ICON 2020

17th International Conference on Natural Language Processing

Proceedings of the TermTraction 2020 Shared Task

December 18 - 21, 2020
Indian Institute of Technology Patna, India

Introduction

These shared task proceedings concluded the shared task on Unsupervised Technical Domain Terms Extraction, also named as TermTraction - 2020, launched on 7th October 2020. The shared task was collocated with the 17th International Conference on Natural Language Processing (ICON 2020), held at IIT-Patna, India. The goal of the shared task was to extract domain-specific terms given an English document.

Four technical domains were chosen for this task - BioChemistry, Communication, Computer Science and Law. The documents that were part of the training corpus did not contain any marked or annotated domain terms. The participants were tasked to develop an unsupervised algorithm for the extraction of the terms.

We received three system submissions and system description papers. Each system description paper was reviewed by two members of the reviewing committee – all papers were accepted. Macro F1-score was used as the evaluation metrics.

Two major categories of algorithms were used by the participants - Graph-based and Filter based. Textrank was the most used graph-based technique. Rapid Automatic Keyword Extraction (RAKE) and PYthon Automated Term Extraction (PYATE) were the filter based techniques employed by the teams. We would like to thank the ICON-2020 organizers, the shared task participants, the authors, and the reviewers for making this shared task successful.

Shared task page: `http://ssmt.iiit.ac.in/TermTraction`
Main conference page: `https://www.iitp.ac.in/~ai-nlp-ml/icon2020/index.html`

Organizing Committee:

Table of Contents

Shared Task Program

Monday, December 21, 2020

+ 14:00 - 14:30 **Talk by Sobha L, AUKBC-Chennai**

+ 14:30 - 14:45 Shared Task Overview

Presentations

16:00 - 16:10	*Graph Based Automatic Domain Term Extraction* Hema Ala and Dipti Sharma
16:13 - 16:23	*N-Grams TextRank A Novel Domain Keyword Extraction Technique* Saransh Rajput, Akshat Gahoi, Manvith Reddy and Dipti Mishra Sharma
16:26 - 16:36	*Unsupervised Technical Domain Terms Extraction using Term Extractor* Suman Dowlagar and Radhika Mamidi

Graph Based Automatic Domain Term Extraction

Hema Ala
LTRC, IIIT-Hyderabad, India
hema.ala@research.iiit.ac.in

Dipti Misra Sharma
LTRC, IIIT-Hyderabad, India
dipti@iiit.ac.in

Abstract

We present a Graph Based Approach to automatically extract domain specific terms from technical domains like Biochemistry, Communication, Computer Science and Law. Our approach is similar to TextRank with an extra post-processing step to reduce the noise. We performed our experiments on the mentioned domains provided by ICON TermTraction - 2020 shared task. Presented precision, recall and f1-score for all experiments. Further, it is observed that our method gives promising results without much noise in domain terms.

1 Introduction

Domain Term, is a word or group of words, carrying a special, possibly complex, conceptual meaning, within a specific domain or subject field or community. Because of their low ambiguity and high specificity, these words are also particularly useful to conceptualize a knowledge subject. For each domain, there is an essential need to identify the domain-specific terms as they play a vital role in many *Natural Language Processing Applications* such as Neural Machine Translation(NMT) (Dinu et al., 2019), Information Retrieval (Chien, 1999), Information Extraction (Yangarber et al., 2000), Text Classification (Liu et al., 2005), etc. The task of automatically extracting domain specific terms from a given text of a certain academic or technical domain, is known as *Automatic Technical Domain Term Extraction*. This is a predominant task in NLP. Extracted terms can be useful in more complex tasks such as NMT (Dinu et al., 2019), Ontology Construction (Kietz et al., 2000; Wu and Hsu, 2002), Domain Identification, Semantic Search, Question-Answering, Word Sense Induction, etc. Several research works have been carried out to extract domain-specific terms. Most of them are either rule based (Collard et al., 2018) or dictionary based (Kim and Cavedon, 2011). Also, there are few term extraction techniques which uses machine learning algorithms (Fedorenko et al., 2014), thereby demanding a huge labelled corpus. But, the existence of domain term annotated corpus is very rare in case of many domains. Also, the availability of such huge labelled corpus is almost nil for low resource languages. Therefore, our domain term extraction approach is motivated more by unsupervised than supervised strategies. Hence, we used a Graph Based Approach which extract not only unigrams but also collocations. ***Collocations*** are expressions of multiple words which commonly co-occur in a given context than its individual word parts. These are the phrases that express stronger sentiment which can be easily captured with bigram, trigram and so on. Hence our approach is not restricted to just unigram extraction, it also considers multi-word domain terms [1]. To demonstrate the performance of our approaches, we used data provided by ICON TermTraction - 2020 shared task. The discussion and analysis on the performance of the approaches are mentioned in section 4. In this paper we performed our experiments on four domains in English. We are still in the process of exploring the possible unsupervised approaches to extract domain terms in a flexible and intuitive manner. Further, it can be applicable to all domains irrespective of any language.

