Education and Information Technologies (2023) 28:16089–16125
https://doi.org/10.1007/s10639-023-11818-1

An unsupervised linguistic-based model for automatic glossary term extraction from a single PDF textbook

Ashraf Soliman[1]

Received: 1 September 2022 / Accepted: 13 April 2023 / Published online: 6 May 2023

Abstract

Term extraction from textbooks is the cornerstone of many different intelligent natural language processing systems, especially those that support learners and educators in the education system. This paper proposes a novel unsupervised domain-independent model that automatically extracts relevant and domain-related key terms from a single PDF textbook, without relying on a statistical technique or external knowledge base. It only relies on the basic linguistic techniques of the natural language processing: pattern recognition, sentence tokenization, part-of-speech tagging, and chunking. The model takes a PDF textbook as an input and produces a list of key terms as an output. Furthermore, the model proposes a novel classification of sentences from which the concept of defining sentences is proposed. The defining sentences are the main textual units that the model revolves around to identify the key terms. The architecture of the proposed work consists of 21 processes distributed across three phases. The first phase consists of five processes for extracting text from a PDF textbook and cleaning it for the next phases. The second phase consists of eight processes for identifying the defining sentences and extracting them from all the textbook's sentences. The last phase consists of eight processes for identifying and extracting the key terms from every defining sentence. The proposed work was evaluated by two experiments in which two PDF textbooks from different fields are used. The experimental evaluation showed that the results were promising.

Keywords Term extraction · Linguistic techniques · Natural language processing · Unsupervised machine learning · PDF textbooks · Intelligent Tutoring

✉ Ashraf Soliman
assoliman@msa.edu.eg

[1] Department of Management Information Systems, October University for Modern Sciences and Arts (MSA), Cairo, Egypt

1 Introduction

Textbooks are considered the main source of any education system. Therefore, almost all educators rely on them in order to prepare their teaching plans and syllabus, moreover, providing them as references to their students. This is because textbooks are high-quality textual resources that introduce the basic concepts and primary knowledge necessary to understand a specific domain (Alpizar-Chacon & Sosnovsky, 2021).

Recently, electronic textbooks are more preferable to a wide range of students and educators than physical or hard copy textbooks. This is because they give learners the ability to easily navigate through their pages, search for specific words or phrases, add editable annotations, and much more. Textbooks can be found in different electronic formats, such as HTML,.DOCX, PDF, etc. However, PDF textbooks are the most popular electronic textbook format that students and educators seek to have more over the other electronic formats (Alpizar-Chacon & Sosnovsky, 2021; Bast & Korzen, 2017). The difference between a document in PDF and other format is that a PDF document does not contain a normal text, but it actually contains thousands of basic objects, various compression techniques, font formats, lines, and curves. These objects in a PDF are instructions to draw shapes, images, and text (Whitington, 2011). The disadvantage of PDF textbooks is that they rarely save information about their structure (Tkaczyk et al., 2015), which affects the quality of the text extracted from them.

Textbooks are created by domain experts who put a considerable effort into identifying the main concepts and facilitating the basic knowledge of a domain. Furthermore, they greatly focus on organizing the content of textbooks in a logical sequence according to their understanding of the domain. This logical sequence appear in a form of a set of chapters, sections, and subsections that are provided with informative headings. (Alpizar-Chacon & Sosnovsky, 2021).

One of the distinguishing features of textbooks is that their authors focus on defining key terms explicitly in a way that makes learners keep reading without struggling to search for the meaning of new terminologies. Thus, they take care of defining simple terms first, then gradually defining more complex ones that are based on understanding the simple ones defined first. In good-quality textbooks, authors create a section, called glossary, at the end of the textbook. The glossary section is very helpful for learners, especially students. It provides all key terms that are defined and explained in different chapters in a textbook in one place with their definitions (Gacitua et al., 2011; Park et al., 2002), which enables learners to easily find and review all the key terms related to the important concepts introduced by the textbook.

In the literature, it is better to distinguish between two research areas relating to key terms (Campos et al., 2020): term extraction, and term assignment. In term extraction, researchers propose solutions for extracting all possible terms referring to all domain concepts given in a particular text document. The extracted terms in this research area are called key terms, glossary terms, keywords, or keyphrases. On the other side, researchers in the term assignment area propose solutions for labeling a document or a collection of documents with one or more terms that refer to a domain concept or more for the purpose of document classification. The selected terms can also be called key terms, keywords, or keyphrases, but not glossary terms.

A term in both areas is a word or group of words that refer to a particular domain concept (Dwarakanath et al., 2013), and, sometimes, terms can also be called concepts. (Castellví et al., 2001; Gul et al., 2022). Extracting terms from text has a vital role and is the foundation of various tasks, including text summarization, clustering, thesaurus building, opinion mining, categorization, query expansion, recommendation, information visualization, retrieval, indexing, and digital libraries (Campos et al., 2020). It can also play an important role in the academic publishing industry to perform tasks such as, recommending articles and books, highlighting missing citations to authors, suggesting available reviewers for submissions (Augenstein et al., 2017).

However, it is not a trivial task for experts to extract terms from text documents (Silva et al., 2014), as the manual term extraction consumes a lot of time and a great human effort (Berry et al., 2003; Liu et al., 2010; Xu & Zhang, 2021). Thus, there is an urgent need to use the computing power of computers for automatic term extraction (Babar & Patil, 2015). Another issue of term extraction is that it is subjective even if it is done manually, according to a study conducted in Silva et al. (2014). This study shows that about 60% of experts agree that identifying terms in a domain is a subjective task. Therefore, this indicates that the automatic extraction of terms may not be totally precise (Silva et al., 2014).

In automatic term extraction, there are two machine learning methods: supervised and unsupervised. Supervised term extraction requires labeling many training data sets to convert term extraction problem into a classification problem (Wang & Wang, 2019; Witten et al., 1999). This method trains a model on labeled training data sets in a particular domain and uses the trained model to determine whether a word in a text is related to that domain. The limitation of this method is that the model must be trained on large labeled documents in order to serve as training sets. Unfortunately, this large amount of labeled documents are not always available in many domains. Moreover, unlabeled documents require large amounts of manual extraction of key terms as well as labeling them in order to have these documents labeled. (Campos et al., 2020; Sun et al., 2020; Xu & Zhang, 2021).

An alternative solution is to consider unsupervised methods that include linguistic, statistical, and graph-based methods for term extraction (Campos et al., 2020; Sun et al., 2020). Unlike supervised methods, the unsupervised methods do not need training data sets. Consequently, this saves time and money of the manual term extraction as well as the manual labeling of the extracted terms in order to create the training data set (Campos et al., 2020). Another advantage over supervised methods is that they can extract terms from text belonging to any domain, as they do not need labeled data sets in the domain of the problem for a training process (Papagiannopoulou & Tsoumakas, 2020). For these advantages, unsupervised methods are preferred as an alternative approach for term extraction (Duari & Bhatnagar, 2019).

This paper proposes a model for an automatic unsupervised linguistic-based method for extracting key terms from a PDF textbook. It is based on the basic linguistic techniques of the natural language processing: pattern recognition, sentence tokenization, part-of-speech (POS) tagging, and chunking. The proposed model succeeds to extract relevant and domain-related terms from a single PDF textbook, even if a term occurs only once in the text. Consequently, the proposed model goes a step over methods relying on statistical techniques to filter domain-related terms from the extracted terms. These methods need to compute term frequencies and set a frequency threshold in order to select top terms that have a frequency exceeds the predefined threshold. It is worth mentioning that, unlike these methods, the proposed method can extract key terms without computing term frequencies or setting a frequency threshold. The proposed method also focuses on textbooks in PDF format in order to provide solutions that work around problems resulting from extracting text from PDF textbooks. This is because the ultimate goal of the proposed method is to facilitate extracting the important terms, for educators and learners, directly from their PDF textbooks, regardless of the domain of the textbook.

The paper is structured as follows. Section 2 discusses a number of recent related work. Section 3 explains the proposed model. Section 4 illustrates the model phases and architecture. Section 5 explains the foundation layer of the model. Section 6 explains the preprocessing phase and its processes. Section 7 describes the defining sentence extraction phase and its processes. Section 8 focuses on explaining the term extraction phase and its processes. Section 9 describes the experiments through which the proposed model is evaluated. Section 10 discusses the experiment result. Section 11 concludes the paper with future work trends.

2 Related work

Chacon and Sosnovsky (Alpizar-Chacon & Sosnovsky, 2021) use PDFBox[1] library to extract text from PDF textbooks and then propose two phases to extract the key terms from their index section. They propose 12 rules to identify the index section, including rules to identify the beginning of the index section, the layout of index pages where text is organized into two columns, the index entries of each column in every index page. In the first phase, they extract index terms from non-hierarchical index entries. In the second phase, they propose an algorithm to extract index terms from hierarchical index entries. Each hierarchical index entry consists of an unordered group of words that have to be arranged in a way that meets a term in the textbook. The algorithm is proposed to detect the right order of these words in order to detect different terms from this group of words. The algorithm uses the stem and lemma forms of each word in the hierarchical index entry and creates all their possible permutations. Then the algorithm breaks down the text into noun chunks, and every permutation is compared to every noun chunk. The matched permutations are chosen as key terms and added to the list of index terms extracted from the non-hierarchical index entry.

[1] https://pdfbox.apache.org/

Mishra and Sharma (Mishra & Sharma, 2020) propose a domain-specific approach to extract glossary terms from large-sized text requirement documents. First, they build a text corpus of the domain of home automation. The text corpus consists of the CrowdRE dataset proposed by Murukannaiah et al. in Murukannaiah et al. (2017); Murukannaiah et al., 2016) and the Wikipedia's home automation web pages[2] crawled by web scraping packages[3] in Python. Then the Python's Natural Language Toolkit (NLTK)[4] is used in the preprocessing phase to tokenize the textual data in the corpus into sentences. All the tokenized words are then converted into lowercase and all alphanumeric tokens and stopwords are removed. The NLTK tagger is used to tag tokens of each sentence and then text chunking is applied to identify noun phrase chunks. Lemmatization is then applied to lemmatize the generated chunks. The lemmatized noun phrases are considered candidate glossary terms. Then, a semantic filter is performed to select domain-related terms among these candidates. The semantic filter uses Word2Vec (Mikolov et al., 2013a, b) model for word embeddings and for computing semantic similarity scores using cosine similarity between candidates to identify all candidates related to the domain as final glossary terms.

Gul et al. (Gul et al., 2022) extract key terms from textual documents in education domain using common NLP techniques and DBPedia as a knowledge base. The NLP techniques are used to divide the textual documents into n-grams representing the candidate terms. All n-grams representing stopwords are removed based on a predefined stopword list. The frequency of each remaining n-gram is computed to remove n-grams that have a frequency less than a predefined threshold. Then the DBPedia is used to remove meaningless n-grams that do not have an entry in the DBPedia. The remaining n-grams are considered the key terms of the textual documents.

