Proceedings of the 6th International Workshop on Computational Terminology (COMPUTERM 2020), pages 101–105
Language Resources and Evaluation Conference (LREC 2020), Marseille, 11–16 May 2020

TermEval 2020: RACAI's automatic term extraction system

Vasile Păiș, Radu Ion
Research Institute for Artificial Intelligence "Mihai Drăgănescu", Romanian Academy
CASA ACADEMIEI, 13 "Calea 13 Septembrie", Bucharest 050711, ROMANIA
{vasile, radu}@racai.ro

Abstract
This paper describes RACAI's automatic term extraction system, which participated in the TermEval 2020 shared task on English monolingual term extraction. We discuss the system architecture, some of the challenges that we faced as well as present our results in the English competition.

Keywords: automatic term extraction, ATE, natural language processing

1. Introduction

Automatic term extraction, also known as ATE, is a well-known task within the domain of natural language processing. Given a text (this can be either a fragment or an entire corpus), an automatic term extractor system will produce a list of terms (single or multiword expressions) characteristic for the domain of text.

Felber, in the "Terminology Manual" (Felber, 1984), defines a term as "any conventional symbol representing a concept defined in a subject field". Nevertheless, considering current practice in natural language processing tasks, it is not always possible to give a general definition applicable for the workings of a term extractor. One question is whether or not to include named entities as part of the identified terms. This problem is also raised by the organizers of the TermEval 2020 shared task, each system being evaluated twice, once including and once excluding named entities[1]. Furthermore, since named entity recognizers can be trained on many classes (such as diseases or chemicals for example), another potential question is what kinds of entities (if any) can be included as part of the identified terms. However, an agreement must be made that all identified terms must be specific to the domain of the analyzed text, regardless of inclusion or not of named entities. For example, in the shared task's provided training dataset, the named entity "United States Dressage Federation" is included as a term in the "equestrian" section.

The present paper presents our attempt at constructing an automatic term extraction system in the context of the TermEval 2020 shared task on monolingual term extraction (Rigouts Terryn et al., 2020). We start by presenting related research, then continue with the description of our system and finally present concluding remarks.

2. Related work

The usefulness of the term identification process is both in its own use, such as creation of document indices, and as a pre-processing step in other more advanced processes, such as machine translation. Furthermore, the output produced by an automatic system can be manually validated by a human user in order to remove irrelevant terms.

Traditional approaches for ATE (Kageura, 1998) make use of statistical features such as word frequency or "termhood" (degree of relatedness of a proposed term to the domain) metrics. Additionally, information such as part of speech can be used to further filter candidate terms. Term formalization attempts can be identified in the literature as early as e.g. 1996, when Frantzi and Ananiadou (1996) defined C-value as a basic measure of termhood, a principle we have also used in one of our algorithms. In this section, we will briefly mention the inner workings of some existing term extraction algorithms that we used in our term extraction system. For a detailed coverage of this rather vast sub-domain of NLP, the reader is referred to e.g. Pazienza et al. (2005) or the more recent Firoozeh et al. (2019).

TextRank (Mihalcea and Tarau, 2004) is a term extraction algorithm using a graph representation of the text in which each word is a node and an edge is created between words collocated within a certain window of words. Based on the number of links to each node a score is computed similar to the PageRank algorithm (Brin and Page, 1998). Further filtering is performed based on the part of speech of the words. The graph is created based on single words. However, as the last step of the algorithm a reconstruction of multi-word terms is performed if multiple single word terms are collocated in the sentence.

RAKE, an acronym for Rapid Automatic Keyword Extraction (Rose et al., 2010), combines graph measures such as the degree (number of connected edges) with statistical measures such as word frequency. Furthermore, RAKE uses a strategy similar to TextRank for combining single words that occur together at least twice into a multi-word term. An interesting idea deriving from the RAKE paper is the importance of the stop words list used. In this context, it is mentioned that FOX (Fox, 1989) stop list produces an increase in the F1 score for the RAKE algorithm. An improvement over the initial RAKE algorithm is described in Gupta et al. (2016).

