WING-NUS at SemEval-2017 Task 10: Keyphrase Identification and Classification as Joint Sequence Labeling

Animesh Prasad **Min-Yen Kan**
School of Computing, National University of Singapore
Computing 1, 13 Computing Drive, Singapore 11741
{animesh,kanmy}@comp.nus.edu.sg

Abstract

We describe an end-to-end pipeline processing approach for SemEval 2017's Task 10 to extract keyphrases and their relations from scientific publications. We jointly identify and classify keyphrases by modeling the subtasks as sequential labeling. Our system utilizes standard, surface-level features along with the adjacent word features, and performs conditional decoding on whole text to extract keyphrases.

We focus only on the identification and typing of keyphrases (Subtasks A and B, together referred as extraction), but provide an end-to-end system inclusive of keyphrase relation identification (Subtask C) for completeness. Our top performing configuration achieves an F_1 of 0.27 for the end-to-end keyphrase extraction and relation identification scenario on the final test data, and compares on par to other top ranked systems for keyphrase extraction. Our system outperforms other techniques that do not employ global decoding and hence do not account for dependencies between keyphrases. We believe this is crucial for keyphrase classification in the given context of scientific document mining.

1 Introduction

Keyphrases are often used for representing the salient concepts of a document. In scientific documents, keyphrase extraction is an important prerequisite task that feeds downstream tasks such as summarization, clustering and indexing, among others. As such, automatic keyphrase extraction has garnered attention and become a focal point for many researchers (Kim et al., 2010). Usually, the most common scenario of keyphrase extraction is to identify the keyphrases over the whole scientific document. Existing techniques in aforesaid setups use elaborate, hand-crafted features fed for selected candidate keyphrases to machine learning models such as support vector machines and multilayer perceptrons, to learn keyphrases (Kim et al., 2013). The scope of features vary from simple, surface-level features like character n-grams, token type, and part-of-speech tags – to features drawn from global statistics and lexical semantics, such as TF-IDF, keywordness, relation to document's logical structure (Nguyen and Kan, 2007).

However, the given task setup is inherently different as it requires to identify all the keyphrases of certain types (or classes – *Material*, *Process* and *Task*) over an excerpt of a scientific document. As inferred from our primary analysis of the training data, some of the crucial challenges of keyphrase extraction in this particular task setup are:

- Keyphrases occur more densely in the excerpts compared against the standard keyphrase extraction task where systems typically identify 5–25 keyphrases over an entire document;
- Keyphrases overlap significantly. For example "equally sized blocks" and "blocks" both need to be extracted as keyphrases of type *Materials*;
- Determining the keyphrase type depends on the context. For example "oxidation test" and "assessment of the corrosion condition" can potentially be either of *Task* or *Process*, depending on the context.

Considering these differences, we believe that sequence labeling based modeling is more suited than the standard, top K keyphrase extraction.

Such a model also easily extends to a joint approach for both extraction and classification, which we investigated.

2 Method

To accomplish Subtasks A and B, we deploy the Conditional Random Field (CRF) (Lafferty et al., 2001), a model capable of capturing conditional dependencies from sequential information. A CRF is a decoder which labels unseen sequences using the parameters learned from annotated examples to maximize conditional probability $p(y|x)$. We describe the inventory of features we provisioned (parenthesized notation used in Table 1).

- Token (F0): The token itself.
- Lower (F1): The token, lowercased.
- N-gram Prefix and Suffix (F2–F9): The initial to first four characters (prefix) and ultimate to last four characters (suffix) of the token.
- Part-of-Speech (F10): The part of the speech tag, as obtained from tagging of the complete sequence using the *nltk.punkt* tagger.
- Capitalization (F11): The orthographic case of the original token; taking one of four values: ALLCAPS, MixedCaps, Startcap or lowercase.
- Alpha/numeric? (F12–F13): Where the token is solely an integer, word, mixed or contains special characters.
- ASCII? (F14): Whether the token consist of non-ASCII special characters (the larger UTF-8/ISO Latin set, commonly used as symbols in scientific writing).
- Quoted? (F15): Whether the token exists between quotes.
- Hyphenated? (F16): Whether the token contains a hyphen.
- Math operators? (F17): Whether the token contains equality or inequality operators.
- Occurs in Title? (F18): Whether the token is present in the title of the article and is not a stop word in *nltk.stopword* for English.
- Output for Previous Token (O): The prediction for the previous token, a contextual label feature.