Campos et al. (Campos et al., 2020) proposed Yake!, an unsupervised statistical method to extract keywords from a single text document without relying on external sources such as Wikipedia as a knowledge base. They used the Python's Segtok[5] sentence segmenter to divide the document into sentences and every sentence into chunks. The chunks are blocks of characters separated by punctuation marks. Then the web_tokenizer module of the Python's Segtok segmenter is used to divide every chunk into tokens. Statistics of each token is computed, including token frequency and index of sentences where the token exists. Moreover, a co-occurrence matrix is created for each token to save its predecessor and subsequent tokens found within a given window. They also propose five features to be extracted for each token. A score for each feature is computed based on the previously computed statistics about each token. The scores of features of a token are used to compute the token score in a document. Candidate keywords are then generated from all n-gram tokens in every chunk. All keywords that are started or ended with a stopword existing in a given list of stopwords[6] are rejected, but those within a keyword are accepted. Then a keyword

[2] https://en.wikipedia.org/wiki/Home_automation.

[3] https://selenium-python.readthedocs.io/

[4] https://www.nltk.org/

[5] https://pypi.org/project/segtok/

[6] https://github.com/LIAAD/yake/tree/master/yake/StopwordsList

score is computed based on scores of all tokens forming the keyword. Keywords having lower scores are considered the most relevant in the domain.

Duari and Bhatnagar (Duari & Bhatnagar, 2019) propose a graph-based method called sCAKE in order to extract keywords from text. They follow the work in Rousseau and Vazirgiannis (2015) to extract candidate keywords as nouns and adjectives using the Apache OpenNLP toolkit[7] for tokenization and POS tagging the text. After that, they remove all stopwords and produce stemmed tokens using Porter Stemming Algorithm.[8] Stemmed tokens are used as candidates that are connected by the proposed Context-Aware Text Graphs (CAG) method that connects candidates related to each other based on their co-occurrences. They also propose the Semantic Connectivity based Word Scoring method (SCScore) that identify important and relevant candidates by scoring every candidate based on its relation with its neighbors and its position in the text. Then the candidate keywords are sorted according to their SCScore and the user can choose the top K candidates as the final keywords.

Dwarakanath et al. (Dwarakanath et al., 2013) propose a linguistic statistical technique to extract glossary terms from software requirement text documents. They identify requirement sentences by those that begin with an explicit label. In every sentence, all words in title case, capital case, in quotes, and URLs are considered acronyms which are automatically treated as glossary terms. All brackets and their contents are removed from sentences. Then every sentence is parsed by the Link Grammar (LG) (Sleator & Temperley, 1993) which is a dependency parser that provides information about the syntax of a sentence. They use the LG parser, instead of a POS tagger, to identify noun phrases and verb phrases to be considered candidate glossary terms. They classify nouns into concrete, process, and abstract nouns. Verbs are also classified into concrete and auxiliary verbs. Abstract nouns that do not refer to a physical object such as ‘capability’, and auxiliary verbs such as ‘is’ are not included in glossary terms. Then they use a statistical technique based on occurrence and frequency to choose the final glossary terms from those candidates.

Conde et al. (Conde et al., 2016) propose a tool called LiTeWi that uses TF-IDF, KP-Miner, and CValue techniques to extract domain-related terms from electronic textbooks. The TF-IDF statistical technique proposed in Salton and Buckley (1988) is used to identify terms in a document and then selecting candidate terms based on their frequencies. KP-Miner, which is a keyphrase extraction statistical-based system proposed in El-Beltagy and Rafea (2009), is used to extract keyphrases from a document as candidate terms. They also proposed a method uses linguistic and statistical technique, called CValue. The method uses a linguistic tool called Illinois POS tagger[9] and chunker[10] to extract noun phrases that may contain multiple adjectives and prepositions as candidate terms; then every noun phrase is given a CValue score based on its frequency. All candidate terms that are extracted by the previous

[7] Apache OpenNLP library. https://opennlp.apache.org/docs/1.8.2/manual/opennlp.html

[8] https://tartarus.org/martin/PorterStemmer/

[9] http://cogcomp.cs.illinois.edu/page/software_view/POS

[10] http://cogcomp.cs.illinois.edu/page/software_view/Chunker

techniques are combined together into one list. Then, based on the combination of stopwords proposed in Salton (1971) and (Fox, 1990), they filter out all stopwords from the list of the candidate terms. The remaining candidates are then mapped to Wikipedia articles to measure their domain-relatedness score. Finally, terms that have a CValue score and domain-relatedness value greater than a specific threshold are considered the final key terms.

Most of the existing work look at stopwords negatively and remove them before identifying keywords, however, they have a positive role in our work to extract domain-related key terms. Some existing work are based on statistical techniques, where they depend on keyword frequencies and a given threshold frequency to select domain-related keywords among candidates. However, specifying a threshold is subjective; moreover, statistical techniques are not effective when extracting terms from a single document. Other methods rely on external sources as knowledge bases to filter domain-related keywords from the extracted terms. This approach has a limitation of not selecting a domain-related keyword from candidates if it does not have an entry in the knowledge base.

Our work is somehow related to approaches that use NLP techniques to extract key terms. However, our work uniquely employs chunks and stopwords to extract not only keywords but also domain-related ones from a single document, without the need to use statistical techniques or external sources as knowledge bases in order to measure the domain-relatedness of the extracted terms. Unlike most existing work that focuses on text documents, our work focuses on text documents in the PDF format, especially textbooks, and provides solutions to work around problems with the text extracted from the PDF format.

3 The proposed model

The proposed model focuses on extracting glossary terms from text, especially text extracted from textbooks in the PDF format. The main idea that this model revolves around is to find out sentences that define the glossary terms. The proposed model refers to these sentences as *defining sentences*. Therefore, it is better to first define a *sentence* and *term*, and then describe the proposed method to extract terms from sentences.

The proposed model defines a *sentence* as the main unit of the text of any textual material, including textbooks, for defining terms in a particular domain or giving more explanation about the defined terms. Simply, a *sentence* consists of a subject, verb, and object. Subjects and objects can be a word or group of words. The verb part consists of either a *main verb* or an *auxiliary verb* followed by a main verb. *Auxiliary verbs* can be the *verb to be*, *verb to do*, *verb to have*, or *modal verbs*. The *Verb to be* includes *am*, *is*, *are*, *was*, and *were*; the *verb to do* includes *does*, *do*, and *did*; the *verb to have* includes *have*, *has*, *had*; the *modal verbs* include *can*, *could*, *will*, *would*, *shall*, *should, may, might,* and *must.* In contrast, the *main verbs* can be any verb that connects the subject with the object in a sentence.

As shown in Fig. 1, the proposed model classifies sentences in text into: (1) *is-a sentences*, denoted by *IS*, which are sentences having the *verb to be* as a main

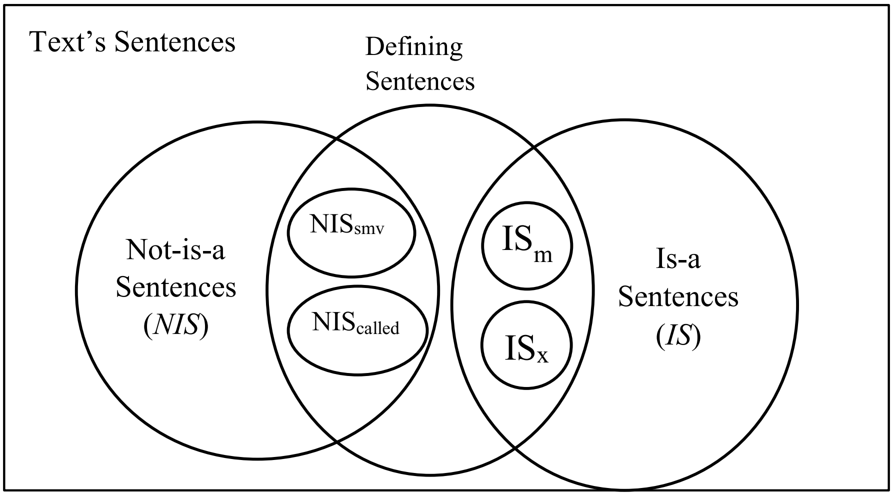

Fig. 1 Defining sentences classification

or auxiliary verb, (2) *not-is-a sentences*, denoted by *NIS*, which are sentences that do not have the *verb to be* as a main or auxiliary verb and are used to give more explanation about the defined key terms. However, the *defining sentences* are sentences that are derived from *is-a sentences* and *not-is-a sentences* under some conditions. The proposed model classifies the *defining sentences* to four main types: (1) IS_m sentences, which are *is-a sentences* that have the *verb to be* as a main verb, (2) IS_x sentences, which are *is-a sentences* that have the *verb to be* as an auxiliary verb followed by a selected main verb, (3) NIS_{smv} sentences, which are *not-is-a sentences* that have a selected main verb other than the *verb to be*, and (4) NIS_{called} sentences, which are *not-is-a sentences* having the verb *called, known as, referred to as, termed, or termed as*.

Moreover, the proposed model defines a *term* as a *word* or *group of words* in a *sentence*, and it can also be called a *key term*, *glossary term*, *keyword,* or *key-phrase* that represents a *concept*, *entity*, *object*, or *process* in a domain. In addition, the proposed model classifies terms on the basis of the forms in which a *term* can come within a *sentence* into two main types: (1) *regular terms* and (2) *irregular terms*. The *regular term* is denoted by *R* and defined as a *term* that consists of a single noun, group of nouns, or group of nouns and adjectives. The *irregular term* is denoted by *IR* and defined as a *term* that consists of not only group of nouns and adjectives but also other part of speech, including determiners, adverbs, prepositions, and conjunctions.

In light of the previous definitions and classifications of a *term* and *sentence*, The proposed model, as shown in Fig. 2, classifies the IS_m sentences to four types: (1) IS_mR sentences, which are IS_m sentences that have a *regular term* as a subject, (2) IS_mR_{adv} sentences, which are IS_m sentences that have a *regular term* as a subject with a preceding adverb sentence, (3) IS_mR_e sentences, which are IS_m sentences that have a *regular term* as a subject with extra information between

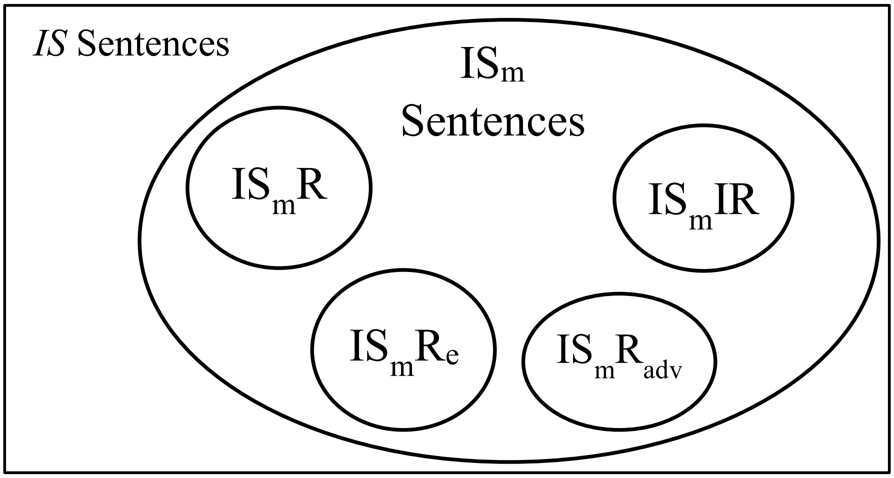

Fig. 2 Types of IS_m sentences

commas before the main verb, and (4) IS_mIR sentences, which are IS_m sentences that have an *irregular term* as a subject.