2 Background & Motivation

There have been a lot of studies regarding the automatic domain term extraction. But very less work carried on unsupervised approaches that too on technical domains like, computer science, chemistry, etc. Automatic domain term extraction is a categorization or classification task where terms

[1] Covalent Bond, Amino Acid, Hydrophobic Hydrogen Bond, Artificial Intelligence, Support Vector Machines, Natural Language Processing, etc are few examples of bigram and trigram collocations

are categorized into a set of predefined domains (Velardi et al., 2001; Xu et al., 2002). Further this task is used in many NLP applications such as domain ontology construction and NMT with Domain Terminology by injecting custom terminology into neural machine translation at run time (Dinu et al., 2019). In order to effectively make use of domain terms in various applications, an ultimate approach which is fast, flexible and reliable is highly required. In spite of many contributions on automatic domain term extraction, very limited study is done so far using unsupervised approaches. Most of the explorations are done using supervised methods such as focusing on various features like contextual, domain concepts and topics to measure the semantic similarity of terms to assign domain concepts to domain-specific terms (Kim and Cavedon, 2011). Similarly, another experimental evaluation is done by comparing the performance of two existing approaches for Automatic Domain Term Recognition: *Machine Learning Method and Voting Algorithm*(Fedorenko et al., 2014). But, the major well known drawback with these supervised algorithms is that, they demand huge labelled training data. Therefore, an unsupervised algorithm is more preferable. Most of such unsupervised approaches extract domain-specific terms using frequency count (VRL, 2009). The basic underlying idea is that, in a particular domain, domain-specific terms occur with markedly higher frequency than they do in other domains, similar to term frequency patterns captured by TF-IDF. Apart from these methods, another experimental approach for domain term extraction is executed using Deep Learning where possible *term spans* within a fixed length in the sentence, is considered to predict a domain term. Deep Learning technique is proven to yield high recall and a comparable precision on term extraction task (Gao and Yuan, 2019). However, for training such Deep Learning models, an enormous training data is mandatory. Conversely, availability of this sort of corpus for diverse multilingual domain is very scarce. Our goal is to formulate a flexible and reliable approach which successfully extracts domain terms irrespective of the domain and language of a document. Accordingly, we present experiments which extract domain terms in a given document disregarding of any domain without having a dependency on labelled corpus. Our approach , *TextRank* is an inspiration from PageRank algorithm (Brin and Page, 1998). (Mihalcea and Tarau, 2004) Introduced TextRank a graph-based ranking model for text processing, and showed how this model can be successfully used in natural language applications. In particular keyword and sentence extraction. We reimplemented TextRank from Mihalcea and Tarau (2004) for extracting domain-specific terms from technical domains like, computer science , chemistry, etc by handling noise generated in the outputs. *TextRank* is merely a graph based approach where *words* are considered as nodes and the *relation* between them as edges. Based on syntactic filters, such as Parts of Speech (POS) Tags, words are selected as nodes and relation between the words is based on word co-occurrences . A window size (N) is assumed for word co-occurrences. For all words that fall in a particular window, an edge is allocated, resulting into a graph of nodes and edges. An undirected and unweighted graph is considered in our approach. This is further discussed in detail in Section 3.

3 Approach

A graph-based ranking algorithm is a way of deciding on the importance of a vertex within a graph by taking into account global information recursively computed from the entire graph, rather than relying only on local vertex-specific information. Applying a similar line of thinking to lexical or semantic graphs extracted from natural language texts, results in a graph-based ranking model that can be applied to a variety of natural language processing applications, where knowledge drawn from an entire text is used in making local ranking/selection decisions. We implemented the TextRank algorithm described in (Mihalcea and Tarau, 2004). Mihalcea and Tarau (2004) described usage of TextRank for keyword extraction and sentence extraction but we adopted that technique for automatic domain term extraction by doing few modifications in syntactic filters, and adding a post processing step for noise removal using top 1000 common words in English from Wikipedia. we used Noun, Proper Nouns, Adjectives as syntactic filters, window size ($N = 4$)is used in all experiments and calculated precision , recall and F1 score.

TextRank is completely unsupervised, and unlike other supervised systems, it relies completely on information drawn from the text itself, which makes it easily portable to other domains, and languages. Intuitively, TextRank works well because it does not only rely on the local context of a text unit (vertex), but rather it takes into account information recursively drawn from the entire text (graph). Through the graphs it builds on texts, TextRank identifies connections between various entities in a text, and implements the concept of recommendation. A text unit recommends other related text units, and the strength of the recommendation is recursively computed based on the importance of the units making the recommendation.