To achieve Subtask C, we placed minimal effort, choosing to build the end-to-end pipeline for sake of completeness. For Subtask C, we employed the *sklearn.ensemble*'s random forest classifier, with syntactic similarity features. The features are computed over the keyphrase pairs annotated with a relation as an instance for that relation and randomly generated pairs between keywords as an instance of no-relation. These features are the substring overlap between the two keyphrases, probabilistic and binary scores for one keyphrase being short/expanded form of other. We note that this approach is suboptimal as it does not incorporate any semantic information required for understanding relatedness.

3 Experiments

We use the CRF++ implementation[1]of CRFs. CRF++ takes as input a feature template file, describing contextual positions (like previous token, next two tokens) to incorporate component features from (F0 – F18, O). The template also can direct CRF++ to compute more complex feature–bigrams. Our final model's expanded feature list includes many of the surrounding tokens features [2].

Dataset. We participate in SemEval-2017 Task 10 on science information extraction (Augenstein et al., 2017) using the dataset consisting of 350 training samples (*Train*), 50 development samples (*Dev*) and 100 testing samples (*Test*). Each data sample is an excerpt of a scientific document. Unlike previous work creating similar benchmark dataset (Handschuh and QasemiZadeh, 2014), the dataset excerpt is taken out of a random section of the document.

Tasks. The excerpt requires its keyphrases to be identified (Subtask A), typed among one of three types: *Materials*, *Process* and *Task* (Subtask B), then finally followed by *Synonym-of* and *Hyponym-of* identification among the extracted keyphrases (Subtask C). Our work focuses specifically only on Subtasks A and B; our effort on Subtask C is minimal.

[1] https://taku910.github.io/crfpp/

[2] Code available at https://github.com/animeshprasad/science_ie

		Subtask A			Subtask B		
	Features	**Precision**	**Recall**	**F_1**	**Precision**	**Recall**	**F_1**
1.	All	0.55	0.38	0.45	0.51	0.32	**0.40**
2.	All - (F0–F1)	0.49	0.34	0.40	0.44	0.26	0.34
3.	All - (F2–F9)	0.53	0.33	0.40	0.46	0.25	0.33
4.	All - F10	0.55	0.36	0.43	0.50	0.30	0.37
5.	All - (F11–F17)	0.55	0.37	0.44	0.51	0.31	0.38
6.	All - F18	0.56	0.39	**0.46**	0.51	0.32	0.39
7.	All - O	0.30	0.39	0.34	0.26	0.32	0.29

Table 1: Model performance over different feature ablation, as evaluated on *Dev*. Best performance is **bolded.**

Evaluation. The designed evaluations test complete end-to-end pipeline of keyphrase identification, classification and relationship identification (Scenario 1); classification and relationship identification, given extracted keyphrases (Scenario 2); and relationship identification, given the extracted and classified keyphrases (Scenario 3). As the most intuitive and challenging scope, we only examine Scenario 1 in depth here. In Scenario 1 all the tasks are performed over the (noisy) system output of previous tasks whenever applicable.

3.1 Feature Ablation

To assess the importance of the component features, we perform ablation testing over the *Dev* (Table 1). We see that the setup of using all features is largely validated, with only a slight 0.01 F_1 loss on Subtask A alone. The decay in performance due to ablation is stable: showing that sequential dependencies on the previous output (O), the token identity, prefixes and suffixes matter the most. These results are expected and validate earlier sequence labeling work for parsing bibliographic reference strings (Councill et al., 2008). As we are dealing with document excerpts taken from random sections, and possibly due to the sparsity of the feature, title occurrence (F18, Row 6) played the least role in performance and could have been omitted. However, it helps in Subtask B, making the overall Subtask B perform slightly better (F18, Row 1). For simplicity, we use all features (Row 1) for our subsequent CRF models.

3.2 Joint Modeling versus Individual Experts

A limitation of CRFs is that they discount overlapping, hierarchical sequence labels, assigning the single most likely label sequence. As an example from *Train*, a *Material* occurring as substring of a *Task* could not be labeled correctly, as in "*[sequences of optimal walks of a growing length in weighted [digraph]$_{material}$]$_{task}$*". In contrast, separate CRF models for individual classes could label such structures; however, these separate models then have no contextual evidence of the existence of other labels in the sequence.

Setup	**Precision**	**Recall**	**F_1**
Joint	0.55	0.38	**0.45**
Unified	0.49	0.40	0.44

Table 2: Subtask A performance for *Joint* versus *Unified* models, as assessed on *Dev*. Best performance is **bolded.**

We test whether the aforementioned model (as in Table 1, referred to here as *Joint*) compares favorably to having individual expert models. We also test whether a single model only trained for identification (Subtask A) outperforms the *Joint* model which performs a relatively complex job. These additional models are:

- *Unified*: We collapse all three keyphrase types into a single type. This system acts as an expert for identifying keyphrases (Subtask A only).
- *Individual*: We use each of the three types of keyphrases to train an expert type-specific keyphrase extractors. This model can potentially predict overlapping (and sometimes conflicting) class instances.