Campos et al. (2020) present YAKE, which makes use of statistical features. According to their analysis[2] it is comparable or even better in some cases to previous state-of-the-art methods. In the HAMLET system (Rigouts Terryn et al., 2019) a number of 152 features are computed on each candidate term and a binary decision tree classifier is trained. Candidates are determined based on their part of speech, but the patterns of occurrence are determined automatically based on training data.

[1] https://termeval.ugent.be/task-evaluation/

[2] https://github.com/LIAAD/yake

3. Dataset and basic processing

The dataset proposed for the TermEval task is described in detail in the task paper (Rigouts Terryn et al., 2020). However, several aspects must be mentioned. It is comprised of 4 domains: wind energy ('wind'), corruption ('corp'), horse dressage ('equi') and heart failure ('hf'). The first 3 domains were provided with annotations for training purposes, while the heart failure domain was used for testing. All the domains were made available in English, French and Dutch.

For the purposes of our experiments, we focused on the English version of the corpus. However, we tried to keep our algorithms independent of the actual language being used. Towards this end, we used only resources normally available for many languages, such as annotations and stop words, and did not create any rules or patterns specific to the English language.

One of the primary processing operations was to annotate the corpus with part-of-speech and lemma information. For this purpose, we used Stanford CoreNLP (Manning et al., 2014). Furthermore, we precomputed statistical indicators based on the corpus, such as n-gram frequency, document frequency and letters used (in some cases terms contained non-English letters). Statistics were computed for both the corpus and the provided training annotations.

Unfortunately, the corpus is not balanced with respect to the different domains. Therefore, some statistical indicators may be less meaningful. For example, the corruption part of the corpus contains 12 annotated texts with an additional 12 texts provided without annotations. However, the equestrianism part contains 34 annotated text files and 55 unannotated documents. Furthermore, the evaluation section on heart failure contains 190 files. This seems to suggest that indicators like document frequency (the number of documents containing a certain word/expression) may be more meaningful for certain sections and less meaningful for others.

More statistics regarding the English domains of the corpus are presented in Table 1.

	equi	corp	wind	hf
Annotated files	34	12	5	190
Unannotated files	55	12	33	-
Unique lowercase tokens	6854	7958	21591	6092
Terms (without NE)	1155	927	1091	2361
Terms (with NE)	1575	1174	1534	2585

Table 1: Statistics regarding the English sections of the corpus

One of the characteristics specific only to the wind energy section of the corpus is the presence of mathematical formulas in some of the files. We could not identify an easy way to automatically remove them and did not want to manually perform this action. For example, "CP" is considered a term and it also appears in some formulas. Furthermore, there are lines of text presumably between formulas which look similar to a formula, like "CP ,max CT CTr" or full lines of text containing embedded formulas. Even more, the term "PCO2", indicated in the gold annotations, seems to only appear inside a formula ("PCO2 = TCO2 – HCO2 PCO2"). Therefore, in order to avoid removal of potentially useful portions of text, the files were used as they were provided.

Given these discrepancies between the different domain sub-corpora, it was our assumption, from the beginning, that different algorithms will obtain different results on each of the domains. Therefore, we started first by analyzing the results provided by known algorithms on the training parts of the corpus. These results are presented in Tables 2, 3, 4 and are compared against the provided annotations with named entities included. In these tables, the algorithm with the best F1 score in each section is marked in bold. The "1W" specification besides an algorithm denotes the score for single word terms.

In accordance with our previous observation, because of the imbalances between the different sections of the corpus, from Table 2 it can easily be seen that most of the algorithms perform better on the "equi" section and worse on the other sections. In some cases, there are even extreme differences. For example, the YAKE implementation gives on multi-word expressions an F1 score of 22.3 on the "equi" section and only 5.94 on the "wind" section. This is improved for single word expressions with 12% on the "equi" section and less then 3% for the other sections.

