EELECTION at SemEval-2017 Task 10: Ensemble of nEural Learners for kEyphrase ClassificaTION

Steffen Eger[†‡], **Erik-Lân Do Dinh**[†], **Ilia Kuznetsov**[†], **Masoud Kiaeeha**[†], **Iryna Gurevych**[†‡]
[†]Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
[‡]Ubiquitous Knowledge Processing Lab (UKP-DIPF)
German Institute for Educational Research and Educational Information
http://www.ukp.tu-darmstadt.de

Abstract

This paper describes our approach to the SemEval 2017 Task 10: "Extracting Keyphrases and Relations from Scientific Publications", specifically to Subtask (B): "Classification of identified keyphrases". We explored three different deep learning approaches: a character-level convolutional neural network (CNN), a stacked learner with an MLP meta-classifier, and an attention based Bi-LSTM. From these approaches, we created an ensemble of differently hyper-parameterized systems, achieving a micro-F_1-score of 0.63 on the test data. Our approach ranks 2nd (score of 1st placed system: 0.64) out of four according to this official score. However, we erroneously trained 2 out of 3 neural nets (the stacker and the CNN) on only roughly 15% of the full data, namely, the original development set. When trained on the full data (training+development), our ensemble has a micro-F_1-score of 0.69. Our code is available from https://github.com/UKPLab/semeval2017-scienceie.

1 Introduction

Although scientific experiments are often accompanied by vast amounts of structured data, full-text scientific publications still remain one of the main means for communicating academic knowledge. Given the dynamic nature of modern research and its ever-accelerating pace, it is crucial to automatically analyze new works in order to have a complete picture of advances in a given field.

Recently, some progress has been made in this direction for the fixed-domain use case[1]. However, creating a universal open-domain system still remains a challenge due to significant domain differences between articles originating from different fields of research. The SemEval 2017 Task 10: ScienceIE (Augenstein et al., 2017) promotes the multi-domain use case, providing source articles from three domains: Computer Science, Material Sciences and Physics. The task consists of three subtasks, namely (A) identification of keyphrases, (B) classifying them into broad domain-independent classes and (C) inferring relations between the identified keyphrases.

For example, for the input sentence 'The thermodynamics of copper-zinc alloys (brass) was subject of numerous investigations' the following output would be expected:

(A) 1. The thermodynamics of copper-zinc alloys
    2. copper-zinc alloys
    3. brass
(B) 1. `TASK`
    2. `MATERIAL`
    3. `MATERIAL`
(C) `synonym(2,3)`

Our submission focuses on (B) keyphrase classification given item boundaries. We avoid task-specific feature engineering, which would potentially render the system domain-dependent. Instead, we build an ensemble of several deep learning classifiers detailed in §3, whose inputs are word embeddings learned from general domains.

2 Task and Data

In the annotation scheme proposed by the task organizers, keyphrases denoting a scientific model, algorithm or process should be classified as ***PROCESS*** (P), which also comprises methods (e.g. 'backpropagation'), physical equipment (e.g. 'plasmatic nanosensors', 'electron microscope') and tools (e.g. 'MATLAB'). ***TASK*** (T) contains

[1] E.g. BioNLP: http://2016.bionlp-st.org/

Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017), pages 942–946,
Vancouver, Canada, August 3 - 4, 2017.

concrete research tasks (e.g. 'powder processing', 'dependency parsing') and research areas (e.g. 'machine learning'), while *MATERIAL* (M) includes physical materials (e.g. 'iron', 'nanotube'), and corpora or datasets (e.g. 'the CoNLL-2003 NER corpus').

The corpus for the shared task consisted of 500 journal articles retrieved from ScienceDirect[2], evenly distributed among Computer Science, Material Sciences and Physics domains. It was split into three segments of 350 (training), 50 (development), and 100 (test) documents. The corpus used in subtask (B) contains paragraphs of those articles, annotated with spans of keyphrases. Table 1 shows the distribution of the classes M, T, and P in the data. We note that class T is underrepresented and makes up less than 16% of all instances.

	Material	Process	Task
Train+Dev	40%	44%	16%
Test	44%	47%	9%

Table 1: Class distribution in the datasets.

Inter-annotator agreement for the dataset was published to be between 0.45 and 0.85 (Cohen's κ) (Augenstein et al., 2017). Reviewing similar annotation efforts (QasemiZadeh and Schumann, 2016) already shows that despite the seemingly simple annotation task, usually annotators do not reach high agreement neither on span of annotations nor the class assigned to each span[3].