If text's sentences are denoted by *S*, the *defining sentences* by *D,* and terms by *T*, the proposed model can be formally described as follows:

$$S = IS \cup NIS \tag{1}$$

$$\left(IS_m \cup IS_x\right) \subset IS \tag{2}$$

$$IS_m \cap IS_x = \varnothing \tag{3}$$

$$\left(NIS_{called} \cup NIS_{smv}\right) \subset NIS \tag{4}$$

$$NIS_{called} \cap NIS_{smv} = \varnothing \tag{5}$$

$$D = \left(IS_m \cup IS_x \cup NIS_{called} \cup NIS_{smv}\right) \tag{6}$$

$$T = R \cup IR \tag{7}$$

$$IS_m = \left(IS_mR \cup IS_mR_e \cup IS_mR_{adv} \cup IS_mIR\right) \tag{8}$$

$$IS_mR \cap IS_mR_e \cap IS_mR_{adv} \cap IS_mIR = \varnothing \tag{9}$$

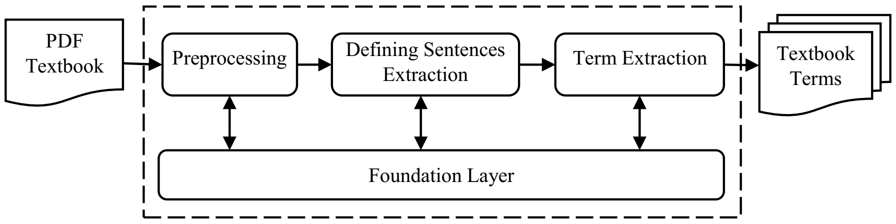

Fig. 3 The main phases and the foundation layer of the model

4 Model architecture and workflow

The model architecture, as shown in Fig. 3, takes a textbook in a PDF format as input and brings out the textbook key terms as output. The textbook passes through three phases into the model until the key terms are extracted. The three phases are (1) preprocessing, (2) *defining sentences* extraction, and (3) *term* extraction. The first phase deals with the PDF textbook where the text is extracted from the PDF format and prepared for the next phases. The other two phases deal with the text extracted from the PDF textbook in order to extract the defining sentences and the key terms it includes. The foundation layer of the model is not a phase, but it shows the tools and techniques that all processes in the three phases of the model rely on in order to achieve the ultimate goal of textbook key terms extraction.

More details are shown in Fig. 4 about the processes involved in every phase and the basic techniques used in the foundation layer. As shown in Fig. 4, the model has 21 processes that depends on four basic techniques. In the following sections, techniques in the foundation layer are described, and each process is discussed with the proposed algorithm that it builds on.

5 Foundation layer

As shown in Fig. 4, the foundation layer consists of a method and four NLP (Natural Language Processing) techniques that all processes of the model are built on:

(1) **PDF text extraction**. A method that is used in only one process in the model. It receives a PDF textbook and extracts the text from its PDF format that all the other NLP techniques will work on.
(2) **Sentence tokenization.** An NLP technique used to break down the extracted text into meaningful sentences.
(3) **Part of speech (POS) tagging.** An NLP technique used to identify the role of each word in every sentence, whether it acts as a noun, verb, adverb, adjective, determiner, conjunction, etc.

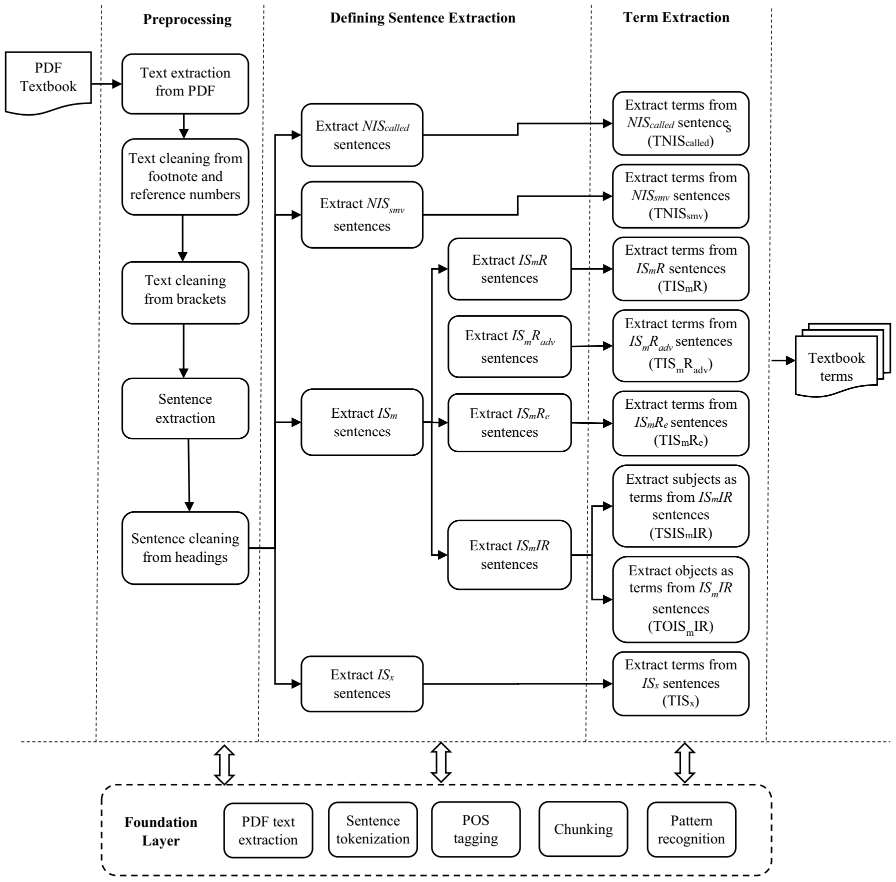
Preprocessing
Defining Sentence Extraction
Term Extraction
PDF Textbook
Text extraction from PDF
Text cleaning from footnote and reference numbers
Text cleaning from brackets
Sentence extraction
Sentence cleaning from headings
Extract NIS_{called} sentences
Extract NIS_{smv} sentences
Extract IS_m sentences
Extract IS_x sentences
Extract IS_mR sentences
Extract IS_mR_{adv} sentences
Extract IS_mR_e sentences
Extract IS_mIR sentences
Extract terms from NIS_{called} sentences (TNIS$_{called}$)
Extract terms from NIS_{smv} sentences (TNIS$_{smv}$)
Extract terms from IS_mR sentences (TIS$_m$R)
Extract terms from IS_mR_{adv} sentences (TIS$_m$R$_{adv}$)
Extract terms from IS_mR_e sentences (TIS$_m$R$_e$)
Extract subjects as terms from IS_mIR sentences (TSIS$_m$IR)
Extract objects as terms from IS_mIR sentences (TOIS$_m$IR)
Extract terms from IS_x sentences (TIS$_x$)
Textbook terms
Foundation Layer
PDF text extraction
Sentence tokenization
POS tagging
Chunking
Pattern recognition

Fig. 4 Architecture and workflow of the proposed model

(4) **Chunking.** An NLP technique used to work on the result of the POS tagging in order to identify certain phrases.
(5) **Pattern recognition.** An NLP technique used to find out specific patterns of characters directly from the extracted text without depending on the result of the sentence tokenization or POS tagging.

6 Preprocessing phase

The preprocessing phase is the first phase of the proposed model. This phase contains only one process to extract text from the PDF format, and the other processes in that phase focus on preparing the extracted text in a way that enables the other phases' processes to perform their tasks effectively. The preprocessing phase consists of five processes: (1) text extraction from a PDF textbook, (2) text cleaning from footnote and reference numbers, (3) text cleaning from brackets, and (4) sentence extraction, and (5) sentence cleaning from headings. The five processes are discussed in the following sections.

6.1 Text extraction from a PDF textbook

The purpose of this process is to receive a PDF textbook and extract the text from its PDF format. This process relies on the PDF text extraction method defined in the foundation layer in order to extract the text from a PDF textbook. However, the text extractor leaves some challenges that must be tackled in order to extract meaningful and precise key terms. Every process, in the following sections, addresses one or more of these challenges that influence the process's performance. In addition, every process provides a solution to work around these challenges within its proposed algorithm.

6.2 Text cleaning from footnote and reference numbers

Another challenge that hinders extracting key terms from the text extracted from PDF textbooks is how the text extractor extracts footnote and reference numbers from PDF textbooks. Textbooks can use footnote numbers with some terms to give more information about them in the footer of pages. Other textbooks can use a reference number at the end of a sentence to cite a reference included in a reference list at the end of a chapter. Both footnote and reference numbers form a challenge to extract accurate terms from the text.

Footnote and reference numbers appear in the extracted text from the PDF format as normal numbers, not in a superscript format, and that raises an issue. For footnote numbers, they seem to be a part of a key term, and thus they are extracted with them. Consequently, footnote numbers hinder extracting accurate key terms. For reference numbers at the end of sentences, the text extractor considers the period ending the sentence and the reference number at the end of the sentence together as a word, and thus it joins the sentence with its next sentence. Therefore, reference numbers mislead the sentence extractor to accurately separate sentences.

This process is proposed to tackle this challenge in order to extract terms and sentences effectively and accurately from the text extracted from the PDF format. It cleans the text from footnote and reference numbers by recognizing all patterns representing them in the text. The following is the regular expression that is proposed in order to identify different patterns of footnote and reference numbers in text. Then the process removes numbers from the result of this regular expression.

```
"\w+\.\d+|\w+\.\s\d+|\w+\s\d+|\w+\d+"
```

6.3 Text cleaning from brackets

Sentences that mention a term with its abbreviation between brackets, and the abbreviation is not at the end but within the term's words raise another challenge of extracting accurate terms. For example (Stair & Reynolds, 2012), the term

Business-to-Business (B2B) E-Commerce is mentioned in a sentence with its abbreviation, and the abbreviation is within the words of the term between brackets. In this example, the term is not *Business-to-Business*, but the term that should be extracted is *Business-to-Business E-Commerce.* Consequently, brackets can hinder extracting terms accurately. Subsequently, this process is proposed in order to clean the text from brackets as a preparation for extracting accurate terms.

6.4 Sentence extraction

This process receives the text extracted from the PDF textbook after it is cleaned from footnote and reference numbers, and after it is cleaned from brackets (see Sections 6.2 and 6.3). Then the process extracts all sentences from the cleaned text. The process relies on NLTK sentence extractor that extracts sentences based on the period at the end of every sentence.

6.5 Sentence cleaning from headings

This process receives all sentences extracted from the text as input, which are the output of sentence extraction process discussed in Section 6.4. It is noticed that headings and subheadings are included in the extracted sentences. These headings and subheadings hinder extracting key terms from the extracted sentences in a precise way, as they appear in the beginning of the first sentence following them. For example, if the heading is *INFORMATION CONCEPTS*, in uppercase, and the first sentence after the heading is *Information is the important concept in information technology.* Although the heading and the first sentence are extracted in separate lines, the sentence extractor extracts them together and consider them in one sentence as follows (Stair & Reynolds, 2012):

> *INFORMATION CONCEPTS Information is the important concept in information technology.*

As a result of merging a heading with the first sentence that follows it, the extracted term from the first sentence, in this example, will be *INFORMATION CONCEPTS Information* instead of *Information.* Therefore, headings and subheadings must be removed from the text to extract terms accurately. A challenge found in the extracted text from the PDF format is about missing any information about the format of words, including headings and subheadings, except that the text extractor leaves words that are in uppercase or capitalized as they are and splits them into separate lines. As shown in Algorithm 1 that is proposed for this process, these two features of headings, the uppercase format and splitting in a separate line, are employed in order to extract headings and subheadings from the text.