	P%	R%	F1%
TFIDF 1W	27.80	26.70	27.24
TFIDF	10.63	19.30	13.71
RAKE 1W	20.43	69.23	31.55
RAKE	15.39	65.97	24.95
YAKE 1W	39.31	31.00	34.66
YAKE	18.39	28.32	22.30
TRANK 1W	**29.21**	**42.76**	**34.71**
TRANK	26.86	25.27	26.04

Table 2: Precision, Recall, F1 measures for tested algorithms on the "equi" section

	P%	R%	F1%
TFIDF 1W	16.02	27.29	20.19
TFIDF	7.81	18.65	11.01
RAKE 1W	**16.80**	**75.30**	**27.47**
RAKE	12.95	65.08	21.60
YAKE 1W	30.94	8.57	13.42
YAKE	11.81	9.88	10.76
TRANK 1W	17.67	39.24	24.37
TRANK	17.05	18.40	17.70

Table 3: Precision, Recall, F1 measures for tested algorithms on the "corp" section

	P%	R%	F1%
TFIDF 1W	17.30	19.96	18.54
TFIDF	13.18	11.60	12.34
RAKE 1W	13.62	58.13	22.07
RAKE	**13.90**	**63.17**	**22.79**
YAKE 1W	64.29	3.18	6.06
YAKE	12.37	3.91	5.94
TRANK 1W	14.57	34.81	20.54
TRANK	14.11	13.62	13.86

Table 4: Precision, Recall, F1 measures for tested algorithms on the "wind" section

4. System Architecture

Looking at the above tables, two observations can be made: a) no single system performs best on all three sections; b) systems tend to balance precision and recall, but in extreme cases they prefer either precision (for example the YAKE method in "corp" and "wind" sections) or recall (for example the RAKE method).

A first idea that we explored was to implement a voting mechanism between the systems. However, the results presented only slight improvements. Without a complete and in-depth analysis, we concluded that each system was good at identifying certain terms (based on their pattern of occurrence) but performing badly for other terms. Therefore, we decided to extend the basic system and implement additional algorithms that would try to complement and extend the previous ones, by using new methods and finally use the same voting mechanism.

The first algorithm, PLEARN (from "pattern learn") is trying to identify patterns based on statistics computed on the train set annotations and their appearance in context. We used the following features: letters accepted in annotations (for example there is no term using ","), stop words accepted at start or end of a term (for example there is no term starting or ending with "and"), stop words accepted inside multi word terms, stop words accepted before or after a term (for example "and" usually is not contained within a term but rather it separates two distinct terms, thus appearing before or after a term), suffixes of words other than stop words present in terms (usually we tend to find nouns as terms, but we tried not to impose this condition, thus we only checked the suffixes of words).

For the purpose of the algorithm, all information was extracted automatically from the training set and no manual conditions or word lists were created. One immediate problem with the algorithm is that the training set did not provide the actual position of the term. Therefore, if the same word or multi-word expression was used both as term and as a non-term then the feature extraction part was not able to identify this case. Nevertheless, the algorithm was able to produce the good recall that we were expecting, presented in Table 5.

	P%	R%	F1%
Equi 1W	21.28	87.56	34.24
Equi	7.96	86.22	14.57
Corp 1W	15.61	91.43	26.66
Corp	4.85	89.86	9.19
Wind 1W	13.37	89.93	23.28
Wind	5.53	88.33	10.41

Table 5: Precision, Recall, F1 measures for the PLEARN algorithm on the training parts of the corpus

A second algorithm used a clustering approach, thus we'll refer to it as "CLUS" for the purposes of this paper. In this case we worked under the assumption that terms belonging to a particular domain will tend to cluster together because they will be related in meaning. In order to model this relation, we represented the words using word embeddings and used the cosine distance. For the clustering algorithm, we implemented a DBSCAN algorithm (Ester et al., 1996).

The input for the clustering algorithm was composed of the terms identified by the PLEARN algorithm. From these terms we kept only the single word terms. Furthermore, we decided to use an approach similar to the one used in TextRank to compose at the end multi-word terms based on the colocation of single word terms. This last operation was done in a post-processing step.