We evaluate on the *Dev* with models trained on the *Train* (Tables 2 and 3). Interestingly, we can

Setup	Class	Precision	Recall	$\mathbf{F_1}$
Joint	Material	0.61	0.36	**0.45**
	Process	0.45	0.34	**0.39**
	Task	0.29	0.12	**0.17**
	Micro Avg.	0.51	0.32	**0.40**
Individual	Material	0.50	0.28	0.36
	Process	0.29	0.23	0.26
	Task	0.22	0.07	0.11
	Micro Avg.	0.37	0.22	0.28

Table 3: Subtask B performance for *Joint* versus *Individual* models, as assessed on *Dev*. Best performance is **bolded.**

Class	Precision	Recall	$\mathbf{F_1}$
Material	0.40	0.40	0.40 (−0.05)
Process	0.37	0.26	0.30 (−0.09)
Task	0.13	0.07	0.09 (−0.08)
Synonym-of	0.06	0.18	0.09
Hyponym-of	0.00	0.01	0.00
Micro Avg.	0.26	0.29	0.27

Table 4: Scores for evaluation on *Test*. Parenthetical numbers give differences from *Dev* performance.

see that the *Joint* model does better in both scenarios. In comparing the *Joint* versus the *Unified* model, we see that there is a useful signal in knowing the type of keyphrase. The *Unified* model benefits from having better annotation density, which could account for the higher recall. However, in the *Unified* model, the output feature (O) loses the information of the keyphrase type (if any) of the previous context. We conjecture that this loss of fidelity in the labels causes the drop in overall performance, particularly in precision. This is corroborated by our feature ablation experiments, where omitting the output feature causes the largest single drop in performance (Table 1, Row 7).

4 Official Run Results

We discuss the final reported system performance as officially recorded in the SemEval-2017 system runs for Task 10 blind *Test*. Table 4 reports the official results of our submitted run on the *Test*, trained over both *Train* and *Dev*.

We see a recognizable degradation of *Task* performance of over 50%, partially due to the skew in the distribution change between *Test* and *Dev* (Table 5). We note from the official run results, systems across the board experienced similar performance loss on *Task*.

Table 6 shows the breakdown of the Scenario 1 result for Subtasks A, B, and C. For Subtasks A and B in terms of precision, our model performed close to the best performing system, with a difference of 0.04. Our recall was significantly lower, by around 0.1. We believe this is caused by the systematic modeling error that our model incurs as it cannot deal with overlapping (nested) annotated types. The results are further worsened by the strict score calculation that discards partially extracted keyphrases as incorrect. This implies that every time a nested instance appears, the model loses at least one keyphrase for recall. This evaluation, also, to a great level contributes to the observed lower performance on *Task* (Table 3) on an average as compared to other classes. As apparent from the dataset, *Task* have more tokens and occasionally encompass smaller *Material* token sequences making it more susceptible to not match completely and hence incur low precision.

Dataset	Material	Process	Task
Dev	562(49%)	455(39%)	137(12%)
Test	904(44%)	954(46%)	193(9%)

Table 5: Type Count and Percentages.

Subtask	Precision	Recall	$\mathbf{F_1}$
A	0.51	0.42	0.46
B	0.37	0.31	0.33
C	0.03	0.10	0.04

Table 6: Official scores on Subtask evaluations.

5 Conclusion

We detail our approach in using the conditional random field to address the science information extraction task in SemEval-2017. We demonstrate that the CRF model, when applied jointly to the task of performing both identification and classification, outperforms sequential models for each task separately.

References

Isabelle Augenstein, Mrinal Kanti Das, Sebastian Riedel, Lakshmi Nair Vikraman, and Andrew McCallum. 2017. SemEval 2017 Task 10: ScienceIE - Extracting Keyphrases and Relations from Scientific Publications. In *Proceedings of the International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Vancouver, Canada.

Isaac G. Councill, C. Lee Giles, and Min-Yen Kan. 2008. ParsCit: An open-source CRF reference string parsing package. In *Language Resources and Evaluation Conference*.

Siegfried Handschuh and Behrang QasemiZadeh. 2014. The ACL RD-TEC: a dataset for benchmarking terminology extraction and classification in computational linguistics. In *COLING 2014: 4th International Workshop on Computational Terminology*.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pages 21–26.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2013. Automatic keyphrase extraction from scientific articles. *Language resources and evaluation* 47(3):723–742.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*. volume 1, pages 282–289.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *Proceedings of the 10th international conference on Asian digital libraries: looking back 10 years and forging new frontiers*. Springer-Verlag, pages 317–326.