The brief explanation of each step in Text Rank algorithms is given as follows, firstly the text is tokenized, and annotated with part of speech tags, for this task we used Spacy (Honnibal and Montani, 2017). To evade the excessive growth of graph size by including all possible combinations of sequences consisting of more than one lexical unit(word), we consider only single words as nodes to build the graph, with multi-word domain terms being eventually reconstructed in the post-processing step. Following, all words that pass the syntactic filter are added to the graph, and an edge added between those words that co-occur within a window of N words. After the graph construction (undirected , unweighted graph), the value of each vertex is set to 1. Next, the ranking algorithm will run on the graph for several iterations until it converges usually for 20-30 iterations, at a threshold of 0.0001(Mihalcea and Tarau, 2004). Once a final score is achieved for each vertex (for each word)in the graph, vertices are sorted in reversed order of their score then the top K words in the ranking are retained for post-processing. In our experiments we take top $K = n/3$ where n is total number of unique words in the text. In post processing step along with constructing n-grams we reduce the noise using top 1000 English words from Wikipedia. To construct n-grams from unigrams we get, first annotate the text with technical domain terms we get then retrieve the terms which occur side by side in the text.

4 Experiments & Results

We evaluate our approach on data provided by ICON TermTraction - 2020 shared task for four domains, Biochemistry, communication , Computer Science and Law. Each domain contains files with text related to that domain. In each domain we have minimum 10 files and maximum 16 files. We did experiments on individual files for the respective domain. As our approach comes under unsupervised learning, there is no requirement of training data. Results of each file in specific domain showed in table 1 , 2 , 3 for Biochemistry, Communication and Computer Science respectively. For Law do-

File	Precision	Recall	F1
1	0.15	0.45	0.22
2	0.07	0.21	0.10
3	0.17	0.35	0.23
4	0.16	0.48	0.24
5	0.07	0.67	0.13
6	0.36	0.62	0.45
7	0.22	0.57	0.32
8	0.17	0.63	0.26
9	0.22	0.63	0.33
10	0.24	0.54	0.33

Table 1: Scores of individual files in BioChemistry

File	Precision	Recall	F1
1	0.08	0.54	0.14
2	0.06	0.5	0.11
3	0.06	0.31	0.10
4	0.16	0.5	0.24
5	0.12	0.77	0.20
6	0.09	0.68	0.16
7	0.05	0.69	0.09
8	0.25	0.37	0.05
9	0.06	0.56	0.11
10	0.08	0.55	0.15

Table 2: Scores of individual files in Communication

File	Precision	Recall	F1
1	0.17	0.52	0.26
2	0.18	0.61	0.27
3	0.12	0.48	0.19
4	0.09	0.55	0.17
5	0.14	0.63	0.23
6	0.06	0.68	0.12
7	0.12	0.57	0.20
8	0.16	0.46	0.23

Table 3: Scores of individual files in Computer Science

Domain	Precision	Recall	F1
BioChemistry	0.18	0.52	0.26
Communication	0.08	0.54	0.14
Computer Science	0.13	0.56	0.20
Law	0.05	0.5	0.10

Table 4: Average precision , recall and f1-scores

main also the behaviour is same as above three domains. From the results of all domains, we can observe that Recall is very high compared to Precision, from this we can infer our algorithm is not producing much noise. In table 4 we have averaged precision , recall and f1-score for each domain. overall we got promising results for technical domain term extraction for all given domains.

5 Conclusion & Future Work

In this paper we showed a graph based approach for automatic technical domain term extraction for four technical domains(BioChemistry, Computer Science, Communication,Law). Our approach showed high recall in all cases for all domains, from this we can conclude that our model has the power to extract domain-specific terms without much noise. Our approach doesn't depend on any language dependant resources except POS tagger, hence we can adopt this method for any language. We plan to extend our approach to possible Indian languages like Telugu, Hindi etc. And we would like to improve this approach with different word relationships(edge relations like we did using cooccurrence of words in given window). One approach for that is like using similarity of words using word2vec etc.

References

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine.

L-F Chien. 1999. Pat-tree-based adaptive keyphrase extraction for intelligent chinese information retrieval. *Information processing & management*, 35(4):501–521.

Jacob Collard, TN Bhat, Eswaran Subrahmanian, Ram D Sriram, John T Elliot, Ursula R Kattner, Carelyn E Campbell, and Ira Monarch. 2018. Generating domain terminologies using root-and rule-based terms 1. *Washington Academy of Sciences. Journal of the Washington Academy of Sciences*, 104(4):31–78.

Georgiana Dinu, Prashant Mathur, Marcello Federico, and Yaser Al-Onaizan. 2019. Training neural machine translation to apply terminology constraints. *arXiv preprint arXiv:1906.01105*.

Denis Fedorenko, N Astrakhantsev, and D Turdakov. 2014. Automatic recognition of domain-specific terms: an experimental evaluation. *Proceedings of the Institute for System Programming*, 26(4):55–72.