3 Implemented Approaches

In this section, we describe the individual systems that form the basis of our experiments (see §4).

Our basic setup for all of our systems was as follows. For each keyphrase we extracted its **left context**, **right context** and the keyphrase itself (**center**). We represent each of the three contexts as the *concatenation* of their word tokens: to have fixed-size representations, we limit the left context to the ℓ previous tokens, the right context to the r following tokens and the center to the c initial tokens of the keyphrase. We consider ℓ, r and c as hyper-parameters of our modeling. If necessary, we pad up each respective context with 'empty' word tokens. We then map each token to a d-dimensional word embedding. The choices for word embeddings are described below. To summarize, we frame our classification problem as a mapping f_θ (θ represents model parameters) from concatenated word embeddings to one of the three classes *MATERIAL*, *PROCESS*, and *TASK*:

$$f_\theta : \mathbb{R}^{\ell \cdot d} \times \mathbb{R}^{c \cdot d} \times \mathbb{R}^{r \cdot d} \rightarrow \{\mathrm{M}, \mathrm{P}, \mathrm{T}\}.$$

Next, we describe the embeddings that we used and subsequently the machine learning models f_θ.

Word Embeddings

We experimented with three kinds of word embeddings. We use the popular Glove embeddings (Pennington et al., 2014) (6B) of dimensions 50, 100, and 300, which largely capture semantic information. Further we employ the more syntactically oriented 300-dimensional embeddings of Levy and Goldberg (2014), as well as the 300-dimensional embeddings of Komninos and Manandhar (2016), which are trained to predict both dependency- and standard window-based context.

Deep Learning models

Our first model is a character-level convolutional neural network (**char-CNN**) illustrated in Figure 1. This model (A) considers each of the three contexts (left, center, right) independently, representing them by a 100-dimensional vector as follows. Each character is represented by a 1-hot vector, which is then mapped to a 32-dimensional

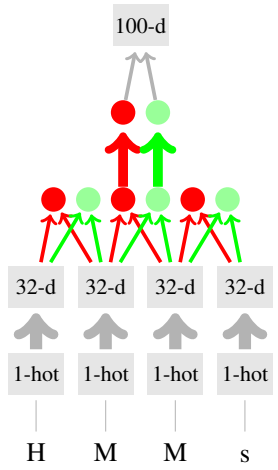

Figure 1: CNN. Each character is represented by a 1-hot vector, which is then mapped to a learned 32-d embedding vector. On these, m ($m = 2$ in the example) filters operate, which are combined to an m-dimensional vector via max-over-time-pooling. The output layer, with $\tanh$ activation, is 100-d and is fully connected with the m-dim layer that feeds into it. We represent the left context, right context, and center via the same illustrated CNN, and then concatenate the 100-d representations to a 300-d representation of the input.

[2] http://www.sciencedirect.com/

[3] F_1-scores ranging from 0.528 to 0.755 for span boundaries and from 0.471 to 0.635 for semantic categories.

embedding (not pre-trained, and updated during learning). Then m filters, each of size s, are applied on the embedding layer. Max-over-time pooling results in an m-dimensional layer which is fully connected with the 100-dimensional output layer, with $\tanh$ activation function. The 100-d representations of each context are then (B) concatenated, resulting in a 300-dimensional representation of the input. A final softmax layer predicts one of our three target classes. The hyper-parameters of this model—additional to ℓ, r, c mentioned above—are: number of filters m, filter size s, and a few others, such as the number of characters to consider in each context window.

Our second model, which operates on the token-level, is a "**stacked learner**". We take five *base classifiers* from scikit-learn (RandomForestClassifier with two different parameterizations; ExtraTreesClassifier with two different parameterizations; and XGBClassifier), and train them repeatedly on 90% of the training data, extracting their predictions on the remaining 10%. This process is iterated 10 times, in a cross-validation manner, so that we have a complete sample of predictions of the base classifiers on the training data. We then use a multi-layer perceptron (MLP) as a *meta-classifier* that is trained to combine the predictions of the base classifiers into a final output prediction. The MLP is trained for 100 epochs and the model with best performance on a 10% development set is chosen as the model to apply to unseen test data.

