SemEval 2017 Task10: Extracting Keyphrases from Scientific Publications

Sagar Chaturvedi and Madhur Pandey
School of Computing, University of Utah
{sagachat@cs.utah.edu, madhur@cs.utah.edu}

Abstract

This paper describes our approach to two subtasks of the SemEval 2017 Task 10: Extracting Keyphrases and Relations from Scientific Publications, Subtask (A): Identification of keyphrases and Subtask (B): Classification of identified keyphrases. The task tackled here is mention-level identification (Named Entity Recognition) and classification of keyphrases (Named Entity Classification). We kept the subtask (C): "Relationship extraction between keyphrases" out of the scope of this project. We explored four different algorithms: CRF based sequence labeling approach, Simple recurrent neural network (Simple RNN), gated recurrent units (GRU) and LSTM cells based RNN. Applied on the pre-annotated dataset, Recurrent neural networks outperformed the CRF for subtask (B) while CRF performed better for subtask (A).

1. Introduction

The task ScienceIE at SemEval 2017 [6] deals with the automatic extraction of keyphrases from Computer Science, Material Sciences and Physics publications, as well as extracting types of keyphrases and relations between keyphrases. PROCESS, TASK and MATERIAL form the fundamental objects in scientific works. A typical question, researchers and practitioners more than often face is: Which paper addresses a specific PROCESS, TASK or MATERIAL or their relation? Scientific researchers often have to go through a huge literature to obtain the answers to this question.

Current tools like Google Scholar [14] and Semantic Scholar [15] focus on navigating author and citations graphs. The ability of these search engines is limited for Scientific publications and also, often the researchers do very abstract queries with very specific idea. So, answering their queries gets very difficult but it can be simplified significantly by extracting keyphrases, classifying them and finding their relations. This SemEval task is crucial in understanding which scientific publications describe which processes, tasks and materials. This process can be extended to creating recommender systems for readers and finding reviewers for submissions.

The dataset is 500 scientific documents annotated manually for Named entity tags and their start and end locations in the documents. After pre-processing, we get 55972 training and 19167 test samples. In this project -

- We experimented with two different types of algorithms: Conditional Random Fields(CRF) and Recurrent Neural Networks.
- We experimented with three different recurrent *CELLS*: Simple RNN, Long Short Term Memory (LSTM) and Gated recurrent Units (GRU).
- We used *BLIO* (Beginning, Last, Inside, Out) tagging for labeling the sequences.
- For subtask (A), the labels were B, L , I and O.
- For subtask (B), the labels were B/L/I-(Name of Tag) and O. E.g. B-PROCESS, I-TASK, L-TASK etc.
- For both the subtasks, we trained the CRF with 3-gram and 5-gram of training sequences with arbitrary features like PoSTags and Character Cases.
- For both the subtasks, we trained the RNN with 3-gram and 5-gram training samples but did not use any arbitrary features.

For subtask (A), best F1 score is .79 which was achieved with CRF using combination of 5-gram features and using both L1 and L2 regularizations. As per current leaderboard, this result will stand 7^{th} among all the submissions.

For subtask (B), best F1 score is .31 which was achieved with LSTM using combination of 3-gram features with 256 batch size and 10 epochs. As per current leaderboard, this result will stand 11^{th} among all the submissions. The details of our experiments and results can be found in later sections.

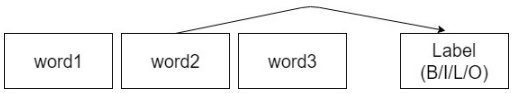


Figure 1: Example of 3-gram sample for CRF. Word2 is the current word, word1 and word3 are the words before and after it respectively. The label is the BILO tag of word2.

2. Related Work

A lot of work has been done before related to keyphrase boundary identification and classification and being done currently which has been mentioned in Augenstein et al. [6]. Kim et al. [11] have explained about majority of previous approaches and feature sets. As per Kim et al. [11], the majority of related work has been statistical that depend on features like Frequency based (Tf-Idf), linguistic (Suffix sequence), syntactic(length of phrase, count of stemming phrase), head noun heuristics and domain knowledge bases.