Yuze Gao and Yu Yuan. 2019. Feature-less end-to-end nested term extraction. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 607–616. Springer.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Jörg-Uwe Kietz, Raphael Volz, and Alexander Maedche. 2000. Extracting a domain-specific ontology from a corporate intranet. In *Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*.

Su Nam Kim and Lawrence Cavedon. 2011. Classifying domain-specific terms using a dictionary. In *Proceedings of the Australasian Language Technology Association Workshop 2011*, pages 57–65.

Tao Liu, Xiao-long Wang, Y Guan, Zhi-Ming Xu, et al. 2005. Domain-specific term extraction and its application in text classification. In *8th Joint Conference on Information Sciences*, pages 1481–1484.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

Paola Velardi, Michele Missikoff, and Roberto Basili. 2001. Identification of relevant terms to support the construction of domain ontologies. In *Proceedings of the ACL 2001 Workshop on Human Language Technology and Knowledge Management*.

NICTA VRL. 2009. An unsupervised approach to domain-specific term extraction. In *Australasian Language Technology Association Workshop 2009*, page 94.

Shih-Hung Wu and Wen-Lian Hsu. 2002. Soat: a semi-automatic domain ontology acquisition tool from chinese corpus. In *COLING 2002: The 17th International Conference on Computational Linguistics: Project Notes*.

Feiyu Xu, Daniela Kurz, Jakub Piskorski, and Sven Schmeier. 2002. A domain adaptive approach to automatic acquisition of domain relevant terms and their relations with bootstrapping. In *LREC*.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Automatic acquisition of domain knowledge for information extraction. In *COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics*.

Unsupervised Technical Domain Terms Extraction using Term Extractor

Suman Dowlagar
LTRC
IIIT-Hyderabad
suman.dowlagar@research.iiit.ac.in

Radhika Mamidi
LTRC
IIIT-Hyderabad
radhika.mamidi@iiit.ac.in

Abstract

Terminology extraction, also known as term extraction, is a subtask of information extraction. The goal of terminology extraction is to extract relevant words or phrases from a given corpus automatically. This paper focuses on the unsupervised automated domain term extraction method that considers chunking, preprocessing, and ranking domain-specific terms using relevance and cohesion functions for ICON 2020 shared task 2: TermTraction.

1 Introduction

The aim of Automatic Term Extraction (ATE) is to extract terms such as words, phrases, or multi-word expressions from the given corpus. ATE is widely used in many NLP tasks, such as machine translation, summarization, clustering the documents, and information retrieval.

Unsupervised algorithms for domain term extraction are not labeled and trained on the corpus and do not have any pre-defined rules or dictionaries. They often use statistical information from the text. Most of these algorithms use stop word lists and can be applied to any text datasets. The standard unsupervised automated term extraction pipeline consists of

- Simple Rules: using chunking or POS tagging to extract Noun phrases for multi-word extraction.
- Naive counting: that counts how many terms each word occurs in the corpus.
- Preprocessing: Removing punctuation and common words such as stop words from the text.
- Candidate generation and scoring: using statistical measures and ranking algorithms to generate the possible set of domain terms
- Final set: Arrange the ranked terms in descending order based on the scores and take the top N keywords as the output.

Currently, there are many methods for automatic term recognition. Evans and Lefferts (1995) used TF-IDF measure for term extraction.Navigli and Velardi (2002) used domain consensus which is designed to recognize the terms uniformly distributed over the whole corpus. The most popular method C-value (Frantzi et al., 2000) is also a statistical measure that extracts a term based on the term's frequency, length of the term, and the set of the candidates that enclose the term such that the term is in their substring. Bordea et al. (2013) proposed the method called Basic, which is a modification of the C-value for recognizing terms of average specificity. The successor of C-value statistic called the NC value (Frantzi et al., 2000) considered scored the term based on the condition if it exists in a group of common words or if it contains nouns, verbs, or adjectives that immediately precede or follow the term. The methods proposed by Ahmad et al. (1999); Kozakov et al. (2004); Sclano and Velardi (2007) are based on extracting the terms of a text by considering the frequency of occurrence of terms in the general domain.

A detailed survey of the existing automated term extraction algorithms and their evaluation are presented in papers by Astrakhantsev et al. (2015); Šajatović et al. (2019)

In this paper, we used the term extractor algorithm (Sclano and Velardi, 2007) present in the pyate[1] library for domain term extraction. The term extractor algorithm is developed initially for ontology extraction from large corpora. It uses domain pertinence/relevance, domain consensus, and lexical cohesion for extracting terms. A detailed description of the modules is given in the

[1]https://pypi.org/project/pyate/

next section.

The paper is organized as follows. Section 2 gives a detailed description of the term extraction algorithm used. Section 3 gives information about the datasets used and results. Section 4 concludes the paper.