Another issue that misleads extracting headings and subheadings is the sentence-special-format words. A sentence-special-format word is a word that appears in a sentence capitalized or in uppercase format, similar to the format of the headings and subheadings. Similarly, the text extractor splits these

Algorithm 1 Extract headings and subheadings from text

```
Input: PDF textbook
Output: headings_list //list of headings and subheadings
01: text = extract text from the PDF textbook
02: lines = Split text into lines
03:    noun_tags = {NN,NNS,NNP,NNPS}
04:    verb_tags = {VB,VBD,VBP,VBZ}
05: For each line in lines do
06:    line_words = Split words in line by space
07:    tagged_words = POStag(line_words)
08:    isHeading = True
09:    For each (word,tag) in tagged_words do
10:       If tag ∈ noun_tags Then
11:          If isLowerCase(word) or isPunctuationMark(word) Then
12:             isHeading = False
13:             Exit For
14:    End For
       //exclude sentence-special-format words
15:    If isHeading == True Then
16:      next_line = GetNextLine(line, lines)
17:      next_line_words = Split words in next_line by space
18:      next_line_tagged_words = POStag(next_line_words)
19:      First_tag = GetFirstTag(next_line_tagged_words)
20:      If First_tag ∈ verb_tags or punctuation_marks Then
21:         isHeading = False
22:    If isHeading == True Then
23:         headings_list.add(line)
24: End For
25: Return headings_list
```

sentence-special-format words in separate lines. Consequently, headings, subheadings, and these sentence-special-format words are very similar without being able to distinguish between them.

This challenge is also addressed in Algorithm 1 in order to only extract headings and subheadings. The algorithm splits the text into lines and if it founds all words in a line capitalized or in uppercase, it checks the next line; and if the next line starts with a verb or punctuation mark such as a period, it decides that the words in the line don't represent a heading or subheading.

7 Defining sentence extraction phase

The defining sentences phase focuses on extracting the *defining sentences*, explained earlier in Section 3. Therefore, it consists of 8 processes: (1) Extract IS_m Sentences, (2) Extract IS_mR Sentences, (3) Extract IS_mR_{adv} Sentences, (4) Extract IS_mR_e Sentences, (5) Extract IS_mIR Sentences, (6) Extract IS_x Sentences, (7) Extract NIS_{called} Sentences, and (8) Extract NIS_{smv} Sentences. In the following sections, each of these processes is discussed with examples and the algorithm that is built on is also given.

Algorithm 2 Extract *is-a sentence* from text and IS_m sentences from *is-a sentences*

```
Input: text // the text extracted from a PDF textbook
Output: $IS_m$_list
    # Step 1: Extract all sentences from text
    all_sents = Extract_Sents(text)
    # Step 2: Extract all sentences having the verb to be
    isa_regex_pattern = '\w+\(?\w*\)?\w*,?(\w+\s)*,?\s+(is|are|was|were)'
    all_isa_sents = Extract_isa_Sents(all_sents, isa_regex_pattern)
    # Step 3: Extract sentences having verb to be as a main verb
    $IS_m$_list = []
    verb_to_be = {is, are, was, were}
    is_be_main_verb = False
    For each sent in all_isa_sents do
       tagged_sent = POStag(sent)
       if tagged_sent has a tag ∈{DT,CD,RB,JJ,
          verb_to_be Then
          is_be_main_verb = True
       elseIf tagged_sent has the tag JJ after verb_to_be Then
          if the tag after the tag JJ ∈ {NN, NNS, NNP, NNPS} Then
             is_be_main_verb = True
       elseIf tagged_sent has the tag RB after verb_to_be Then
          if the tag after the tag RB ∈{DT, CD,VBN, NN, NNS, NNP, NNPS} Then
             is_be_main_verb = True
       if is_be_main_verb == True Then
          $IS_m$_list.add(sent)
          is_be_main_verb = False
    End For
    Retrun $IS_m$_list
```

7.1 Extract IS_m sentences

This process receives all sentences from the process discussed in Section 6.5. Then it extracts IS_m sentences that are described in Section 3 as one of *defining sentences*. The following are examples (Stair & Reynolds, 2012) that show how IS_m sentences define key terms:

- *Input* <u>*is the*</u> *activity of gathering and capturing raw data* defines the key term *input*.
- *Workstations* <u>*are more*</u> *powerful than personal computers* defines the key term *Workstations*.
- *A byte* <u>*is typically*</u> *eight bits* defines the key term *byte*.
- *An enterprise system* <u>*is central*</u> *to an organization* defines the key term *enterprise system*.

This process is built on Algorithm 2 that proposes a regular expression pattern to recognize *is-a sentences* and extracts them from all text's sentences that are extracted by the process discussed in Section 6.5.Then it shows how IS_m sentences are extracted from the *is-a sentences*.

Algorithm 3 Extract IS_mR sentences from IS_m sentences

```
Input: ISm_sents //sentences having verb to be as a main verb
Output: ISmR_sents
01: ISmR_chunk_Pattern =
<DT>?<RB>*<VBN>*<JJ.?>*<NN.?.?>*<JJ.?>*<:>?(<VBG>*<NN.?.?>*<VBG>*)+<RB>*<VBZ|VBP|VB
D>
02: ISmR_sents = []
03: For each sent in ISm_sents do
04:    tagged_sent = POStag(sent)
05:    if tagged_sent has ISmR_chunk_pattern Then
06:       ISmR_sents.add(sent)
07: End For
08: Return ISmR_sents
```

7.2 Extract IS_mR sentences

This process focuses on extracting IS_mR sentences, which are IS_m sentences having *regular terms*. It also does not focus on identifying *regular terms* in any position in a sentence. Instead, as it is found that the traditional way to define a key term is to use it as a subject at the beginning of a sentence, this process focuses only on *regular terms* that come in the subject position of IS_m sentence. The process is built on Algorithm 3 that proposes the following chunk pattern to identify *regular terms* in the subject position.

`<DT>?<RB>*<VBN>*<JJ.?>*<NN.?.?>*<JJ.?>*<:>?(<VBG>*<NN.?.?>*<VBG>*)+<RB>*<VBZ|VBP|VBD>`

As shown in this chunk pattern, the sentence is extracted if it starts with a term that consists of a noun(s) (NN, NNS, NNP, or NNPS tags) and may also be preceded by a determiner (DT tag), an adverb (RB tag), a verb in past participle (VBN tag), and/or an adjective (JJ tag). The tag(< : >) is used to include multiple hyphenated adjectives. The pattern also makes sure that the *regular term* is a subject when it is followed by a verb in the 3rd person singular present (VBZ tag), a verb in the non-3rd person singular present (VBP tag), or a verb in the past (VBD tag). The following sentences give an example (Stair & Reynolds, 2012) of *regular terms* that come as a subject:

- *Knowledge workers are people who create, use, and disseminate knowledge.* The term in this sentence is *Knowledge workers*.
- *Highly structured problems are straightforward, requiring known facts and relationships*. The term in this sentence is *Highly structured problems*.

7.3 Extract IS_mR_{adv} sentences

This process focuses on extracting IS_mR_{adv} sentences, which are sentences having a *regular term* in the subject position, but there is an adverb sentence preceding it. The following gives an example (Stair & Reynolds, 2012) of IS_mR_{adv} sentences:

Algorithm 4 Extract IS_mR_{adv} sentences from IS_m sentences

```
Input : ISm_sents //is-a sentences having verb to be as a main verb
Output:  ISmRadv_sents
01: ISmRadv_chunk_pattern =
^<^(DT)|RB|RBR|VBN|IN|JJ><.+>*<,><DT>?<RB>*<VBN>*<JJ.?>*<:>?<NN.?.?>+<VBZ|VBP|VBD>
02: ISmRadv_sents = []
03: For each sent in ISm_sents do
04:     tagged_sent = POStag(sent)
05:     if tagged_sent has ISmRadv_chunk_pattern Then
06:         ISmRadv_sents.add(sent)
07: End For
08: Return  ISmRadv_sents
```

- *Originally, a hacker was a person who enjoyed computer technology.* The term is *hacker.*
- *In the context of systems development, stakeholders are people who benefit from the system.* The term is ***stakeholders.***

The process is built on Algorithm 4 that proposes the following chunk pattern to recognize IS_mR_{adv} sentences.

```
^<^(DT)|RB|RBR|VBN|IN|JJ><.+>*<,><DT>?<RB>*<VBN>*<JJ.?
>*<:>?<NN.?.?>+<VBZ|VBP|VBD>
```

As it is shown in the pattern, it looks like the pattern that recognizes IS_mR sentences except that it has a pattern in the beginning to recognize adverb sentences as well.

7.4 Extract IS_mR_e sentences

Although *regular terms* usually come as subjects followed by the *verb to be* as a main verb, they can also be followed by extra information mentioned between two commas before the main verb. The following is an example (Stair & Reynolds, 2012) of IS_mR_e sentences:

Algorithm 5 Extract IS_mR_e sentences from IS_m sentences

```
Input: ISm_sents //is-a sentences having verb to be as a main verb
Output: ISmRe_list
01: ISmRe_chunk_pattern =
        ^<DT>?<RB>*<JJ.?>*<:>?<NN.?.?>+<,><.+>+<,><VBZ|VBP|VBD>
02: ISmRe_list = []
03: For each sent in ISm_sents do
04:     tagged_sent = POStag(sent)
05:     if tagged_sent has ISmRe_Chunk_pattern Then
06:         ISmRe_list.add(sent)
07: End For
08: Return ISmRe_list
```

Algorithm 6 Extract IS_mIR sentences from IS_m sentences

```
Input: ISm_sents //is-a sentences having verb to be as a main verb
Output: ISmIR_list
01: ISmIR_chunk_pattern =
^<DT>*<VBG|VBN|JJ.?|RB|NN.?.?>*<IN><DT>*<VBG|VBN|JJ.?|RB|NN.?.?>+<VBZ|VBP|VBD>
02: ISmIR_list = []
03: For each sent in ISm do
04:    tagged_sent = POStag(sent)
05:    if tagged_sent has ISmIR_Chunk Then
06:       ISmIR_list.add(sent)
07: End For
08: Return ISmIR_list
```

A CD burner, the informal name for a CD recorder, is a device…. The term is *CD burner.*

This process focuses on extracting these sentences from IS_m sentences. It is built on Algorithm 5 that proposes the following chunck pattern to extract IS_mR_e sentences from IS_m sentences:

```
^<DT>?<RB>*<JJ.?>*<:>?<NN.?.?>+<,><.+>+<,><VBZ|VBP|VBD>
```

This pattern is similar to the one that recognizes IS_mR sentences except that it adds a pattern for recognizing the extra information coming between commas after the subject and before the verb.