As mentioned by Augenstein et al. [6], a lot of teams attempt the tasks using RNNs, CRFs and their combinations. Some teams used Support vector Machines on top of engineered features like Tf-Idf, Noun phrase chunks and IDF weighted embeddings.

Zhang et al. [4] and Bhaskar et al. [1] discuss the CRF based approach using Word, PosTag, dependency and context features.

Wang et al. [9] proposed a Bidirectional LSTM that uses word embeddings. Chiu et al. [2] discuss a model that uses tune a Bidirectional LSTM on CNN extracted features. [5] explains a model that trains a CRF on top of output of a Bidirectional LSTM which is similar to idea of Collobert et al. [7].

3. Model

3.1. Data Pre-processing

The data of this task is available on the task homepage [6]. It consists of -

- TXT files - Contains the original text of the article
- ANN files - Contains the annotation start and end location, it's keyphrase type and value.

We perform following steps to convert the data for subtask (A) to BILO encoding:

- loop over each word in the TXT files and search it in their respective ANN file annotations.
- If the word is found in the ANN file annotations then we assign it -

  B - If the annotation starts with the word

  I - If word lies inside the annotation

  L - If the annotation ends with the word

  O - If the word is not among the annotations

For subtask (B), we attach Name of the tag (PROCESS, TASK, MATERIAL) with the BILO tags.

3.2. Conditional random fields

CRF is well suited for the Named entity recognition tasks because it models tagging decisions jointly and can relax the assumption of conditional independence of the observed data often used in generative approaches. It encodes a conditional probability distribution over the set of labels and tries to maximize the regularized log-likelihood function. Also, CRF avoids the label bias problem.

We used python library CRFSuite [3] as our CRF implementation.

3.3. Features for CRF

For both the subtasks, we train CRF with 3-gram and 5-gram training samples. For each word, we use it's features combined with the features of it's previous and next words, depending upon the n-gram we are using. Then in the next simple we slide the word window by one. Following are the features for each sample created for CRF -

- current word
- If 3-gram then next and previous word, if 5-gram then next 2 and previous 2 words
- If 3-gram then PoSTags of next and previous word, if 5-gram then those of next 2 and previous 2 words
- **Padding:** If the current word does not have previous or next words then it's sample is padded with either BGN (begin) or END.
- For the current word, first letter, all letters are capital or not
- Length of the current word
- Current word contains digit or not
- **Label:** The only change in the subtasks is the label.
  In subtask (A), Lable is one from the set B,I,L,O
  In (B) it is (B/I/L)-(Name of the tag) and O.
  Each sample uses the tag of current (middle) word as label. Refer the Figure 1.

E.g. if the text is "The Power X-Ray Oxidation helps in fabrication.", the 3-gram training samples for subtask (B) would look like -

- BGN BGN THE ..... O

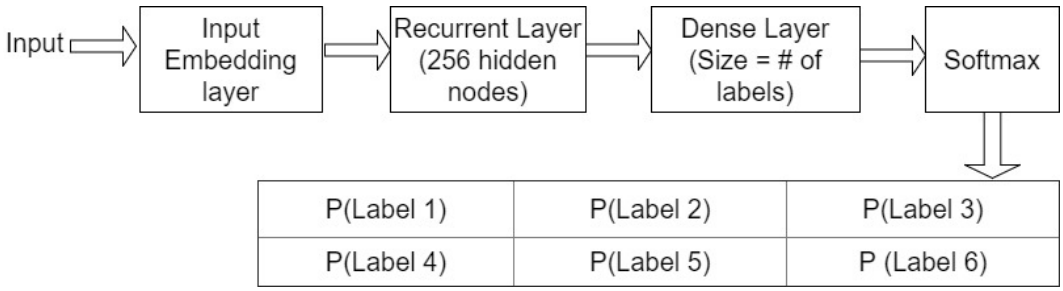


Figure 2: The layer architecture of our RNN models

- BGN THE Power ..... O
- THE Power X-Ray ..... B-PROCESS
- Power X-Ray Oxidation ..... I-PROCESS
- X-Ray Oxidation helps..... L-PROCESS

NLTK [13] was used for text pre-processing and PoS-Tagging.