2 Our Approach

In this section, we describe in detail the methods used in the term extractor algorithm.

Initially, TermExtractor performs chunking and proper name recognition and then extracts structures based on linguistic rules and patterns, including stop words, detection of misspellings, and acronyms. The extraction algorithm uses Domain Pertinence, Domain Cohesion, and Lexical Cohesion to decide if a term is considered a domain term.

Domain Pertinence, or Domain Relevance (DR), requires a contrastive corpus and compares a candidate's occurrence in the documents belonging to the target domain to its occurrence in other domains, but the measure only depends on the contrastive domain where the candidate has the highest frequency. The Domain Pertinence is based on a simple formula,

$$DR_{D_i}(t) = \frac{tf_i}{max_j(tf_j)} \tag{1}$$

Where tf_i is the frequency of the candidate term in the input domain-specific document collection and $max_j(tf_j)$ is the general corpus domain, where the candidate has the highest frequency, and D_i is the domain in consideration.

Domain Consensus (DC) assumes that several documents represent a domain. It measures the extent to which the candidate is evenly distributed on these documents by considering normalized term frequencies (ϕ),

$$DC_{D_i}(t) = \sum_{d_k \epsilon D_i} \phi_k log \phi_k \tag{2}$$

Here, we assume k distinct documents for the domain D_i.

Lexical cohesion involves the choice of vocabulary. It is concerned with the relationship that exists between lexical items in a text, such as words and phrases. It compares the in-term distribution of words that make up a term with their out-of-term distribution.

Domain	**#Train docs**	**#Test docs**
Bio-Chemistry	229	10
Communication	127	10
Computer-Science	201	8
Law	70	16

Table 1: Data statistics

$$LC_{D_i}(t) = \frac{n * tf_i * logtf_i}{\sum_j tf_{w_j} i} \tag{3}$$

Where n is the number of documents in which the term t occurs.

The final weight of a term is computed as a weighted average of the three filters above,

$$score(t, D_i) = \alpha * DR + \beta * DC + \gamma * LC \tag{4}$$

where α, β, γ are the weights, and they are equal to $1/3$

3 Experiments

This section describes the dataset used for domain terms extraction, implementation of the above approach, followed by results, and error analysis.

3.1 Dataset

We used the dataset provided by the organizers of TermTraction ICON-2020. The task is to extract domain terms from the given English documents from the four technical domains like Computer Science, Physics, Life Science, Law. The data statistics of the documents in the respective domains are shown in the table 1.

3.2 Implementation

We used Pyate (python automated term extraction library) that contains the term extractor method and is trained on the general corpus. With the help of the term extraction method, we extracted the relevant terms from the given corpus.

We have submitted two runs, one run (run 1) is the term extractor function itself, and the other run (run 2) is term extractor combined with NP chunks of phrase length ¿ 2 obtained from NLTK ConsecutiveNPChunkTagger[2].

[2]ConsecutiveNPChunkTagger

Biochemistry			Communication			Computer Science			Law		
Data	**run 1**	**run 2**	**Data**	**run 1**	**run 2**	**Data**	**run 1**	**run 2**	**Data**	**run 1**	**run 2**
M12S1	0.247	0.222	**M2-1**	0.109	0.086	**KL2**	0.220	0.225	**A01**	0.079	0.077
M15S2	0.208	0.195	**M2-2**	0.102	0.104	**KL4**	0.241	0.246	**A02**	0.099	0.066
M16S2	0.224	0.207	**M2-3**	0.094	0.074	**KL8**	0.138	0.146	**A03**	0.144	0.126
M23S3	0.266	0.233	**M3-1**	0.240	0.236	**W12**	0.143	0.122	**FA1**	0.104	0.116
M26S2	0.096	0.081	**M3-2**	0.159	0.148	**W1332**	0.216	0.195	**FA2**	0.077	0.067
T18	0.463	0.427	**M3-3**	0.140	0.132	**W13**	0.108	0.089	**FC1**	0.082	0.073
T25	0.310	0.282	**RM16**	0.101	0.088	**W1436**	0.181	0.165	**FC2**	0.032	0.021
T39	0.265	0.247	**RM17**	0.067	0.065	**W921**	0.221	0.188	**FC3**	0.016	0.014
T4	0.271	0.234	**RM18**	0.098	0.115				**FR1**	0.149	0.113
T9	0.323	0.315	**SW1AW**	0.120	0.113				**FR2**	0.144	0.112
									FR3	0.073	0.062
									G3	0.103	0.098
									G4	0.056	0.052
									R1	0.022	0.055
									R2	0.033	0.026
									R3	0.044	0.048

Table 2: Term Extraction macro-F1 score.