Our third model (Figure 2), also operating on the token level, is an attention based Bi-directional Long Short-Term Memory network (**AB-LSTM**)[4]. After loading pre-trained word embeddings, we apply 4 convolutional layers with filter sizes 2, 3, 5 and 7, followed by max-over-time-pooling. We concatenate the respective vectors to create an *attention vector*. The forward and backward LSTM layers (64-dimensional) are supplied with the pre-trained embeddings and the computed attention vector. Their output is concatenated and, after applying dropout of 0.5, is used by the final softmax layer to predict the label probabilities.

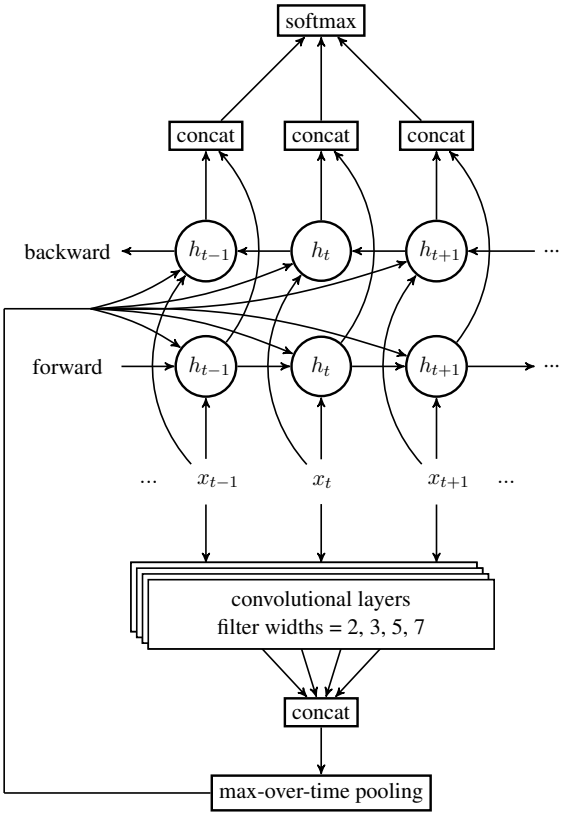

Figure 2: Bi-LSTM with attention. Pre-trained word embeddings x_t are fed to an ensemble of CNN layers with 4 different filter widths. For each timestep the outputs are concatenated and we employ max-over-time pooling. The resulting attention vector is supplied to the nodes in the forward and backward LSTM layers. The output of both LSTM layers is concatenated to a 128-dim vector, which is fed to the final softmax layer.

4 Submitted Systems

We set the c hyper-parameter to 4, and draw left and right context length hyper-parameters ℓ, r ($\ell = r$) from a discrete uniform distribution over the multi-set $\{1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5\}$.

Performance measure was micro-F_1 as computed by the task's evaluation script.[5] Table 2 shows average, maximum, and minimum performances of the systems we experimented with. We indicate the 'incorrect' systems (those trained on only the dev set) with a star. We tested 56 different CNNs—hyper-parameters randomly drawn from specific ranges; e.g., we draw the number of filters m from a normal distribution $\mathcal{N}(\mu = 250, \sigma = 50)$—90 different stackers, and 20 different AB-LSTMs. Our three submitted systems were simple majority votes of (1) the 90 stackers, (2) the 90 stackers and 56 CNNs, (3) the 90 stackers, 56 CNNs and 20 AB-LSTMs. Overall, majority voting is considerably better than the mean performances of each system.

[4] Code was adapted from https://github.com/codekansas/keras-language-modeling

[5] We report results without the "rel" flag, i.e., corresponding to the column "Overall" in Augenstein et al. (2017), Table 4. Setting "rel" leads to consistently higher results. E.g., with this flag, we have 72% micro-F_1 for our best ensemble (corresponding to column "B" in Augenstein et al. (2017), Table 4), rather than 69% as reported in our Table 2.

	Mean	Max	Min
CNN	58.32*/64.08	61*/65	54*/60
Stacker	61.57*/67.11	64*/68	59*/65
AB-LSTM	59.12	64	56
Majority	63*/69	63*/69	62*/68

Table 2: Micro-F_1 results in % for our systems.