3.4. Recurrent Neural Networks

Since recurrent nets use the result of previous node in the hidden layer of next node, they are very useful for sequence prediction tasks like Named entity recognition. We experimented with three different types of RNN cells:

- Simple RNN: Regular feedback RNN
- Long short term Memory (LSTM): Uses gated functions (input and forget gates) to deal with vanishing and exploding gradients. helps with preservation of long range dependencies
- Gated Recurrent Units (GRU): Controls the flow of information like LSTM but without using a memory unit. It just exposes the full hidden content.

We used Keras' [10] implementation of RNN on top of Tensorflow [16].

3.5. RNN architecture

Refer the Figure 2 for the layer architecture of our RNN. Following is the explanation:

- **Layer 1:** Input embedding layer of size 256 created using word vocabulary extracted from training and test data
- **Layer 2:** Recurrent layer with 256 nodes in the hidden layer
- **Layer 3:** A fully connected dense layer with nodes = number of labels
- **Layer 4:** A softmax layer that normalizes the output and gives a probability distribution over all the labels
- **Regularization:** Dropout = 0.5
- **Loss Function:** Since the output of our model is categorical, our RNN tries to minimize the *categorical cross-entropy* loss. The optimization method is ADAM [12].

3.6. Features for RNN

RNNs are trained with the the same feature scheme as CRF except that we do not use any arbitrary feature to train RNN i.e. we only use the words for creating 3/5-grams.

Vocabulary: To train the network, we create a vocabulary of words using all training and test sentences. Then using this vocabulary we create input embedding of each sample for RNN.

Labels: Labels are categorized into 1-Hot vector before submitting to RNN. Categorization means that if there are 10 labels then we create a vector of size 10 for each sample. Indices of the vector represent the labels. So for each sample, there will be 1 at the index of it's label and 0's at rest of the indices.

4. Experiments and Results

(CRF results are very extensive and it's difficult to include them all here. We have attached a sheet in our submission which contains all the results)

We performed various experiments with hyperparameters and compared CRF based approach with RNN based approaches. The table in Figure 3 shows our entire experiment space. The parameters setup for our models are:

- **CRF:** We experimented with all the combinations of below hyper-parameter settings:
  - Max Number of iterations = 50. Although we tried 150 and 200 iterations, they did not have a significant increase in accuracy of our model.
  - Regularizers: L1 and L2
  - Learning Hyperparameter C: [5, 1]

No	Task	Model	regularization	Activation	Loss function
1	subtaskA	crf	l2	NA	regularized log likelihood
2	subtaskA	crf	l1	NA	regularized log likelihood
3	subtaskB	crf	l1	NA	regularized log likelihood
4	subtaskB	crf	l2	NA	regularized log likelihood
5	subtaskB	rnn	dropout = 0.5	Softmax	categorical_crossentropy
6	subtaskB	lstm	dropout = 0.5	Softmax	categorical_crossentropy
7	subtaskB	gru	dropout = 0.5	Softmax	categorical_crossentropy
8	subtaskA	rnn	dropout = 0.5	Softmax	categorical_crossentropy
9	subtaskA	lstm	dropout = 0.5	Softmax	categorical_crossentropy
10	subtaskA	gru	dropout = 0.5	Softmax	categorical_crossentropy

Figure 3: Entire Experiment Space

- N-grams: [3, 5]

- **RNN:**
  - Cell types: Simple RNN, LSTM, GRU
  - Epochs: [10, 35]
  - N-gram: [3, 5]
  - Batch Size: [128, 256]
  - Input Embedding Length: 256
  - Hidden layer nodes: 256
  - Optimization Method: ADAM

4.1. Epoch Selection

We trained the Gated Recurrent Unit (GRU) with 100 epochs with 3-gram samples and a batch size of 128. We monitored the training process using TensorBoard monitoring tool. The Figure 4 shows the result for the same. It can be observed that after 35 epochs the training becomes stagnant. So we decided to train on maximum 35 epochs.

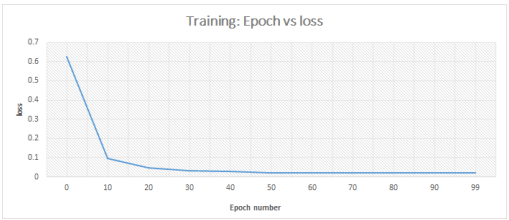


Figure 4: Loss vs Number of Epoch

4.2. Subtask A: Keyphrase Extraction

For our experiments, we only used 3-gram and 5-gram features. The details of model and features are available in section "Model". The table in Figure 5 shows the result of our RNN model for various combination of hyper parameters.

