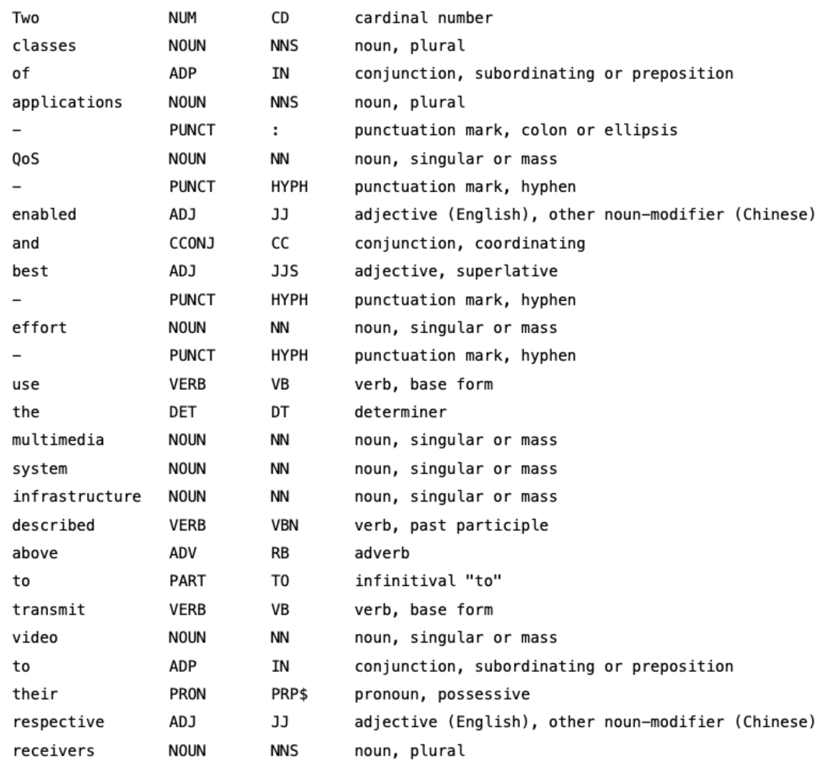
2.2 Part of Speech Tagging

To apply the proposed method for keyphrase extraction to a corpus, we first need to preprocess the corpus using a method called **Part of Speech (POS) Tagging** so that we can select candidate keyphrases. The POS Tagging is the process of allocating the part of speech tag or other philological class sign to each and every word in a sentence [3]. For example, consider the following example sentence:

(Example 1) "Two classes of applications - QoS-enabled and best-effort - use the multimedia system infrastructure described above to transmit video to their respective receivers."

Then, the result of POS tagging is shown in Figure 1. We can see that each word is classified to its class, like "applications" to noun and "use" to verb. Once we have this result for each sentence in the corpus, we can proceed to candidate keyphrase selection. Since the authors believe that keyphrases are usually noun phrases, we utilize the specified POS pattern for noun phrases, $(< NN.* > + < JJ.* >?)|(< JJ.* >? < NN.* > +)$, where NN and JJ are nouns and adjectives, respectively [4, 6]. That is, we collect any phrase with "noun + adjective" or "adjective + noun" form into our candidate keyphrases. When we pass *Example 1* through the preprocessing, we get two candidate keyphrases – *multimedia system infrastructure* and *respective receivers.*