Template Sentence	**Domain terms identified**
We are not going to that , remove it completely, but nevertheless this is an indication that , **NO plus** is going to be a **poorer donor** , compared to **carbon monoxide** . So , this drastic reduction in the **stretching frequency** can only happen if you have , a large population of the **anti - bonding orbitals** of **NO plus** . And it has got a structure , which is very similar , a structure which is very similar to the structure of **nickel tetra carbonyl** . You will see that , while **carbon monoxide** is ionized with 15 **electron volts** , if you supply 15 **electron volts** , **carbon monoxide** can be **oxidized** or **ionized** .	large population similar ionized carbonyl frequency poorer donor anti - bonding orbitals indication carbon monoxide electron volts nickel plus drastic reduction structure

Table 3: Error analysis on the template sentence

3.3 Results and Error Analysis

We evaluated the performance of the method using average precision. The results are tabulated in Table 2.

For the template sentence given in Table 3, our algorithm failed to recognize the domain terms **NO plus** and **nickel tetra carbonyl**. It considered **NO** as the stop word (no or negation) and discarded it while preprocessing. The algorithm also misunderstood words like “similar” as domain terms and failed to identify **nickel tetra carbonyl** as a domain term. It indicates that further study is necessary, which considers the candidate terms’ capitalization and uses better methods that support the more reliable form of compound words or multi-word expressions.

4 Conclusion

For domain term extraction from technical domains like Bio-Chemistry, Law, Computer-Science, and communication, We used the term extractor method from pyate library for obtaining technical terms.

The term extractor method uses keywords from the general corpora, and it considers Domain Pertinence, Domain Cohesion, and Lexical Cohesion methods for extracting domain terms in the given corpus.

As mentioned above, it did not give preference to capitalized terms and did not consider some compound words. So we have to work towards better methods that consider capitalization, better formation of compound words for the more reliable performance of the automated domain term extractor.

References

Khurshid Ahmad, Lee Gillam, Lena Tostevin, et al. 1999. University of surrey participation in trec8: Weirdness indexing for logical document extrapolation and retrieval (wilder). In *TREC*, pages 1–8.

Nikita A Astrakhantsev, Denis G Fedorenko, and D Yu Turdakov. 2015. Methods for automatic term recognition in domain-specific text collections: A survey. *Programming and Computer Software*, 41(6):336–349.

Georgeta Bordea, Paul Buitelaar, and Tamara Polajnar. 2013. Domain-independent term extraction through domain modelling. In *The 10th international conference on terminology and artificial intelligence (TIA 2013), Paris, France*. 10th International Conference on Terminology and Artificial Intelligence.

David A Evans and Robert G Lefferts. 1995. Clarit-trec experiments. *Information processing & management*, 31(3):385–395.

Katerina Frantzi, Sophia Ananiadou, and Hideki Mima. 2000. Automatic recognition of multi-word terms:. the c-value/nc-value method. *International journal on digital libraries*, 3(2):115–130.

Lev Kozakov, Youngja Park, T Fin, Youssef Drissi, Yurdaer Doganata, and Thomas Cofino. 2004. Glossary extraction and utilization in the information search and delivery system for ibm technical support. *IBM Systems Journal*, 43(3):546–563.

Roberto Navigli and Paola Velardi. 2002. Semantic interpretation of terminological strings. In *Proc. 6th Int'l Conf. Terminology and Knowledge Eng*, pages 95–100.

Antonio Šajatović, Maja Buljan, Jan Šnajder, and Bojana Dalbelo Bašić. 2019. Evaluating automatic term extraction methods on individual documents. In *Proceedings of the Joint Workshop on Multiword Expressions and WordNet (MWE-WN 2019)*, pages 149–154.

Francesco Sclano and Paola Velardi. 2007. Termextractor: a web application to learn the shared terminology of emergent web communities. In *Enterprise Interoperability II*, pages 287–290. Springer.

N-Grams TextRank : A Novel Domain Keyword Extraction Technique

Saransh Rajput **Akshat Gahoi** **Manvith Reddy** **Dipti Mishra Sharma**

Language Technologies Research Center
International Institute of Information Technology, Hyderabad, India

{saransh.rajput,akshat.gahoi}@research.iiit.ac.in
manvith.reddy@students.iiit.ac.in
dipti@iiit.ac.in

Abstract

The rapid growth of the internet has given us a wealth of information and data spread across the web. However, as the data begins to grow we simultaneously face the grave problem of an *Information Explosion*. An abundance of data can lead to large scale data management problems as well as the loss of the true meaning of the data. In this paper, we present an advanced domain specific keyword extraction algorithm in order to tackle this problem of paramount importance. Our algorithm is based on a modified version of TextRank(Mihalcea and Tarau, 2004) algorithm - an algorithm based on PageRank(Page et al., 1998) to successfully determine the keywords from a domain specific document. Furthermore, this paper proposes a modification to the traditional TextRank algorithm that takes into account bigrams and trigrams and returns results with an extremely high precision.