The overall performance of CRF was better for this task compared to the RNN. Figure 6 shows the best accuracy of each of the models used. However, the better performance of CRF may be due to the fact that we are training CRF with arbitrary features.

4.3. Subtask B: Classification of Keyphrases

Similar to Subtask A, we train our CRF and RNN based models on subtask B with the same features mentioned in Section 3. The results of various RNN models with various hyper-parameter combinations can be seen in the Figure 8.

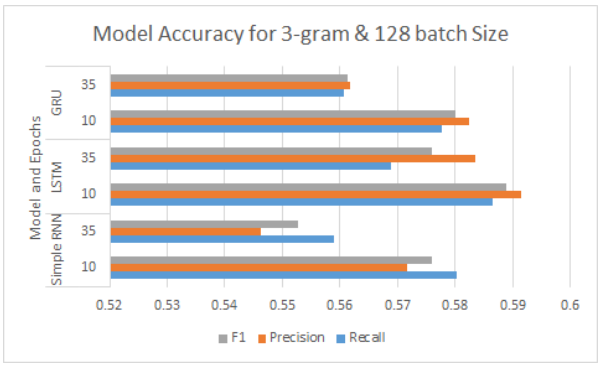
For this task, the overall performance of RNN was better than CRF. Figure 7 shows the best accuracy of each of the models used.

4.4. Hardware and Training Time

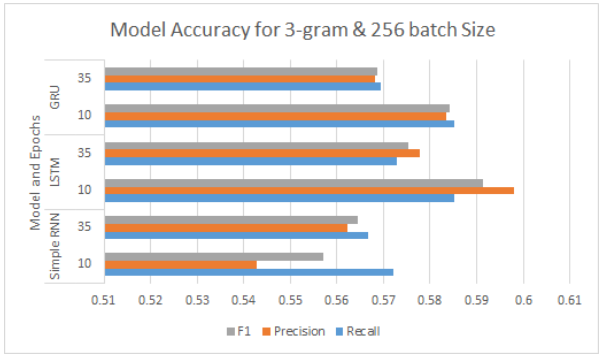
- We used a NVIDIA GeForce 940MX GPU with 2 GB memory for execution of Recurrent neural networks on 55972 training samples.
- For 35 epochs, average training time was ~10 minutes while for 10 epochs it was ~2.5 minutes
- Highest training time was ~18 minutes taken by LSTM with 35 epochs.

5. Analysis/Observations

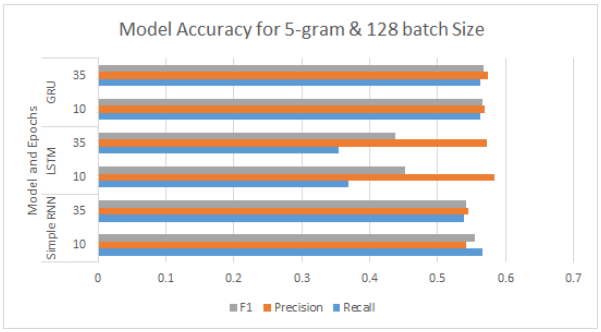
Following are our observations during the analysis of the results:



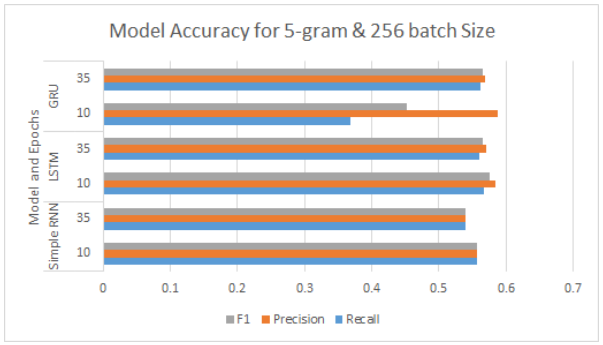
(a) 3gram - Batch Size 128



(b) 3gram - Batch Size 256



(c) 5gram - Batch Size 128



(d) 5gram - Batch Size 256

Figure 5: Comparision of various RNN on Subtask A

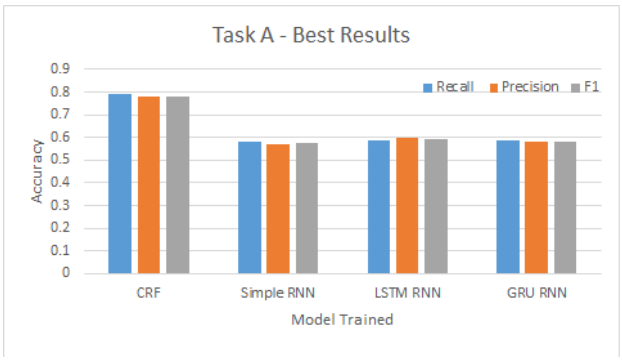


Figure 6: Task A- Best results achieved for each model

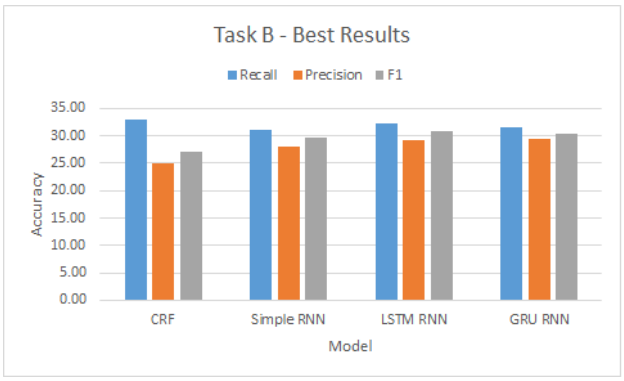
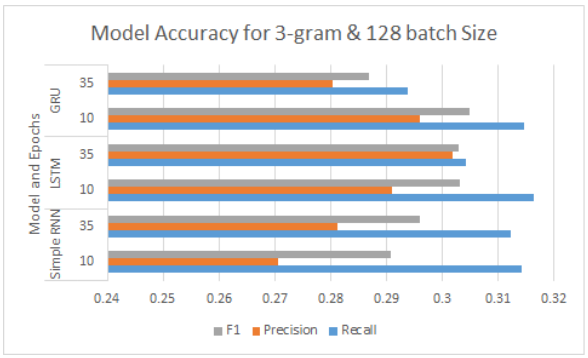
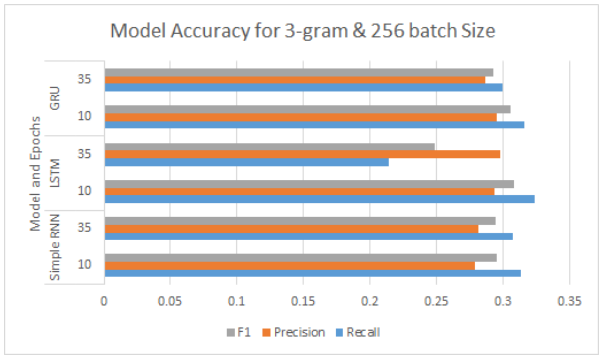


Figure 7: Task B - Best results achieved for each model

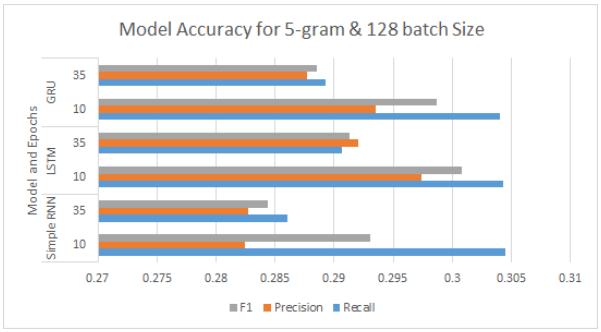
- We tried to train the RNN only for a single keyphrase type ”PROCESS”. It resulted in a F1 score of 0.5 which is 66 percent higher than the F1 score of the model trained on all the tags. This gives us insight that if we train models for all the tags separately and then combining them, we can achieve better accuracy.
- CRF achieves much higher recall for 5-gram samples than the 3-gram samples but RNNs do better with 5-grams.
- For subtask (A), the tag ’L’ achieves the best F1 score among all the BLIO tags. Least is for the tag ’I’.
- For subtask (B), the tag ’B-PROCESS’ achieves the best F1 score among all the BLIO-NER tags. Least is for the tag ’I-TASK’.
- CRF results in pretty high accuracy than RNN for subtask (A). We think there are two reasons behind it: First, CRF is trained with arbitrary features and sec-