After these phrases are collected, we process these using the TeKET method to filter out those that have less chance to be final keyphrases. The processing using the TeKET method is explained with details in Section 3.



```
Two             NUM     CD      cardinal number
classes         NOUN    NNS     noun, plural
of              ADP     IN      conjunction, subordinating or preposition
applications    NOUN    NNS     noun, plural
-               PUNCT   :       punctuation mark, colon or ellipsis
QoS             NOUN    NN      noun, singular or mass
-               PUNCT   HYPH    punctuation mark, hyphen
enabled         ADJ     JJ      adjective (English), other noun-modifier (Chinese)
and             CCONJ   CC      conjunction, coordinating
best            ADJ     JJS     adjective, superlative
-               PUNCT   HYPH    punctuation mark, hyphen
effort          NOUN    NN      noun, singular or mass
-               PUNCT   HYPH    punctuation mark, hyphen
use             VERB    VB      verb, base form
the             DET     DT      determiner
multimedia      NOUN    NN      noun, singular or mass
system          NOUN    NN      noun, singular or mass
infrastructure  NOUN    NN      noun, singular or mass
described       VERB    VBN     verb, past participle
above           ADV     RB      adverb
to              PART    TO      infinitival "to"
transmit        VERB    VB      verb, base form
video           NOUN    NN      noun, singular or mass
to              ADP     IN      conjunction, subordinating or preposition
their           PRON    PRP$    pronoun, possessive
respective      ADJ     JJ      adjective (English), other noun-modifier (Chinese)
receivers       NOUN    NNS     noun, plural
```

Figure 1: POS tagging result of Example 1

2.3 Refining process

Cleaning: Any candidate keyphrase, (i) that contains non-alphabetic characters; (ii) that contains single alphabetic word(s); or (iii) whose frequency is less than the **least seen allowable frequency (lsaf)** factor, is removed from our consideration. The lsaf factor is a constant value which separates non-popular keyphrases from popular keyphrases [6].

Stemming: To incorporate different forms of the same stem of a word, we take the stemmed form of words. For example, *manage* and *management* have the same stemmed form, *manag*. Then in practice, after we obtain final keyphrases, we can translate words in stemmed form back to the original form based on the context and their positions. For instance, the phrase *inform storag* would be translated back to *information storage* as a meaningful phrase.

3 Method

After obtaining a list of candidate keyphrases from preprocessing, the proposed method introduces a variant of binary tree for keyphrase extraction, called **KeyPhrase Extraction (KePhEx) tree**, from which the final keyphrases will be extracted and ranked. A KePhEx tree is grown for each root word, and all the similar keyphrases that contain the root word would be used to build and update the tree. Each node in the tree has three attributes: (i) a word, (ii) the TF-IDF value of the word, and (iii) a μ value.

TeKET consists of five main steps. The first step is to construct the KePhEx tree for a root word by adding nodes (Section 3.1). The next step is to update the μ values for all nodes in the tree (Section 3.2). The last step before landing on the final tree is pruning, which is discussed in Section 3.3. Section 3.4 shows how to extract the final keyphrases from the final KePhEx tree. With the ranking procedure presented in Section 3.5, we can select the top N phrases as the most relevant final keyphrases. We implement TeKET on Python, adapting the authors' codes which can be found in [5].

3.1 Step 1: Tree Construction

The formation of a KePhEx tree starts at selecting a root word γ, which is a noun and set as the root node. Then a tree with this root node is constructed, by adding nodes one at a time, using all the similar keyphrases that contain γ. For each similar keyphrase, we traverse from the root word γ to the leftmost word, and see if each of these words is qualified to turn into a node in the left subtree. Afterwards, we traverse from the word after γ to the rightmost word, and check if they are qualified to turn into nodes in the right subtree. The level (depth) of a node associated with the word w depends on the distance between w and γ in a similar keyphrase σ. The further away they are, the deeper level the node will be at. Each new node contains the associated word, the corresponding TF-IDF value, and an initial μ value of 1.

Figure 2 is an example of a newly formed tree using the first similar keyphrase (in stemmed form), *scalabl grid servic discoveri base*, where γ = *servic*. The root word *servic* becomes the root node of this binary tree, along with its TF-IDF value (not shown in the figure) and the initial μ value, 1. Then we go through the words before *servic*, add the nodes in the left subtree, and get two new nodes, `grid` and `scalabl`. After that, we go through the words after *servic*, add the nodes in the right subtree, and obtain another two new nodes, `discoveri` and `base`. All the μ values of these newly added nodes are set to be 1 in the current step.