For the word embedding representation we considered necessary to use a model trained on a large enough corpus to allow for words to be used in different domains, including those of interest for this work. Therefore, we decided to use a word embeddings model trained on the Open American National Corpus (Ide, 2008). Furthermore, due to the relatively short time available for the task participation, we decided to use a pre-trained model[3]. Results are given in Table 6.

This algorithm already has a much better F1 score for single word terms then all the other algorithms tested. In the case of the "wind" section the F1 score is almost double (45.02%) then the best previous result (22.79%).

	P%	R%	F1%
Equi 1W	42.37	48.98	45.44
Equi	32.58	33.97	33.26
Corp 1W	44.14	28.49	34.62
Corp	36.46	12.27	18.36
Wind 1W	40.71	50.35	45.02
Wind	36.45	21.58	27.11

Table 6: Precision, Recall, F1 measures for the CLUS algorithm on the training parts of the corpus

Since the CLUS algorithm works on single word terms and only in the post-processing step combines them to create multi-word terms, we decided to work on a third algorithm that would work directly with multi-word expression candidates.

The third (and last) algorithm that we developed is called WEMBF (word embeddings filtered) and, as its name implies, uses the word embeddings vector representation of words to measure the termhood of each word. The algorithm executes the following steps:

1) Tokenizes and POS tags all text files of the specified domain of the corpus, using the NLTK Python library (Bird et al., 2009);

2) Extracts all NPs from the domain sub-corpus, using simple prenominal-nominal patterns, including all prepositional phrases headed by the preposition 'of', which are almost always attached to the previous NP. Furthermore, it deletes any determiners that start NPs and removes URLs, emails, numbers and other entities considered to be irrelevant for the term extraction task;

3) For each content word (i.e. nouns, adjectives, adverbs and verbs) of each NP, computes a cosine distance between two word embeddings vectors. The first vector is obtained from training on a "general"-domain corpus containing news, literature, sports, etc., being careful *not to include* texts from the domain of interest. The second vector is obtained from training only on the domain of interest (e.g. 'wind');

[3] https://data.world/jaredfern/oanc-word-embeddings

4) Score each NP by averaging the previously computed cosine distance of its member content words.

Step 4 of the WEMBF algorithm gives us a preliminary term list on the assumption that the larger the cosine distance of the general and domain word embeddings vectors is, the more likely is that the word is a term in the domain of interest. However, the obtained list contains too many NPs which makes it perform poorly in terms of precision. Thus, we decided to remove some term NPs from this initial list, using the following filters:

a) Only keep NPs which appear (are embedded) in other NPs from the preliminary term list (Frantzi and Ananiadou, 1996). The number of occurrences (in other NPs) is kept for each surviving NP to be rescored later;

b) Remove all single-word terms that appear as head nouns in other NPs on the assumption that if they can be modified, they are too general to be kept as terms.

The termhood score of each NP in the final list is modified by multiplying the following indicators: the original score of the NP, the number of words in the NP, the number of NPs in which this NP appeared.

Thus, if an NP has more words, it appeared in many other NPs and its average cosine distance (between the general domain and the domain of interest) of its member content words is higher, the NP is more likely to be a term.

Results of the WEMBF term extraction algorithm are given in Table 8.

	P%	R%	F1%
Equi 1W	30.48	41.06	34.99
Equi	32.83	31.49	32.15
Corp 1W	15.42	52.79	23.86
Corp	16.50	36.80	22.78
Wind 1W	7.72	52.65	13.47
Wind	8.97	38.72	14.56

Table 8. Precision, Recall, F1 measures for the WEMBF algorithm on the training parts of the corpus

The WEMBF algorithm has a performance similar to the PLEARN algorithm for single words, even though with a more balanced precision and recall, but better performance for multi-word terms.

The final step in our approach was to construct an ensemble module that takes the annotations from different algorithms and combines them together via a voting scheme. This is presented schematically in Figure 1.