7.5 Extract IS_mIR sentences

This process focuses on extracting IS_mIR sentences from IS_m sentences. IS_mIR sentences are sentences that have *irregular terms* in the subject position and the *verb to be* as a main verb. The following sentence (Stair & Reynolds, 2012) gives an example of an IS_mIR sentence:

Redundant array of independent/inexpensive disks (RAID) is a method of storing data.

The *irregular term* of this sentence is *Redundant array of independent/inexpensive disks (RAID).*

This process is built on Algorithm 6 that proposes the following chunk pattern to recognize IS_mIR sentences:

```
^<DT>*<VBG|VBN|JJ.?|RB|NN.?.?>*<IN><DT>*<VBG|VBN|JJ.?|
RB|NN.?.?>+<VBZ|VBP|VBD>
```

7.6 Extract IS_x sentences

Besides IS_m sentences, there are other *defining sentences* (IS_x) that have the *verb to be* as an auxiliary verb but followed by one of selected main verbs. The selected

Algorithm 7 Extract IS_x sentences from *is-a sentences*

```
Input: isa_sents // all sentences having verb to be
Output: IS_x_List
01: IS_x_regex_pattern =
  '.*\s+(is|are|was|were|can be)\s*\w*\s*(called|defined as|referred to as|known
    as|termed|termed as)'
02: IS_x_List = []
03: For each sent in all_isa_sents do
04:     if sent has IS_x_regex_pattern then
05:         IS_x_List.add(sent)
06: End For
07: Returen IS_x_List
```

main verbs are *called, termed, termed as, referred to as*, *defined as*, and *known as*. The following sentence (Stair & Reynolds, 2012) gives an example of IS_x sentence:

Processing that uses several processing units is called multiprocessing

This sentence defines the key term *multiprocessing*, however, it has the *verb to be as a*n auxiliary verb but with the verb *called*, one of the selected main verbs. Algorithm 7 shows the proposed regular expression pattern to recognize these sentences and how to extract them from *is-a* sentences.

7.7 Extract NIS_{called} Sentences

The *not-is-a sentences*, as defined earlier in Section 3, can be considered *defining sentences* when they satisfy some conditions. This section sets a condition that turns a *not-is-a sentence* into a *defining sentence*, which is having the verb *called, known as, referred to as, termed, or termed as* among its words.

The following are examples (Stair & Reynolds, 2012) of sentences that are considered *not-is-a sentences* and *defining sentences* as well. They also show different forms of having the verb *called* in *not-is-a sentences*.

- *This concept, called cloud computing, allows people to get the information they need from the Internet.* This sentence defines the term *cloud computing*.
- *An executive support system, also called an executive information system, helps top-level managers.* This sentence defines the term *executive information system.*
- *A small company called Microsoft developed PC-DOS and MS-DOS to support the IBM personal computer introduced in the 1980s*. This sentence defines the term *Microsoft*.
- *A group support system includes the DSS elements just described as well as software, called groupware, to help groups make effective decisions*. This sentence defines the term *groupware.*
- *A company can either develop a one-of-a-kind program for a specific application (called proprietary software*). This sentence defines the term *proprietary software*.

Algorithm 8 Extract NIS_{called} sentences from all sentences in the text

```
Input: all_sents // all sentences in the text
Output: $NIS_{called}$_List
01: called_regex_pattern =
     ' \w*\(?\w*\)?\w*,?(\w*\s)*,?\s*((C|c)alled|(K|k)nown as|(R|r)eferred to
        as|termed|(T|t)ermed as)'
02: is_called_regex_pattern =
    '\w*\(?\w*\)?\w*,?(\w+\s)*,?\s+(is|are|was|were)\s*\w*(called|known
        as|referred to as|termed|termed as)'
03: $NIS_{Called}$_List = []
04: For each sent in all_sents do
05:    if sent has called_regex_pattern then
06:       if sent does not have is_called_regex_pattern then
07:          $NIS_{called}$_List.add(sent)
08: End For
09: Return $NIS_{called}$_List
```

Algorithm 9 Extract NIS_{smv} sentences from all sentences in the text

```
Input: all_sents // all sentences in the text
Output: $NIS_{SMV}$_List
01: $NIS_{SMV}$_regex_pattern =
   '\w*\(?\w*\)?\w*,?(\w+\s)*,?\s+(involves?| refers? to| specify|specifies|
     includes?|combines?|defines?|contains?| consists? of| house?| means?)'
02: $NIS_{SMV}$_List = []
03: For each sent in all_sents do
04:    if sent has $NIS_{SMV}$_regex_pattern then
05:       $NIS_{SMV}$_List.add(sent)
06: End For
07: Return $NIS_{SMV}$_List
```

The following algorithm, Algorithm 8, is proposed to extract the NIS_{called} sentences from all sentences in a text, regardless of the form in which the verb *called* comes. As there is another algorithm, Algorithm 7, that extracts IS_x sentences that also can have the verb *called* as a main verb followed by the auxiliary *verb to be*, this algorithm focuses only on extracting NIS_{called} sentences that do not have the verb *called* as a main verb after the *verb to be*.

7.8 Extract NIS$_{smv}$ sentences

The NIS_{smv} sentences define another condition that can transform a *not-is-a* sentence into a *defining sentence*. The condition is that it must have one of these selected verbs `involve, refer to, specify, include, combine, define, contain, consist of, house, mean` to become a *defining sentence*. Algorithm 9 shows the regular expression pattern used to select the NIS_{smv} sentences from all sentences in the text.

8 Term extraction phase

This phase focuses on extracting *terms* that come either in the form of *regular terms* or *irregular terms*. These *terms* are only extracted from the *defining sentences* that are extracted in the previous phase. This phase consists of 8 processes: (1) Extract TIS_mR terms from IS_mR Sentences, (2) Extract TIS_mR_{adv} terms from IS_mR_{adv} sentences, (3) Extract terms TIS_mR_e from IS_mR_e sentences, (4) Extract $TSIS_mIR$ terms from IS_mIR sentences, (5) Extract $TOIS_mIR$ terms from IS_mIR sentences, (6) Extract TIS_x terms from IS_x sentences, (7) Extract $TNIS_{called}$ terms from NIS_{called} sentences, and (8) Extract $TNIS_{smv}$ terms from NIS_{smv} sentences. Each process focuses on extracting terms from a particular type of the *defining sentences*. In the following sections, each process is discussed with examples and with the algorithms that it builds on.

8.1 Extract TIS_mR terms from IS_mR sentences

The TIS_mR terms are *regular terms* in the subject position of the IS_mR sentences that is discussed in Section 7.2. every IS_mR sentence is extracted based on the following chunk:

```
<DT>?<RB>*<VBN>*<JJ.?>*<NN.?.?>*<JJ.?>*<:>?(<VBG>*<NN.
?.?>*<VBG>*)+<RB>*<VBZ|VBP|VBD>
```

Consequently, every *regular term* in these sentences must be extracted without any determiner. The following sentence gives an example (Stair & Reynolds, 2012) of this case:

- *An information system is a set of interrelated components that collect, manipulate, store, and disseminate data and information.* The term in this sentence is *information system* without the article *an*.

In addition, the process extracts *regular terms* that can be a noun(s) preceded by two or three words hyphenated as an adjective. The following sentence gives an example (Stair & Reynolds, 2012) of the two words hyphenated as an adjective:

- *A computer-based information system is a single set of hardware.* The extracted term in this sentence is *computer-based information system.*

Another sentence that gives an example (Stair & Reynolds, 2012) of three hyphenated words as an adjective is:

- *Consumer-to-consumer e-commerce is a subset of e-commerce.* The extracted term in this sentence is *Consumer-to-consumer e-commerce.*

Algorithm 10 Extract *regular terms* TIS_mR from IS_mR sentences

```
Input: IS_mR_sents
Output: TIS_mR_list
01: TIS_mR_chunk_pattern = '^<DT>?<RB>*<VBN>*<JJ>*<:>?<NN.?>+<VBZ|VBP|VBD>'
02: For each sent in IS_mR_sents do
03:    tagged_sent = POStag(sent)
04:    TIS_mR_chunk = Extract_chunk (tagged_sent, TIS_mR_chunk_pattern)
05:    subject = []
06:    For each (word,tag) in TIS_mR_chunk do
07:       if tag ∉ {DT,RB,VBZ,VBP,VBD} Then
08:          subject.add(word)
09:    End For
10:    TIS_mR = ' '.join(subject)
11:    TIS_mR_list.add(TIS_mR)
12: End For
13: Return TIS_mR_list
```

The following chunk pattern is proposed to recognize a *regular term* in the subject position with three hyphenated adjectives:

```
^<DT>?<JJ>*<:>?<NN.?>+<VBZ|VBP|VBD>
```

The tag < : > is used in the pattern to work around a problem of wrong tagging three hyphenated adjectives, where the NLTK POS tagger tags the first two hyphenated adjective together as < JJ >, the second hyphen as < : >, and the third adjective as < NN >.

Moreover, the process can extract *regular terms* that contain an apostrophe. The following sentence (Stair & Reynolds, 2012) gives an example of this case:

- *Nvidia's GeForce D is software that can display images on a computer screen.* The extracted term in this sentence is *Nvidia's GeForce D*.

Sometimes, in PDF textbooks, there is not enough space in a line to have a whole word written completely. Therefore, a word is divided into two parts, the first part is written to complete the line and the second part is written at the beginning of the next line. A hyphen is written after the first part to indicate that the remaining part of the word is written at the beginning of the next line. Unfortunately, the PDF text extractor leaves words with end-of-line hyphens not merged, which represents a real challenge of extracting terms accurately. However, this process also focuses on facing this challenge by combining together the whole word representing a key term.

The following is an example (Stair & Reynolds, 2012) of a term with the end-of-line hyphenation:

bandwidth, the more information can be exchanged at one time. Broadband communications is a relative term but generally means a telecommunications system that can

Algorithm 11 Extract *regular terms* TIS_mR_{adv} from IS_mR_{adv} sentences

```
Input  : $IS_mR_{adv}$_sents
Output:  $TIS_mR_{adv}$_list
adv_chunk_pattern = '^<^(DT)|RB|RBR|VBN|IN|JJ><.+>*<,>'
$TIS_mR$_chunk_pattern = '<DT>?<RB>*<VBN>*<JJ.?>*<:>?<NN.?.?>+<VBZ|VBP|VBD>'
For each sent in $IS_mR_{adv}$_sents do
    tagged_sent = POStag(sent)
    adv_chunk_text = Extract_text (tagged_sent, adv_chunk_pattern)
    sent_without_adv = sent.remove(adv_chunk_text)
    tagged_sent = POStag(sent_without_adv)
    $TIS_mR$_chunk = Extract_chunk (tagged_sent, $TIS_mR$_chunk_pattern)
    subject = []
    For each (word, tag) in $TIS_mR$_chunk do
        if tag $\notin$ {DT,RB,VBZ,VBP,VBD} Then
            subject. add(word)
    End For
    $TIS_mR_{adv}$ = ' '.join(subject)
    $TIS_mR_{adv}$_list.add($TIS_mR_{adv}$)
End For
Return $TIS_mR_{adv}$_list
```

As shown in the this example, the term that has the end-of-line hyphen is *Broadband communications*. The word *communication* is divided into two parts, *commu* and *nications*, because there is not enough space in the line to write the whole word. The first part is written and followed by a hyphen, *commu-,* and the second part, *nications*, is written at the beginning of the next line. The process extracts the whole term without a hyphen as *Broadband communications*. This process is built on the Algorithm 10 that proposes a chunk pattern to recognize *regular terms* in IS_mR sentences and also explains how they are extracted.