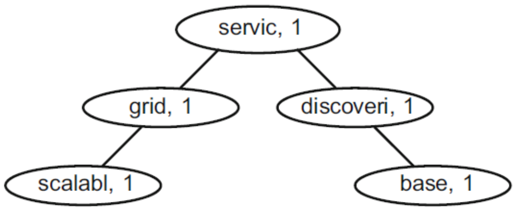


Figure 2: Example of a newly created tree using one similar keyphrase - *scalabl grid servic discoveri base*, where the root word is *servic* (Figure 4 of Rabby et al. (2020) [6])

A word w_1 is called a **predecessor** (**descendant**) of a word w_2 if w_1 comes before (after) w_2 in a keyphrase. When w_1 is right before (after) w_2, we call it an **immediate predecessor** (**immediate descendant**). Since the tree is binary, the root word in the final keyphrases can have as a root node no more than two children with at most one immediate predecessor and one immediate descendant, respectively. Each node in the left (right) subtree can have no more than two of its immediate predecessors (descendants). In general, across all the similar keyphrases, there could be different immediate predecessors or descendants for the root word, and more than two different immediate predecessors or descendants for the other words. Say a word, w, is qualified to be included as a new node at level l where there are already two children at that level (one child if at level 0, the root level). Then the term frequency of w is compared with the term frequency of the two children (left child first and then right child if w appears before γ; right child first and then left child if w appears after γ) respectively. The first time we find a child who has lower term frequency than w, the new node will replace that child node and the subtree with the child node being the root will be deleted. When both children have higher term frequency than w, no new node will be added into the tree.

3.2 Step 2: Cohesiveness Index (μ) Updating

The next step is to update the μ values. All the μ values in the tree will be updated simultaneously when we process one similar keyphrase. In fact, we can update the μ values as soon as all the words in a similar keyphrase have gone through the first step. Therefore, in implementation, we perform Step 1 and Step 2 consecutively for one similar keyphrase before moving on to the next phrase.

In this report, a phrase σ^* is called a **subphrase** of a keyphrase σ if σ^* contains coherent words in σ, including the root word. For instance, say a similar keyphrase, σ, consists of seven words in order: $\sigma = [w_1, w_2, w_3, w_4, \gamma, w_5, w_6]$, where γ is the root word. We say that $[w_3, w_4, \gamma, w_5, w_6]$ is a subphrase of σ while $[w_2, w_3, w_4]$ or $[w_1, w_4, \gamma, w_5, w_6]$ is not. Also, we say that a subphrase $\sigma^* = [w_3, w_4, \gamma, w_5, w_6]$ **forms a path** in the tree if (i) $\gamma - w_4 - w_3$ is a subpath of a root to leaf path in the left subtree, and (ii) $\gamma - w_5 - w_6$ is a subpath of a root to leaf path in the right subtree.

Suppose now we have processed all the words in a similar keyphrase σ through Step 1. Then, all the words in the longest subphrase of σ that forms a path in the tree would have their associated μ values increased by 1. Afterwards, the μ values of all other nodes in the tree would be decreased by 1.

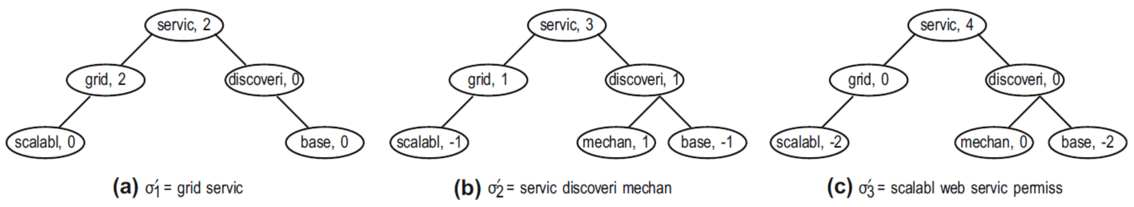


Figure 3: Several steps of tree processing for various similar keyphrases (adapted from Figure 5 of Rabby et al. (2020) [6])

Figure 3 shows several steps of tree processing, including Step 1 and 2, for various similar keyphrases. In panel (a), after processing the second similar keyphrase, $\sigma'_1 =$ *grid servic*, no new node is added, but the μ values change. We can see that the whole phrase, *grid servic*, forms a path in the tree. Therefore, nodes `grid` and `servic` have μ values increased by 1, and the other three nodes in the tree have μ values decreased by 1.