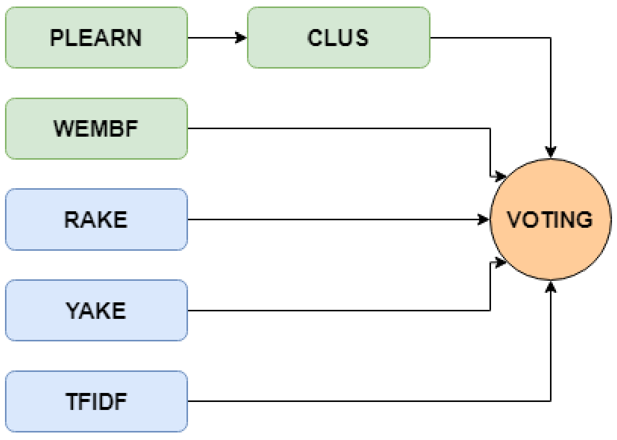

Figure 1. RACAI's term extraction system architecture that participated in TermEval 2020

Each algorithm is fed into the voting module, having one vote for the final result. An exception is in the case of PLEARN and CLUS algorithms which are linked together and thus constitute a single vote.

5. System evaluation

Once the test set annotations were released, we were able to evaluate our system, including all the other algorithms on the final data. When comparing this information with results based on the different training sections, we must keep in mind the peculiarities of each section of the corpus, as presented in Table 1 above. Evaluation results on the "heart failure" section are presented in Table 9.

Our CLUS algorithm performed best on the single word terms giving an F1 score of 53.48 with balanced precision and recall. Furthermore, the PLEARN algorithm produced the best recall, which was to be expected since it was designed especially for this purpose. However, the final algorithm with the combination of all of them did perform better on the multi-word terms, this being reflected in the final F1 score.

	P%	R%	F1%
TFIDF 1W	23.22	24.27	23.74
TFIDF	12.57	15.67	13.95
RAKE 1W	29.79	58.29	39.43
RAKE	19.48	58.88	29.27
YAKE 1W	28.93	62.22	39.50
YAKE	11.11	54.89	18.47
TRANK 1W	32.72	42.39	36.93
TRANK	28.93	22.28	25.17
PLEARN 1W	24.53	90.94	38.64
PLEARN	6.45	87.12	12.02
CLUS 1W	**49.11**	**58.72**	**53.48**
CLUS	41.17	35.82	38.31
WEMBF 1W	38.32	32.82	35.36
WEMBF	38.98	20.74	27.07
FINAL 1W	42.20	67.95	52.06
FINAL	42.40	40.27	41.31

Table 9. Precision, Recall, F1 measures of different algorithms on the evaluation set ("heart failure").

6. Conclusions and future work

This paper presented our system proposal[4] for the TermEval 2020 shared task. We started by investigating the performance of existing algorithms. Then went on and created three new algorithms: PLEARN, CLUS and WEMBF as described in section 4. Finally, we constructed an ensemble module, based on voting, which combined the results of all the algorithms in order to produce the final results. Evaluation on the "heart failure" dataset is presented in Table 9 above.

The approach behind the ACTER dataset, of building a term annotated corpus in multiple languages is very interesting and it was extremely helpful for building our automatic term extractor system. It is our hope that this or

[4] https://github.com/racai-ai/TermEval2020

a similar approach could be used for Romanian language as well. In this context, we envisage extending our term extractor to support Romanian language and further include it in the RELATE platform (Păiș et al., 2019) dedicated to processing Romanian language.

We managed to successfully use pre-trained word embeddings on a large corpus for our CLUS algorithm. This proves that transfer learning is a possibility that should be explored also in the field of term extraction. Therefore, amongst our future work we'll try to use the same approach for the Romanian language, by using pre-trained word embeddings (Păiș and Tufiș, 2018) on the Reference Corpus of Contemporary Romanian Language (CoRoLa) (Mititelu et al., 2018).

7. Acknowledgements

Part of this work was conducted in the context of the ReTeRom project. Part of this work was conducted in the context of the Marcell project.