We observe how the precision and f1-score of this model outperforms other models in many domains and the recall can be easily increased by increasing the number of results without affecting the precision. We also discuss about the future work of extending the same algorithm to Indian languages.

1 Introduction

Graph based ranking algorithms have proved to be useful for tasks which involve ranking or ordering. This includes important tasks like citation analysis and ranking webpage results. Graph based ranking Algorithms are used in many key areas even today. PageRank, an algorithm developed by the founders of Google, was the primary algorithm used to rank webpage searches until 2018.

The fundamental idea behind any graph based ranking algorithm is make use of global knowledge for making local decisions. To determine the importance of a node in a graph, we recursively look at other nodes to gain more information.

More recently, the applications of graph based algorithms have extended to other domains as well, including Natural Language Processing. This includes the use of Textrank algorithms for summarisation, word sense disambiguation(Mihalcea et al., 2004) and keyword extraction tasks. Knowledge extracted from the whole text is considered while making local decisions.

In this paper we introduce and evaluate an unsupervised approach for the task of domain terminology extraction. We employ the Textrank algorithm for this task with a few modifications. Taking into account that domain terms are often multi-worded expressions, we consider bigrams and trigrams as nodes in the graph with suitable additional weight to these nodes. Furthermore, terms are filtered based on their POS(Manning, 2011) tags in order to remove excessive domain-less words.

2 Pre-processing and Data

For our study we used a dataset that contained a collection of over 800 domain specific documents. The dataset featured documents from 4 distinct domains namely : **Bio-Chemistry, Communication, Computer Science and Law**.

Before passing a document through the model, it was crucial to carry out fundamental preprocessing in order to achieve a high standard of results. We first removed non-essential punctuations and tokenized the the document. In addition to the elementary NLTK(Bird, 2006)/Spacy Stop word list, we curated an additional specific list of common words that we observed added no meaning to our algorithm. POS Tagging(Brants, 2002) was a critical part of our model and was based on the powerful assumption that if a term is domain specific then it is often a **Noun** or a **Verb**, which we made after analyzing the data meticulously. The addition of a POS tagger gave us a significant increase in the f1-score.

3 TextRank

TextRank is a graph-based ranking model(like HITS(Kleinberg, 1999)) for text processing which can be used in order to find the most relevant keywords in a text. TextRank is an algorithm based on PageRank which we will explain briefly.

PageRank is an algorithm used for computing a ranking for every web page based on the graph of the web and helps in measuring the relative importance of specific web pages. We can take all web pages to be directed graph(Georgiadis et al., 2014). In this graph, a node is a webpage. If webpage A has the link to web page B, it can be represented as a directed edge from A to B. After we construct the whole graph, we can assign weights for web pages by the following formula.

$$S(V_i) = (1-d) + d * \sum_{j \in In(v_i)} \frac{S(V_j)}{|Out(V_j)|} \tag{1}$$

where S represents the weight of a webpage, d represents a damping factor, $In(V_i)$ represents the set of nodes having an edge directed to node i and $Out(V_i)$ represents the set of nodes which have an incoming edge from node i.

TextRank is conceptually the same algorithm as PageRank with the difference being that nodes in the graph are words rather than webpages. In order to find relevant keywords, the TextRank algorithm constructs this word graph. The graph is constructed by looking which words follow one another. An edge is set up between two words if they are located within a window of a size of our choice, the link gets a higher weight if these 2 words occur more frequently next to each other in the text. As we can see, preprocessing plays a huge part in the TextRank algorithm without which the results would easily be skewed towards common stop words and punctuations.

4 Our Implementation

In our model(Code can be found here [1]), we began the process by passing a document through our preprocessing pipeline. An input document was first split into sentences on the basis of end of sentence punctuation marks and then further tokenized using the SpaCy tokenizer. Stop Words were filtered out using the SpaCy stop words list along with our extensive custom list of domain-less terminology. Furthermore, we use the powerful tool of POS tagging in order to filter out irrelevant words and make the computation of our model much quicker. After extensive research we determined that **Nouns** and **Verbs** contained most of the important domain related terminology.

We are now left with tokenized sentences of each document. We further extract Unigrams, Bigrams and Trigrams from these documents and take them as seperate nodes in our TextRank Graph. We initialize our TextRank graph with a window-size of 4, which means that 4 words around every n-gram will be considered eligible to have an edge with the n-gram. After extensive trial and error, we observed that using a larger window size significantly increased the execution time of the model without much improvement in results. In some cases, an edge was added between two totally unrelated nodes due to large window size. Inspecting equation will reveal that we need to set a damping factor to assign how much relative importance to give the score calculated by the graph. After multiple runs, we achieved the greatest results with a damping factor of 0.85 which gives great importance to the score churned out by the TextRank Graph. We further set our convergence threshold at 10e-5 for our termination condition.