Moving from panel (a) to (b), the next similar keyphrase is σ'_2 = *servic discoveri mechan*. Since the word *mechan* is not in the tree and it appears after *discoveri*, it becomes a child of `discoveri`. Because it is a new node, the μ value is initialized to be 1. Also, since the subphrase *servic discoveri* forms a path in the tree, the μ values of these two words are increased, while the remaining three nodes, `grid, scalabl`, and `base` have μ values decreased.

If we process one more similar keyphrase, σ'_3 = *scalabl web servic permiss* (panel (c) of Figure 3), although the node `scalabl` is in the tree, *scalabl web servic* does not form a path in the tree. In fact, the words *web* and *permiss* are not in the tree. Therefore, only the root node `servic` can have μ value increased and all other nodes have their μ values decreased.

3.3 Step 3: Tree Pruning

Before proceeding to keyphrase extraction, the nodes with μ values less than mamu value, the threshold, are pruned from the KePhEx tree. When a root of a subtree is being pruned, that entire subtree is also deleted from the tree. The experiment conducted by Rabby et al. (2020) suggests that the proposed method performs the best when setting mamu = 2 [6].

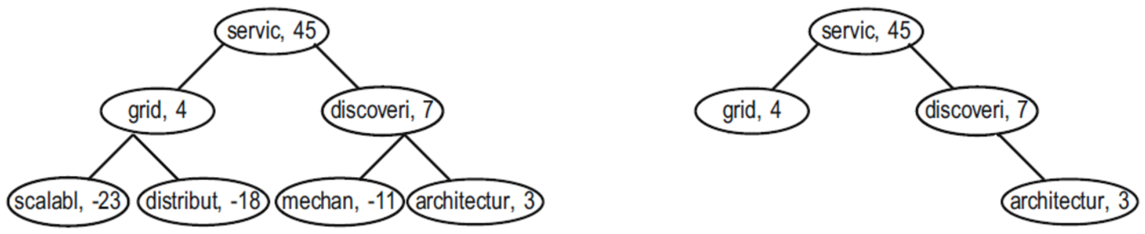


Figure 4: Pruning the constructed tree on the left with mamu = 2, results in the final KePhEx tree on the right (adapted from Figure 5 and 6 of Rabby et al. (2020) [6])

In Figure 4, on the left we have a tree constructed using all the similar keyphrases that contain the root word *servic*. On the right is the final KePhEx tree after pruning with mamu = 2. We can see that the nodes `scalabl, distribut`, and `mechan` are pruned from the tree since their μ values are less than 2.

3.4 Step 4: Final Keyphrases Extraction

Once we obtain a final KePhEx tree for a given root word, we can extract the final keyphrases from the tree. We will extract one from each root to leaf path in the left subtree. In this report, these phrases are called **left final keyphrases**. We will extract one from each root to leaf path in the right subtree. They are called **right final keyphrases**. We can also combine a left and a right final keyphrase to get a **combined final keyphrase**. For example, for the final tree shown on the right of Figure 4, we have only one left final keyphrase, which is *grid servic*. We also have only one right final keyphrase, *servic discoveri architectur*. Hence, we have $1 \times 1 = 1$ combined final keyphrase, *grid servic discoveri architectur*. In total, there are three final keyphrases for the root word *servic*.

So far we have described the process of obtaining the final keyphrases that contain a particular root word. Next, another root word would be chosen from the prespecified list of root words and Steps $1-4$ are repeated. This continues until all the words in the root list are considered. We prespecify the list of root words by collecting all the nouns with high term frequency from a document.