8. Bibliographical References

Bird, S., Klein, E. and Loper, E. (2009). Natural Language Processing with Python --- Analyzing Text with the Natural Language Toolkit. O'Reilly Media; available online at http://www.nltk.org/book_1ed/.

Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7).

Campos, R., Mangaravite, V., Pasquali, A., Jatowt, A., Jorge, A., Nunes, C. and Jatowt, A. (2020). YAKE! Keyword Extraction from Single Documents using Multiple Local Features. In *Information Sciences Journal*. Elsevier, Vol 509, pp 257-289.

Ester, M., Kriegel, H. P., Sander, J. and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*,pp 226-231.

Felber, H. (1984). Terminology Manual. Paris: International Information Centre for Terminology.

Firoozeh, N., Nazarenko, A., Alizon, F. and Daille, B. (2019). Keyword extraction: Issues and methods. Natural Language Engineering, pages 1-33, Cambridge University Press.

Fox, C. (1989). A stop list for general text. *ACM SIGIR Forum*, vol. 24, pp. 19–21. ACM, New York, USA.

Frantzi, K. T. and Ananiadou, Sophia. (1996) Extracting Nested Collocations. In Proceedings of the 16th conference on Computational Linguistics - Volume 1, pages 41—46. Association for Computational Linguistics.

Gupta, S., Mittal, N., & Kumar, A. (2016). Rake-pmi automated keyphrase extraction: An unsupervised approach for automated extraction of keyphrases. In *Proceedings of the International Conference on Informatics and Analytics*, pp. 1-6.

Ide, N. (2008). The American National Corpus: Then, Now, and Tomorrow. In Michael Haugh, Kate Burridge, Jean Mulder and Pam Peters (eds.), *Selected Proceedings of the 2008 HCSNet Workshop on Designing the Australian National Corpus: Mustering Languages*, Cascadilla Proceedings Project, Sommerville, MA.

Kageura, K.; Umino, B. (1998). Methods of automatic term recognition. Terminology. 3(2):259-289.

Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J. and McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55-60.

Mihalcea, R., Tarau, P. (2004). TextRank: Bringing Order into Text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing EMNLP 2004*, pp 404-411.

Mititelu, B.V., Tufiș, D. and Irimia, E. (2018). The Reference Corpus of Contemporary Romanian Language (CoRoLa). In *Proceedings of the 11th Language Resources and Evaluation Conference – LREC'18,* Miyazaki, Japan, European Language Resources Association (ELRA).

Pazienza M.T., Pennacchiotti M. and Zanzotto F.M. (2005). Terminology Extraction: An Analysis of Linguistic and Statistical Approaches. In: Sirmakessis S. (eds) Knowledge Mining. Studies in Fuzziness and Soft Computing, vol 185. Springer, Berlin, Heidelberg

Păiș, V., Tufiș, D. (2018). Computing distributed representations of words using the COROLA corpus. In *Proceedings of the Romanian Academy*, Series A, Volume 19, Number 2/2018, pp. 403–409.

Păiș, V., Tufiș, D. and Ion, R. (2019). Integration of Romanian NLP tools into the RELATE platform. In *Proceedings of the International Conference on Linguistic Resources and Tools for Processing Romanian Language – CONSILR 2019*, pages 181-192.

Rigouts Terryn, A., Drouin, P., Hoste, V., & Lefever, E. (2020). TermEval 2020: Shared Task on Automatic Term Extraction Using the Annotated Corpora for Term Extraction Research (ACTER) Dataset. In *Proceedings of CompuTerm 2020*.

Rigouts Terryn, A., Drouin, P., Hoste, V., & Lefever, E. (2019). Analysing the Impact of Supervised Machine Learning on Automatic Term Extraction: HAMLET vs TermoStat. In *Proceedings of Recent Advances in Natural Language Processing – RANLP 2019*, pages 1012–1021, Varna, Bulgaria, Sep 2–4, 2019.

Rose, S., Engel, D., Cramer, N., & Cowley, W. (2010). Automatic keyword extraction from individual documents. *Text mining: applications and theory*, 1, 1-20.

Sparck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation, 28:11-21.