For the stacker, the Komninos embeddings worked consistently best, with an average F_1-score of 63.83%. Levy embeddings were second (62.50), followed by Glove embeddings of size 50 (61%), size 300 (60.80) and size 100 (59.50). We assume this is due to the Komninos embeddings being 'richest' in nature, capturing both semantic and syntactic information. However, with more training data (corrected results), mean performances as a function of embedding type are closer: 67.77 (Komninos), 67.61 (Levy), 67.38 (Glove-300), 66.88 (Glove-50), 65.77 (Glove-100). The AB-LSTM could not capitalize as much on the syntactic information, and performed best with the Glove embeddings, size 100 (60.35%), and worst with the Levy embeddings (57.80).

The char-level CNN and the stacker performed individually considerably better than the AB-LSTM. However, including the AB-LSTM in the ensemble slightly increased the majority F_1-score on both the M and T class, as Table 3 shows.

Ensemble	M	P	T
(1) Stackers	76	71	46
(2) Stackers+CNNs	76	72	46
(3) Stackers+CNNs+AB-LSTMs	77	72	47

Table 3: F_1 results in % across different classes.

Error analysis: Table 4 details that *TASK* is often confused with *PROCESS*, and—though less often—vice versa, leading to drastically lower F_1-score than for the other two classes. This mismatch is because *PROCESS* and *TASK* can describe similar concepts, resulting in rather subtle differences. E.g., looking at various 'analysis' instances, we find that some are labeled as *PROCESS* and others as *TASK* in the gold data. This holds even for a few seemingly very similar keyphrases ('XRD analysis', 'FACS analysis'). The ensemble has trouble labeling this correctly, tagging 6 of 17 'analysis' instances wrongly. Beyond further suspicious labelings in the data (e.g., 'nuclear fissions reactors' as Task), other cases could have been resolved by knowledge of syntax ('anionic *polymerization* of styrene' is a process, not a material) and/or POS tags, and by knowledge of common abbreviations such as 'PSD'.

We note that our submitted systems have the best F_1-score for the minority class *TASK* (45%*/47% vs. ≤28% for all other participants). Thus, our submission would have scored 1st using *macro*-F_1 (60.66*/65.33 vs. ≤56.66), even in the erroneous setting of much less training data.

		Prediction		
		Material	Process	Task
Gold	Material	710	194	0
	Process	218	708	28
	Task	22	105	67

Table 4: Stackers+CNNs+AB-LSTMs confusion matrix.

5 Conclusion

We present an ensemble-based keyphrase classification system which has achieved close-to-the-best results in the ScienceIE Subtask (B) while using only a fraction of the available training data. With the full training data, our approach ranks 1st. To avoid using expert features has been one of our priorities, but we believe that incorporating additional task-neutral information beyond words and word order would benefit the system performance.

We also experimented with document embeddings, created from additionally crawled ScienceDirect[6] articles. Even though the stacker described in §3 acting as a document classifier obtained a reasonably high accuracy of ∼87%, its predictions had little effect on the overall results.

Manual examination of system errors shows that using part-of-speech tags, syntactic relations and simple named entity recognition would very likely boost the performance of our systems.

Acknowledgments

This work has been supported by the Volkswagen Foundation, FAZIT, DIPF, KDSL, and the EU's Horizon 2020 research and innovation programme (H2020-EINFRA-2014-2) under grant agreement № 654021. It reflects only the authors' views and the EU is not liable for any use that may be made of the information contained therein.

[6] https://dev.elsevier.com/api_docs.html

References

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. Semeval 2017 task 10: Scienceie - extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, Vancouver, Canada, pages 544–553. http://www.aclweb.org/anthology/S17-2091.

Alexandros Komninos and Suresh Manandhar. 2016. Dependency Based Embeddings for Sentence Classification Tasks. In *Proceedings of NAACL-HLT '16*. ACL, San Diego, CA, USA, pages 1490–1500. http://www.aclweb.org/anthology/N16-1175.

Omer Levy and Yoav Goldberg. 2014. Dependency-Based Word Embeddings. In *Proceedings of ACL '14*. ACL, Baltimore, MD, USA, pages 302–308. http://aclweb.org/anthology/P/P14/P14-2050.pdf.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of EMNLP '14*. ACL, Doha, Qatar, pages 1532–1543.

Behrang QasemiZadeh and Anne-Kathrin Schumann. 2016. The ACL RD-TEC 2.0: A Language Resource for Evaluating Term Extraction and Entity Recognition Methods. In *Proceedings of LREC '16*. ELRA, Portorož, Slovenia, pages 1862–1868.