3.5 Final Keyphrases Ranking and Selection

In many applications in the digital information processing area, including document indexing [9] and recommender systems [8], it is common to utilize a certain number of top keyphrases. For that, once we have extracted the final keyphrases, we rank them using weights and select the top N phrases. The popular choices of N are 5, 10, and 15. In our implementation later (Section 4), we choose $N = 15$.

The **weight** of a keyphrase σ, denoted as ω_σ, is defined as

$$\omega_\sigma = \sum_{k=1}^{n} (tf)_k \times \sum_{k=1}^{n} \mu_k,$$

where n is the total number of words in σ, $(tf)_k$ is the TF-IDF value of the k^{th} word, and μ_k is the μ value of the k^{th} word.

4 Implementation

4.1 Dataset

We apply the data preprocessing and TeKET method described in Section 2 and 3 to the SemEval 2010 corpus dataset [2, 10]. This corpus is composed of total 244 documents, covering topics from the following four areas. The first one is Distributed Systems, denoted by C, and there are 59 documents on this topic. The second is information search and retrieval, denoted by H, and there are 64 documents on this topic. The corpus also has 60 documents about distributed artificial intelligence on multiagent systems, which is denoted by I, and 61 documents about social and behavioral sciences on economics, which is denoted by J [2]. Moreover, this corpus also comes with author- and reader-assigned keyphrases, which are treated as the gold standard and will be used to evaluate the performance of the proposed TeKET method.

4.2 Evaluation metrics

According to Goutte and Gaussier (2005), empirical evaluation plays an integral role in estimating the performance of natural language processing or information retrieval systems. The performance is typically estimated based on one-dimensional indicators such as the precision, recall and F1-score [1]. Thus, we would use these three metrics to assess the performance of TeKET.

Let $\kappa_{correct}$ denote the number of correctly matched keyphrases of our top 15 final keyphrases with the gold standard. $\kappa_{extract}$ is the number of extracted keyphrases from a document. Since we are going with top 15 final keyphrases, $\kappa_{extract} = 15$. We also denote $\kappa_{standard}$ as the number of keyphrases in gold standard keyphrases list. Then, precision, recall, and F1-score are defined as follows.

$$\textbf{Precision} : \rho = \frac{\kappa_{correct}}{\kappa_{extract}}$$

$$\textbf{Recall} : \zeta = \frac{\kappa_{correct}}{\kappa_{standard}}$$

$$\textbf{F1-score} : \phi = \frac{2 \times \rho \times \zeta}{\rho + \zeta}$$

Note that F1-score is the harmonic mean of precision and recall, and all three metrics are bounded between 0 and 1.

4.3 Results

The result of the proposed keyphrase extraction technique applied on one of the documents, H-49, is shown in Table 1. We choose Top 15 final keyphrases and three of them, *spatial autocorrel, language model score*, and *cluster hypothesi*, match to the gold standard.

Extracted: 'retriev time', 'spatial autocorrel', 'languag model score', 'document novelti', 'languag model', 'inform storag', 'high spatial autocorrel', 'acm press', 'languag model retriev', 'anyth autocorrel', 'autocorrel manifest', 'acm intern confer', 'cluster hypothesi', 'score diffus', 'averag precis'
Standard: 'perform predict', 'inform retriev', 'spatial autocorrel', 'autocorrel', 'cluster hypothesi', 'zero relev judgment', 'relationship of predictor+predictor relationship', 'predict power of predictor+predictor predict power', 'languag model score', 'rank of queri+queri rank', 'regular'

Table 1: Extracted and gold standard keyphrases for document H-49

The precision is 0.2, recall is 0.27, and F1-score is 0.23. These are close to the values that the authors obtained after they applied the algorithm to the entire collection of the datasets and took the average.

