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.