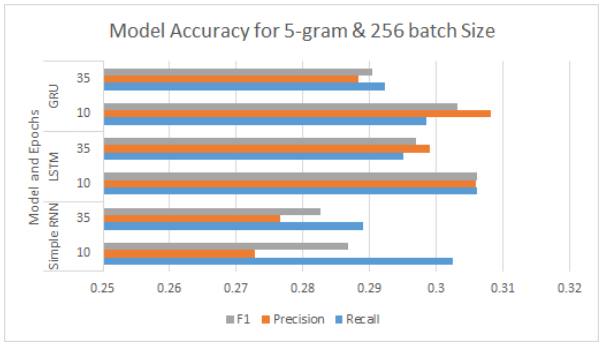
(a) 3gram - Batch Size 128



(b) 3gram - Batch Size 256



(c) 5gram - Batch Size 128



(d) 5gram - Batch Size 256

Figure 8: Comparision of various RNN on Subtask B

ond it's results represent for all the tags including "O".

- RNNs achieve more F1 score than CRF for subtask (B) but CRF scores the best recall.
- When we removed the words with no alpha-numeric characters from the training and test data, there was a slight improvement (1.5-2 percent) in the accuracy.
- Using Sigmoid activation function at place of softmax didn't affect the results much.
- As results display, 10 epochs yield better accuracy than 35 epochs in many cases.

6. Future Work

We can explore how training separate models for each named entity i.e. Process, Material and Task and combining them works out. We hope it will yield better results, refer to the first point of Analysis section for more details.

Also, we can use arbitrary features like POS tags, length and character cases with Recurrent Neural Networks. Using a Bidirectional RNN can be another direction also. Steffen et al., 2017 [8] used an ensemble of a character-level convolutional neural network (CNN), a stacked learner with an MLP meta-classifier, and an attention based Bi-LSTM on this task and got great results.

We can also experiment with training a CRF on top of RNN as mentioned by G. Lample et al. [5]. Another experiment can be trying optimization methods other than ADAM [12] like SGD, rmsprop (exponential decay in learning rate) and Adagrad (step decay in learning rate).

References

[1] P. Bhaskar. Keyphrase extraction in scientific articles: A supervised approach. *COLING*, pages 21–26, 2012.

[2] J. P. Chiu and E. Nichols. Named entity recognition with bidirectional lstm-cnns. *arXiv:1511.08308v5*, 2016.

[3] CRFSuite. Crf training toolkit.

[4] C. Z. et al. Automatic keyphrase extraction from documents using conditional random fields. *Journal of Computational Information Systems*, 4(3):1169–1180, 2008.

[5] G. L. et al. Neural architectures for named entity recognition. *NAACL-HLT*, pages 260–270, 2016.

[6] I. A. et al. Semeval 2017 task10: Scienceie - extracting keyphrases and relations from scientific publications. *arXiv:1704.02853v2*, 2017.

[7] R. C. et al. Natural language processing from scratch, 2011.

[8] S. E. et al. Eelection at semeval-2017 task 10: Ensemble of neural learners for keyphrase classification. *ACL*, pages 260–270, 2017.

[9] W. et al. A unified tagging solution: Bidirectional lstm recurrent neural network with word embedding. *arXiv:1511.00215v1*, 2015.

[10] keras. Recurrent neural network toolkit.

[11] S. N. Kim and M.-Y. Kan. Re-examining automatic keyphrase extraction approaches in scientific articles. *ACL-IJCNLP*, 2009.

[12] D. Kingma and J.Lei. Adam : A method for stochastic optimization, 2015.

[13] NLTK. Natural language toolkit.

[14] G. Scholar. Repository of scholarly articles.

[15] S. Scholar. Corpus of computer and neuroscience papers.

[16] TensorFlow. Open source library for machine intelligence.