8.2 Extract TIS_mR_{adv} terms from IS_mR_{adv} sentences

The IS_mR_{adv} sentences, discussed in Section 7.3, are IS_m sentences that starts with an adverb sentence before the subject. The TIS_mR_{adv} terms are *regular terms* in the subject position of the IS_mR_{adv} sentences. The Algorithm 11 is proposed to enable this process to extract TIS_mR_{adv} terms from IS_mR_{adv} sentences.

8.3 Extract TIS_mR_e terms from IS_mR_e sentences

The IS_mR_e sentences, discussed in Section 7.4, have an extra information about the subject that is located between the subject and the main verb, *verb to be*, and separated by commas. The TIS_mR_e terms are subjects in the IS_mR_e sentences. The following is an example (Stair & Reynolds, 2012) of a sentence having TIS_mR_e term:

- *A CD burner, the informal name for a CD recorder, is a device.* The term is *CD burner.*

Algorithm 12 Extract *regular terms* TIS_mR_e from IS_mR_e sentences

```
Input: $IS_mR_e$_sents
Output: $TIS_mR_e$_list
01: $TIS_mR_e$_chunk_pattern = '^<DT>?<RB>*<JJ.?>*<:>?<NN.?.?>+<,>'
02: For each sent in $IS_mR_e$_sents do
03:    tagged_sent = POStag(sent)
04:    $TIS_mR_e$_chunk = Extract_chunk (tagged_sent, $TIS_mR_e$_chunk_pattern)
05:    subject = []
06:    For each (word,tag) in $TIS_mR$_chunk do
07:       if tag $\notin$ {<DT>,<RB>,<,>} Then
08:          subject.add(word)
09:    End For
10:    $TIS_mR_e$ = ' '.join(subject)
11:    $TIS_mR_e$_list.add($TIS_mR_e$)
12: End For
13: Return $TIS_mR_e$_list
```

Algorithm 13 Extract subjects as *irregular terms* $TSIS_mIR$ from IS_mIR sentences

```
Input: $IS_mIR$_sents
Output: $TSIS_mIR$_list
01: For each sent in $IS_mIR$_sents do
02:    tagged_sent = POStag(sent)
03:    subject = []
04:    For each (word,tag) in tagged_sent do
05:       if word $\in$ {is,are,was,were} Then
06:          Exit For
07:       else
08:          subject.add(word)
09:    End For
10:    $TSIS_mIR$ = ' '.join(subject)
11:    $TSIS_mIR$_list.add($TSIS_mIR$)
12: End For
13: Return $TSIS_mIR$_list
```

This process is built on the Algorithm 12 that is proposed to extract TIS_mR_e terms from IS_mR_e sentences.

8.4 Extract $TSIS_mIR$ terms from IS_mIR sentences

The $TSIS_mIR$ terms are *irregular terms* in the subject position of the IS_mIR sentences that are discussed in Section 7.5. The Algorithm 13 is proposed to enable this process to extract $TSIS_mIR$ terms from IS_mIR sentences.

8.5 Extract $TOIS_mIR$ terms from IS_mIR sentences

The $TOIS_mIR$ terms are *irregular terms* in the object position of the IS_mIR sentences that are discussed in Section 7.5. The following are examples (Stair & Reynolds, 2012) of IS_mIR sentences having $TOIS_mIR$ terms:

Algorithm 14 Extract objects as *regular terms $TOIS_mIR$* from *IS_mIR* sentences

```
Input: $IS_mIR$_sents
Output: $TOIS_mIR$_list
object_chunk_pattern =
   '<VBZ|VBP|VBD><DT>?<VBG|VBN|JJ.?|RB|NN.?.?>*<IN>?<CC>?<DT>?<VBG|VBN|JJ.?|RB
     |NN.?.?>+<,>?'
For each sent in $IS_mIR$_sents do
   tagged_sent = POStag(sent)
   if tagged_sent has object_chunk_pattern then
      object_chunk_text = Extract_text (tagged_sent,
                       object_chunk_pattern)
      tagged_chunk_text = POStag(object_chunk_text)
      subject = []
      For each (word,tag) in tagged_chunk_text do
         if tag ∉ {<VBZ>,<VBP>,<VBD>,<DT>,<,>} Then
            subject.add(word)
      End For
   $TOIS_mIR$ = ' '.join(subject)
   $TOIS_mIR$_list.add($TOIS_mIR$)
End For
Return $TOIS_mIR$_list
```

Algorithm 15 Extract *regular* or *irregular terms TIS_x* from *IS_x* sentences

```
Input: $IS_x$_sents
Output: $TIS_x$_list
For each sent in $IS_x$_sents do
   tagged_sent = POStag(sent)
   term_words = []
   For each (index,word,tag) in tagged_sent do
      if word ∈ {called,termed,defined,referred,known} then
         For next_index in range(index + 1,len(tagged_sent)) do
            if tag_of(next_index) ∈ {NN,NNS,NNP, NNPS,VBG,JJ} or
            word_of(next_index) = '-' then
               term_words.add(word_of (next_index))
            elseif word_of(next_index) ∉{a,an,the,to,as} then
               Exit For
         End For
   End For
   term = ' '.join(term_words)
   $TIS_x$_list.add(term)
End For
Return TISx_list
```

- *Another major activity of a TPS is* *<u>data manipulation</u>, the process of…..* The term is *data manipulation.*
- *An important part of an expert system is the <u>explanation facility</u>, which allows* …. The term is *explanation facility.*
- *Turning data into information is a <u>process</u>, or a set of logically related tasks.* The term is *process.*

Algorithm 16 Extract *regular* or *irregular terms* $TNIS_{called}$ from NIS_{called} sentences

```
Input: NIS_called_sents
Output: TNIS_called_list
01: For each sent in NIS_Called_sents do
02:    tagged_sent = POStag(sent)
03:    term_words = []
04:    For each (index,word,tag) in tagged_sent do
05:       if word ∈ {called,Called,known as,Known as, Referred to, referred
              to, termed as, Termed as} then
06:          For next_index in range(index + 1,len(tagged_sent)) do
07:             if tag_of(next_index) ∈ {NN,NNS,NNP, NNPS,VBG,JJ,CC,IN}
                Or word_of(next_index) = '-' then
08:                term_words.add(word_of(next_index))
09:             elseif word_of(next_index) ∉ {a,an,the} then
10:                Exit For
11:          End For
12:    End For
13:    term = ' '.join(term_words)
14:    TNIS_called_list.add(term)
15: End For
16: Return TNIS_called_list
```

Algorithm 17 Extract subjects as terms $TNIS_{smv}$ from NIS_{smv} sentences

```
Input: NIS_smv_sents
Output: TNIS_smv_list
01: NIS_smv_chunk_pattern = '^<.+>+<VBZ|VBP|VBD>'
02: For each sent in NIS_smv_sents do
03:    tagged_sent = POStag(sent)
04:    NIS_smv_chunk_text = Extract_text(tagged_sent, NIS_smv_chunk_pattern)
05:    tagged_chunk_text = POStag(NIS_smv_chunk_text)
06:    subject = []
07:    For each (word,tag) in tagged_chunk_text do
08:       if tag ∈ {VBZ,VBP, VBD} Then
09:          Exit For
10:       else
11:          subject.add(word)
12:    End For
13:    TNIS_smv = ' '.join(subject)
14:    TNIS_smv_list.add(TNIS_smv)
15: End For
16: Return TNIS_smv_list
```

The Algorithm 14 is proposed to enable this process to extract $TOIS_mIR$ terms from IS_mIR sentences.

8.6 Extract TIS_x terms from IS_x sentences

The IS_x sentences, discussed in Section 7.6, are IS_m sentences that have the *verb to be* as an auxiliary verb with one of the selected main verbs. The TIS_x terms are *regular or irregular* terms in the subject position of the IS_x sentences. The

Algorithm 15 is proposed to enable this process to extract TIS_x terms from IS_x sentences.

8.7 Extract TNIS$_{called}$ terms from NIS$_{called}$ sentences

The NIS_{called} sentences, discussed in Section 7.7, are *NIS* sentences having the verb *called* that is not a main verb after the auxiliary verb, *verb to be*. The $TNIS_{called}$ terms are *regular or irregular* terms that come after the verb *called* or one of *called* sisters mentioned in Section 7.7. The Algorithm 16 is proposed to enable this process to extract $TNIS_{called}$ terms from NIS_{called} sentences.

8.8 Extract TNIS$_{smv}$ terms from NIS$_{smv}$ sentences

The NIS_{smv} sentences, discussed in Section 7.8, are *NIS* sentences having one of the selected main verbs. The $TNIS_{smv}$ terms are *regular or irregular* terms that come after a selected main verb. The Algorithm 17 is proposed to enable this process to extract $TNIS_{smv}$ terms from NIS_{smv} sentences.

9 Experiments

This section explains the experimental evaluation of the proposed model to assess the effectiveness of the different processes and algorithms. The proposed model is implemented by using Textract[11] , which is a Python package for PDF text extraction, and Python's NLTK for sentence tokenization, POS tagging, chunking, and pattern recognition. Textract and NLTK are Python's libraries or tools used to implement methods and NLP techniques mentioned in the foundation layer (Section 5).

The proposed algorithms are evaluated through two experiments. In the first experiment, a PDF textbook is used from the business domain, but, in the second experiment, another one is used from the science domain. The extracted terms from every PDF textbook is compared to the textbook's key terms (glossary terms) included in its key terms section. The textbook key terms are used as a gold standard to evaluate the effectiveness of the proposed work. Our work provides a parameterless method to extract keywords, so that no parameters are needed to be given or tuned except for a PDF textbook that is given as an input. More details about the two experiments are given in the following sections.

9.1 Experiment 1

In this experiment, a free and openly licensed PDF textbook is used from the business domain, entitled the *Organization development* (OpenStax, 2019). The textbook consists of 704 pages which contain a cover page, title page, copyright

[11] https://textract.readthedocs.io/en/stable/python_package.html

pages, acknowledge page, table of contents, preface, 19 chapters, 2 appendixes, and index. In every chapter, there are figures, caption for each figure, tables, and header and footer on every page. At the end of each chapter there are key terms, summary, a case study, review questions, and exercises. The textbook has 19 lists of key terms containing 468 terms in total.

In order to evaluate the different processes of the proposed work, the 468 key terms of the textbook are classified into 15 categories with their weights. As shown in Table 1, these categories can be classified into two sections: (1) defined categories and (2) undefined categories. The defined categories are those introduced by the proposed model. The undefined categories are new categories that are not proposed by the model, but discovered through the experiments. The weight of each category is calculated as a ratio of the number of terms in the category to the total number of the key terms.

The first eight categories in Table 1 are for terms that are described in detail in the previous sections of the proposed work, but categories from number 9 to 15 are new categories that are described as follows:

Category (9). This category is about terms that are not defined using a sentence, but they are mentioned between brackets after their explanation. The following sentence (OpenStax, 2019) gives an example of these terms:

- *Successful employees know what they want to achieve (<u>direction</u>).* This sentence defines the term *direction*.