Next, Table 2 shows a more interesting result from another document, H-37. Among the 15 selected keyphrases, there is only one exactly matched keyphrase, *spam filter*, which leads the precision, recall, and F1-score to be all low as 0.07. We can still see, however, that the extracted keyphrases captured some important phrases that basically have the same meaning as some other phrases in the gold standard. For example, *svm* in the extracted phrases is actually the acronym of *support vector machin* in the gold standard. Also, *content-bas spam detect* is the same as *content-base spam detect.* One of the reasons why this situation happens seems to be related to the characteristic of a text dataset and also to the limitation of preprocessing stage to some degree.

Extracted: 'contentbas spam', 'content-bas spam detect', 'large-scal content-bas spam detect', 'spam detect', 'data mine', 'svm', 'spam comment', 'blog comment spam detect', 'compar perform', 'activ set', 'spam filter', 'blog identif', 'benchmark data set', 'comment spam detect experi', 'expens comput'

Standard: 'support vector machin', 'content-base filter', 'spam filter', 'blog', 'splog', 'link analysi', 'machin learn techniqu', 'link spam', 'content-base spam detect', 'bayesian method', 'increment updat', 'logist regress', 'hyperplan', 'featur map', 'spam filter'

Table 2: Extracted and gold standard keyphrases for document H-37

5 Discussion

In summary, we introduce the main idea of a keyphrase extraction method (TeKET) which bases on tree, implement the proposed method on real-word text documents, and get comparable results to those in Rabby et al. (2020) [6]. TeKET does not require training data or any prior knowledge in the domain or in the language. The proposed algorithms run fast once preprocessing is properly done.

We also find two connections to the class. First, TeKET is an unsupervised machine learning method, which unlike the supervised learning methods, can extract keyphrases without the need for human intervention. Second, this paper shows a different usage of binary trees in a non-statistics area to solve problems other than regression or classification.

In the future, if we are able to match two different words with the same meaning (e.g. *svm* and *support vector machin*, *content-bas* and *content-base*), the performance may be improved. Also, using a general tree, where each node can have more than two children, instead of binary tree may allow more various combinations of the keyphrases. Lastly, choosing a list of root words with high quality would lead to more pertinent final keyphrases, because the list of root words significantly impacts the selection of similar keyphrases. Also, ranking root words may prevent important-but-less-frequent phrases from being eliminated in the final list of extracted keyphrases.

References

[1] Cyril Goutte and Eric Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *European conference on information retrieval*, pages 345–359. Springer, 2005.

[2] Su Nam Kim, Olena Medelyan, Min Yen Kan, and Timothy Baldwin. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. *Language Resources and Evaluation*, 47:21–26, 08 2010.

[3] Deepika Kumawat and Vinesh Jain. Pos tagging approaches: A comparison. *International Journal of Computer Applications*, 118(6):32–38, 05 2015. Full text available.

[4] Gollam Rabby, Saiful Azad, Mufti Mahmud, Kamal Z Zamli, and Mohammed Mostafizur Rahman. A flexible keyphrase extraction technique for academic literature. *Procedia computer science*, 135:553–563, 2018.

[5] Gollam Rabby, Saiful Azad, Mufti Mahmud, Kamal Zuhairi Bin Zamli, and Mohammed Mostafizur Rahman. Automatic keyphrase extraction. `https://drive.google.com/drive/folders/1e2UrDtYqRAjAE5hso4oXobX_Djuo_VUW`, 2020.

[6] Gollam Rabby, Saiful Azad, Mufti Mahmud, Kamal Zuhairi Bin Zamli, and Mohammed Mostafizur Rahman. Teket: a tree-based unsupervised keyphrase extraction technique. *Cognitive Computation*, 12:811 – 833, 2020.

[7] Ronan G Reilly and Noel Sharkey. *Connectionist approaches to natural language processing*. Routledge, 2016.

[8] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

[9] Jennifer Rowley and Richard Hartley. *Organizing knowledge: an introduction to managing access to information*. Routledge, 2017.

[10] SIGLEX. Semeval-2010 corpus. `https://semeval2.fbk.eu/semeval2.php?location=data`, 2010.