Once built, this graph is then used to calculate weights for each node. The weight of a node essentially represents its contribution to the document. We observed that although multi word domain terms are important to the document and contribute significantly, they are usually less frequent. Often they can get replaced by pronoun terms as well. In order to counter this neglection of the n-gram nodes, we introduce a novel weighting system to the traditional TextRank Algorithm. The weights of Bigram and Trigram nodes are taken as a parameter to our model and our multiplied to the final score returned by the traditional TextRank Algorithm. This imporves our results astronomically with many key bigrams and trigrams showing up as a result. These weights are useful in the hands of a

[1] https://github.com/akshatgui/Domain_Teminology_Extraction

Domain	Run	Precision	Recall	F-1
Law	$Run1$	$\mathbf{0.4}$	$\mathbf{0.32}$	$\mathbf{0.355}$
	$Run2$	0.266	0.32	0.29
	$Run3$	0.133	0.285	0.181
$Communication$	$Run1$	$\mathbf{0.25}$	0.208	0.227
	$Run2$	0.233	$\mathbf{0.291}$	$\mathbf{0.259}$
	$Run3$	0.1	0.125	0.111
$ComputerScience$	$Run1$	0.251	0.13	0.174
	$Run2$	0.3	0.134	0.185
	$Run3$	$\mathbf{0.466}$	$\mathbf{0.152}$	$\mathbf{0.229}$
$Bio-Chemistry$	$Run1$	$\mathbf{0.501}$	0.131	0.208
	$Run2$	0.3	0.173	0.219
	$Run3$	0.466	$\mathbf{0.184}$	$\mathbf{0.264}$

Table 1: Results for the task of Domain Extraction for different Bigram and Trigram Weights

domain expert who would be able to determine the right weights for each domain in order to get the best results.

Across the many domains, our Law Domain results were of extremely high quality and showed that both high precision and recall can be attained by our model.

5 Results and Evaluation

The model was tested on 10 domain specific documents of **Bio-Chemistry, Communication,Computer Science and Law**. We ran three different runs of the model, each with varying bigram and trigram weights. Run 1 featured a bigram weight of '1.8' and trigram weight of '1.5'. Run 2 featured a bigram weight of '1.8' and trigram weight of '2.5'. Run 3 featured a bigram weight of '1.8' and trigram weight of '2.5' along with lemmatization.

The precision we got for each domain is very high. Although we attained a much lower recall score, this was mostly due to the fact that we returned only 20 results for each document. Increasing the number of terms our model would return would drastically increase Recall. We further noticed that Recall is often document specific. Low Recall was observed in Computer Science and Bio-Chemistry which both featured much longer documents. We came to a conclusion that adjusting the number of results produced by our model to be a function of the length of the document would be a great idea.

Across the many domains, our Law Domain results were of extremely high quality and showed that both high precision and recall can be attained by our model.

Observing our results for various runs, we can see varied results for each domain. We notice that weights for bigrams and trigrams can be changed according to the domain in order to attain the best results. Domain Knowledge will help in particular, as a domain specialist can identify whether a specific domain will be having more bigrams or trigrams than unigrams so that there weights can be increased.

One particularly interesting thing to note was that the longer papers did much better when they were preprocessed with a lemmatizer. In domains like law and communication where we observed that words with same roots are used in different places differently, we see a negative impact of lemmatisation.

6 Future Work

We have evaluated the model for English, but the model can be made to work with pretty much any language with very minor modifications. Since we are considering the words as nodes and using a graph based approach to assign weights to these nodes, we are not concerned with what the word means or represents. We're only interested in its node weight. This allows our model to work with pretty much any language, given that tools for preprocessing text in that language are available. Restrictions can be made on number of letters for a specific language. Furthermore, transliteration can help to improve our scores. Transliterating the documents can help the results if a script of a language is unknown to the system. It will help to clearly distinguish between similar looking words. Further study of bigrams and trigrams weight in a specific language will also help the model. Ideally, the model should be able to learn the factor by which the bigram and trigram node weights need to be bumped. These bumping factors will not be same across different domains or even different documents, since it is not a reasonable assumptions that different documents will have a similar distribution of multi word terms.

References

Steven Bird. 2006. NLTK: The Natural Language Toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 69–72, Sydney, Australia. Association for Computational Linguistics.

Thorsten Brants. 2002. Tnt: A statistical part-of-speech tagger. *ANLP*.

Loukas Georgiadis, Giuseppe Italiano, Luigi Laura, and Nikos Parotsidis. 2014. 2-vertex connectivity in directed graphs.

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632.

Christopher D. Manning. 2011.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Rada Mihalcea, Paul Tarau, and Elizabeth Figa. 2004. PageRank on semantic networks, with application to word sense disambiguation. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 1126–1132, Geneva, Switzerland. COLING.

Larry Page, Sergey Brin, R. Motwani, and T. Winograd. 1998. The pagerank citation ranking: Bringing order to the web.

Author Index