Category (10). This category is about terms that are defined indirectly within a sentence such as the following sentences (OpenStax, 2019):

- *This set of perceptions often leads to <u>abundance-based change</u>, in which leaders assume*. This sentence defines the term *abundance-based change***.**
- *He meant that <u>corporate culture</u> is more influential*. This sentence defines the term *corporate culture*.
- *Entrepreneurs can access <u>debt capital</u>, which is ….* This sentence defines the term *debt capital*.
- *Over time, these areas of specialization mature through <u>differentiation</u>, the process of organizing employees into groups*. This sentence defines the term *differentiation*.

Category (11). Terms came in the object but with not IS_mIR sentences. The following sentence (OpenStax, 2019) gives an example of these terms:

- *Another prevalent type is <u>family entrepreneurship</u>* that …. This sentence defines the term *family entrepreneurship*.
- *An appraisal system that has received increasing attention in recent years is the <u>behaviorally anchored rating scale</u>*. This sentence defines the term *behaviorally anchored rating scale*.

Category (12). This category is about terms that are defined by main verbs that are other than *verb to be* as a main verb and other than the selected main

Table 1 Categories of the textbook key terms used in Experiment 1

Category number	Term category	# Key Terms	Category Weight
Defined Categories:			
1	TIS_mR terms	137	29.3%
2	TIS_mR_{adv} terms	17	3.6%
3	TIS_mR_e terms	7	1.5%
4	TIS_mIR terms	3	0.6%
5	TIS_mOIR terms	4	0.9%
6	TIS_x terms	28	6%
7	$TNIS_{called}$ terms	18	3.8%
8	$TNIS_{smv}$ terms	37	7.9%
Undefined Categories:			
9	Terms not defined in a sentence but mentioned between brackets after explaining them	1	0.2%
10	Terms defined indirectly within a sentence	87	18.6%
11	Terms came in the object but with not IS_mIR sentences	4	0.9%
12	Terms with main verbs other than *verb to be* and other than the selected main verbs	70	15%
13	Terms with *verb to be* as an auxiliary verb but with not-selected main verbs	2	0.4%
14	Terms came in sentences starting with a conjunction	1	0.2%
15	Terms defined in Key Terms section only	52	11.1%
	Total Key Terms	**468**	
15	(-) Terms defined in Key Terms section only	52	
	Fair Total Key Terms (Gold Standard)	**416**	

verbs. Examples of these verbs are *enable, begin, allow, automate, connect, combine, handle, give, perform, require, support,* and much more.

Category (13). This category is about terms that come with *verb to be* as an auxiliary verb, but the main verb is not one of the selected main verbs. For example (OpenStax, 2019):

- *Expert power is demonstrated when person ….* This sentence defines the term *Expert power*.

Category (14). This category is about terms came in sentences starting with a conjunction. For example (OpenStax, 2019):

- *Whereas frustration is a reaction to an obstruction in instrumental activities or behavior, anxiety is a feeling of inability*. This sentence started with a conjunction whereas to combine two sentences defining two terms, *frustration* and *anxiety*.

Category (15). This category is about terms that are defined in the key terms section only and don't have a sentence defining them outside this section.

As shown in Table 1, terms of category 15 are excluded from the total key terms because they are not mentioned in the main text. Moreover, in the key terms section, these terms are not involved in a complete sentence. Therefore, the fair key terms that is used as a gold standard will become 416 instead of 468 terms after excluding terms of category 15.

9.2 Experiment 2

In this experiment, another free and openly licensed PDF textbook is used from the science domain, entitled the *University Physics Volume 1* (Ling et al., 2021). The textbook consists of 979 pages which contain a cover page, title page, copyright pages, acknowledge page, table of contents, preface, 17 chapters, 7 appendixes, answer key, and index. In every chapter, there are figures, caption for each figure, tables, equations, and header and footer on every page. At the end of each chapter, there are key terms, key equations, summary, conceptual questions, and problems. The textbook has 17 lists of key terms containing 314 terms in total.

As shown in Table 2, the 314 key terms of the textbook are classified, similar to experiment 1, using the 15 categories with their weights. The fair total key terms used as a gold standard for evaluation will become 304 terms after excluding terms in category 15.

10 Result and discussion

This section discusses the result of the two experiments. It also evaluates the proposed work by comparing the extracted terms with the PDF textbooks' key terms in each experiment. As preparation for this comparison, the textbooks' key terms

Table 2 Categories of the textbook key terms used in Experiment 2

Category number	Term category	# Key Terms	Category Weight
Defined Categories:			
1	TIS_mR terms	118	37.6%
2	TIS_mR_{adv} terms	14	4.5%
3	TIS_mR_e terms	2	0.6%
4	TIS_mIR terms	12	3.8%
5	TIS_mOIR terms	14	4.5%
6	TIS_x terms	52	16.6%
7	$TNIS_{called}$ terms	20	6.4%
8	$TNIS_{smv}$ terms	12	3.8%
Undefined Categories:			
9	Terms not defined in a sentence but mentioned between brackets after explaining them	0	0%
10	Terms defined indirectly within a sentence	33	10.5%
11	Terms came in the object but with not IS_mIR sentences	8	2.5%
12	Terms with main verbs other than *verb to be* and other than the selected main verbs	15	4.8%
13	Terms with *verb to be* as an auxiliary verb but with not-selected main verbs	4	1.3%
14	Terms came in sentences starting with a conjunction	0	0%
15	Terms defined in Key Terms section only	10	3.2%
	Total Key Terms	**314**	
15	(-) Terms defined in Key Terms section only	10	
	Fair Total Key Terms (Gold Standard)	**304**	

and the extracted terms are kept in a singular form and any duplicated terms are removed. Besides that, a key term is assumed to be extracted even if the extracted term is not exactly the same as the key term, i.e., containing one or more words over the words in the key term. There were a few number of extracted terms that followed that assumption. The performance of the proposed term extraction method is measured in terms of recall and precision as follows:

$$Category\ Recall = \frac{Number\ of\ terms\ extracted\ from\ a\ category}{Number\ of\ the\ textbook\ key\ terms\ in\ that\ category} \tag{10}$$

$$Recall = \frac{Number\ of\ terms\ extracted\ from\ all\ categories}{Number\ of\ the\ textbook\ key\ terms\ in\ all\ categories} \tag{11}$$

$$Precision = \frac{Number\ of\ terms\ extracted\ from\ all\ categories}{Number\ of\ all\ extracted\ terms\ in\ the\ result} \tag{12}$$

The result of term extraction in each experiment is shown in Table 3. The table shows category numbers, the number of key terms in each category, the number of extracted terms from each category, the recall percentage of each category. In addition, it shows the total key terms, the total number of the extracted terms from all categories, and the recall percentage of each experiment. In experiment 1, as shown

Table 3 The recall percentage of the extracted terms

Category number	Experiment 1			Experiment 2		
	# Key terms	# Extracted terms	Recall	# Key Terms	# Extracted Terms	Recall
Defined Categories:						
1	137	126	92%	118	114	96.6%
2	17	14	82.4%	14	13	92.9%
3	7	6	85.7%	2	2	100%
4	3	3	100%	12	12	100%
5	4	4	100%	14	14	100%
6	28	25	89.3%	52	50	96.2%
7	18	18	100%	20	17	85%
8	37	25	67.6%	12	10	83.3%
Undefined Categories:						
9	1	0	0%	0	0	0%
10	87	0	0%	33	0	0%
11	4	0	0%	8	0	0%
12	70	0	0%	15	0	0%
13	2	0	0%	4	0	0%
14	1	0	0%	0	0	0%
Total	**416**	**221**	**53.1%**	**304**	**232**	**76.3%**

in Table 3, the proposed method was able to extract 221 out of 416 key terms with recall 53.1%. It also shows that, in experiment 2, it was able to extract 232 out of 304 key terms with recall 76.3%. As mentioned in Section 9.1, the Fair Total Key Terms of every experiment is used to calculate the recall, which is the Total Key Terms after excluding the terms of category 15. The difference of the performance in the two experiments can be explained that the proposed work can extract the key terms from mathematical and science textbooks better than from theoretical textbooks.

It is noticed, as shown in Table 3, that the key terms in the defined categories represent about 60% and 80% of the total key terms in experiment 1 and 2 respectively. This observation is an evidence that the proposed model succeeded to propose patterns and algorithms to identify up to 80% of key terms defined by textbooks' authors without relying on a statistical technique or external knowledge base to measure the domain-relatedness of the extracted terms.

The proposed model deliberately avoids extracting key terms in categories 12 and 13 because their key terms came with main verbs other than the selected main verbs. Not including these verbs with the selected main verbs avoids extracting a lot of terms that are not domain-related, as these verbs can come with any term regardless of being a key term or not. Although those two categories have low weight among the key terms, 15% and 0.4% in experiment 1 and 4.8% and 1.3% in experiment 2, which means they have a low negative impact on the total recall. Therefore, avoiding them achieves two advantages: (1) having a great positive impact on the precision and (2) increasing the proposed model's accuracy of extracting domain-related terms.

On the other hand, as shown in Table 4, terms extracted in experiment 1 are 1140 out from 17889 sentences, whereas, in experiment 2, the extracted terms are 1783 out from 20878 sentences. However, the proposed work extracts terms with a precision 19% in experiment 1 and 13% in experiment 2. Nevertheless, it is worth to note that most of terms extracted other than those in the textbook's key terms section fall in the highest weight term categories. Therefore, these extracted terms are recommended to be included into the key terms section, as the textbook's author introduced them in a way that reflects their considerable importance and relevance to the domain.

Table 4 Recall and precision of the two experiments

	Experiment 1	Experiment 2	Average
Key terms	416	304	
Extracted sentences	17889	20878	
Extracted terms	1140	1783	
Key terms in defined categories	251	244	
Extracted terms from defined categories	221	232	
Recall of defined categories	88%	95%	91.5%
Recall	53.1%	76.3%	64.7%
Precision	19%	13%	16%

Another observation about the proposed model is that it can automatically filter out nouns and noun phrases in textbook's figures without the need to remove them by preprocessing operations, as they do not come in a sentence. This feature has a positive impact on precision and considered another evidence of how the proposed model employs the NLP linguistic techniques uniquely to avoid selecting not relevant terms. Nevertheless, not filtering pages of preface, copyright, dedication, table of contents, exercises, caption of figures, sentences in figures out from the textbook in the preprocessing phase has a negative impact on the precision, because the proposed work extracts all terms following the proposed patterns from these pages, which most of them are not relevant. Moreover, extracting not relevant subjects from IS_mIR sentences when extracting the TIS_mOIR terms (objects) is a bad side effect that has a negative impact on precision.

11 Conclusion and future work

This paper introduces a novel model for automatically extracting key terms from a single PDF textbook. The proposed model is a linguistic-based unsupervised machine learning approach, which does not rely on training huge amount of data or require a particular computing resources. The novelty of this approach is that it utilizes the basic NLP linguistic techniques: pattern recognition, sentence tokenization, POS tagging, and chunking, not only to extract key terms, but also to select the most relevant and domain-related ones. Furthermore, the proposed model extracts domain-related terms without relying on a statistical technique or an external knowledge base, such as Wikipedia, DBPedia, or WordNet, to measure the domain-relatedness of terms.

The proposed model proposes a unique classification of textbook sentences and terms. Sentences are classified into two main types: *is-a sentences* and *not-is-a sentences*. The *is-a sentences* include two types: IS_m and IS_x sentences. The *not-is-a sentences* also include two types of sentences: NIS_{called} and NIS_{smv}. From this classification, the proposed model identifies the *defining sentences* that are used to define relevant and important terms in textbooks. The *defining sentences* include four types of sentences: IS_m, IS_x, NIS_{called}, and NIS_{smv}. The proposed work also classifies terms into two types: *regular terms (R)* and *irregular terms (IR)*. Based on the term type, IS_m sentences only are classified into: IS_mR, IS_mR_{adv}, IS_mR_e, and IS_mIR. Consequently, the model proposes a process to extract *regular* and *irregular terms* from each type of the *defining sentences*, including the types of IS_m sentences.

The input of the proposed model is a PDF textbook, and the output of the model is a list of key terms extracted from the textbook. These key terms are extracted through 21 processes that are classified into three phases. All of these processes are built on basic NLP linguistic techniques: pattern recognition, sentence tokenization, POS tagging, and chunking, which form the foundation layer of the model. The first phase contains a process for extracting the text from a PDF textbook. It also contains four processes for cleaning text in order to work around problems found in the text extracted from the PDF textbook. The second phase contains eight processes in which eight patterns and algorithms are proposed to identify the *defining sentences* and extract them from the text. The third and last phase contains another eight processes that each has an algorithm that focuses on manipulating with a specific *defining sentence* in order to extract their key terms. The proposed model succeeded to achieve recall of 53.1% and 76.3% in experiments 1 and 2 respectively with the average of 64.7%. From the precision perspective, the proposed work extracts terms with the precision of 19% and 13% in experiments 1 and 2 respectively, with the average of 16%. This low precision is due to the large number of terms extracted by the proposed work that most of them fall in the high weight categories, even though they are not included as key terms in the key terms section by the textbook authors.

In the near future, it is planned to improve the average recall percentage of the defined categories, currently, it is 91.5%. Furthermore, studying the possibility of maximizing the overall recall percentage through extracting more terms from the undefined categories. Moreover, it is also planned to take more steps towards improving the precision percentage of the proposed work through proposing a filtering process to reduce or eliminate the number of not relevant terms in the result. It is also planned to evaluate our promising proposed work against the state-of-the-art methods that have available implementation on extracting terms from textbooks.

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB). The open access fees for this research is covered by the agreement between Springer Nature and Science, Technology & Innovation Funding Authority (STDF) in cooperation with Egyptian Knowledge Bank (EKB).

Data availability The dataset analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests The author has no relevant financial or non-financial interests to disclose.

References

Alpizar-Chacon, I., & Sosnovsky, S. (2021). Knowledge models from PDF textbooks. *New Review of Hypermedia and Multimedia, 27*(1–2), 128–176. https://doi.org/10.1080/13614568.2021.1889692

Augenstein, I., Das, M., Riedel, S., Vikraman, L., McCallum, A. (2017). Semeval 2017 task 10: Scienceie—Extracting keyphrases and relations from scientific publications. In: Proceedings of the 11th international workshop on semantic evaluation. Association for Computational Linguistics, Vancouver, Canada, pp. 546–555. https://doi.org/10.18653/v1/S17-2091

Babar, S. A., & Patil, P. D. (2015). Improving performance of text summarization. *Procedia Computer Science, 46*, 354–363. https://doi.org/10.1016/j.procs.2015.02.031

Bast, H., Korzen, C. (2017). A benchmark and evaluation for text extraction from PDF. In: 2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL), pp. 1–10. https://doi.org/10.1109/JCDL.2017.7991564

Berry, D. M., Kamsties, E., Krieger, M. (2003). From contract drafting to software specification: Linguistic sources of ambiguity. https://www.cs.uwaterloo.ca/~dberry/handbook/ambiguityHandbook.pdf . Accessed 01 Feb 2023

Campos, R., Mangaravite, V., Pasquali, A., Jorge, A., Célia, N., & Jatowt, A. (2020). YAKE! keyword extraction from single documents using multiple local features. *Information Sciences, 509*, 257–289. https://doi.org/10.1016/j.ins.2019.09.013

Castellví, M. T. C., Bagot, R. E., & Palatresi, J. V. (2001). Automatic term detection: a review of current systems. In D. Bourigault, C. Jacquemin, & M. C. L'Homme (Eds.), *Recent advances in computational terminology* (pp. 53–88). John Benjamins.

Conde, A., Larrañaga, M., Arruarte, A., Elorriaga, J. A., & Roth, D. (2016). LiTeWi: A combined term extraction and entity linking method for eliciting educational ontologies from textbooks. *Journal of the Association for Information Science and Technology, 67*(2), 380–399. https://doi.org/10.1002/asi.23398

da Silva, C. M., Felippo, A. D., Salgueiro Pardo, T. A., et al. (2014). A survey of automatic term extraction for brazilian portuguese. *Journal of the Brazilian Computer Society, 20*(1), 12. https://doi.org/10.1186/1678-4804-20-12

Duari, S., & Bhatnagar, V. (2019). sCAKE: Semantic connectivity aware keyword extraction. *Information Sciences, 477*, 100–117. https://doi.org/10.1016/j.ins.2018.10.034

Dwarakanath, A., Ramnani, R. R., Sengupta, S. (2013). Automatic extraction of glossary terms from natural language requirements. In: 2013 21st IEEE International Requirements Engineering Conference (RE), Conference Publishing Consulting, D-94034 Passau, Germany. IEEE Computer Society, pp. 314–319

El-Beltagy, S. R., & Rafea, A. (2009). KP-Miner: A keyphrase extraction system for English and Arabic documents. *Information Systems, 34*(1), 132–144.

Fox, C. (1990). A stop list for general text. *ACM SIGIR Forum, 24*(1–2), 19–21. https://doi.org/10.1145/378881.378888

Gacitua, R., Sawyer, P., & Gervasi, V. (2011). Relevance-based abstraction identification: Technique and evaluation. *Requirements Engineering, 16*(3), 251–265. https://doi.org/10.1007/s00766-011-0122-3

Gul, S., Räbiger, S., & Saygın, Y. (2022). Context-based extraction of concepts from unstructured textual documents. *Information Sciences, 588*, 248–264. https://doi.org/10.1016/j.ins.2021.12.056

Ling, S. J., Sanny, J., Moebs, W. (2021). University Physics Volume 1. OpenStax and Rice University, Houston, Texas. Retrieved February,1, 2023, from https://assets.openstax.org/oscms-prodcms/media/documents/UniversityPhysicsVol1-WEB.pdf?_gl=1*jsv1me*_ga*NDI1NzgyMDQxLjE2NjkzODk5NTg.*_ga_T746F8B0QC*MTY3NDEyMTIzMC42LjEuMTY3NDEyMTYwMi42MC4wLjA

Liu, Z., Huang, W., Zheng, Y., Sun, M. (2010). Automatic keyphrase extraction via topic decomposition. In: Proceedings of the 2010 conference on empirical methods in natural language processing (EMNLP '10), Cambridge, Massachusetts, USA, 2010, pp. 366–376.

Mikolov, T., Chen, K., Corrado, G., Dean, J. (2013a). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2. NIPS 2013, pp. 3111–3119.

Mishra, S., & Sharma, A. (2020). Automatic word embeddings-based glossary term extraction from large-sized software requirements. In N. Madhavji, L. Pasquale, A. Ferrari, & S. Gnesi (Eds.), *Requirements engineering: foundation for software quality, REFSQ 2020, Lecture Notes in Computer Science* (Vol. 12045, pp. 203–218). Springer. https://doi.org/10.1007/978-3-030-44429-7_15

Murukannaiah, P. K., Ajmeri, N., Singh, M. P. (2016). Acquiring creative requirements from the crowd: understanding the influences of individual personality and creative potential in crowd RE. In: 24th IEEE International Requirements Engineering Conference (RE), pp. 176–185.

Murukannaiah, P. K., Ajmeri, N., Singh, M. P. (2017). Toward automating crowd RE. In: 25th IEEE International Requirements Engineering Conference (RE), pp. 512–515.

OpenStax. (2019). Organizational Behavior. OpenStax and Rice University, Houston, Texas. Retrieved February,1, 2023, from https://assets.openstax.org/oscms-prodcms/media/documents/OrganizationalBehavior-OP_TtwWIeQ.pdf?_gl=1*zh7fax*_ga*NDI1NzgyMDQxLjE2NjkzODk5NTg.*_ga_T746F8B0QC*MTY2OTM5MzYyNC4zLjEuMTY2OTM5MzgyNy41Ni4wLjA

Papagiannopoulou, E., & Tsoumakas, G. (2020). A review of keyphrase extraction. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 10*(2), e1339. https://doi.org/10.1002/widm.1339

Park, Y., Byrd, R. J., Branimir, K. B. (2002). Automatic glossary extraction: beyond terminology identification. In: Proceedings of the 19th international conference on computational linguistics, Volume 1. Association for Computational Linguistics, pp. 1–7. https://doi.org/10.3115/1072228.1072370

Rousseau, F., & Vazirgiannis, M. (2015). Main core retention on graph-of-words for single-document keyword extraction. In A. Hanbury, G. Kazai, A. Rauber, & N. Fuhr (Eds.), *Advances in information retrieval, ECIR 2015, Lecture Notes in Computer Science.* (Vol. 9022). Springer. https://doi.org/10.1007/978-3-319-16354-3_42

Salton, G. (1971). *The smart retrieval system - experiments in automatic document processing*. Prentice-Hall.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management, 24*(5), 513–523.

Sleator, D. D. K., Temperley, D. (1993). Parsing English with a link grammar. In: Proceedings of the third international workshop on parsing technologies, Tilburg, Netherlands and Durbuy, Belgium. Association for Computational Linguistics, pp. 277–292

Stair, R. M., & Reynolds, G. W. (2012). *Fundamentals of information systems* (6th ed.). Cengage Learning.

Sun, C., Hu, L., Li, S., Li, T., Li, H., & Chi, L. (2020). A review of unsupervised keyphrase extraction methods using within-collection resources. *Symmetry, 12*(11), 1864. https://doi.org/10.3390/sym12111864

Tkaczyk, D., Szostek, P., Fedoryszak, M., Dendek, P. J., & Bolikowski, Ł. (2015). CERMINE: Automatic extraction of structured metadata from scientific literature. *International Journal on Document Analysis and Recognition, 18*(4), 317–335. https://doi.org/10.1007/s10032-015-0249-8

Wang, R., & Wang, G. (2019). Web text categorization based on statistical merging algorithm in big data. *International Journal of Ambient Computing and Intelligence, 10*(3), 17–32. https://doi.org/10.4018/IJACI.2019070102

Whitington, J. (2011). *PDF explained.* O'Reilly Media.

Witten, I. H., Paynter, G. W., Frank, E., Gutwin, C., Nevill-Manning, C. (1999). KEA: Practical automatic keyphrase extraction. In: Proceedings of the fourth ACM conference on digital libraries, Berkeley, CA, USA, pp. 11–14

Xu, Z., & Zhang, J. (2021). Extracting keywords from texts based on word frequency and association features. *Procedia Computer Science, 187*, 77–82. https://doi.org/10.1016/j.procs.2021.04.035