SciTraction
Extracting Keyphrases and Relations from Scientific Publications
iNLP Final Submission Report

Team Number 31
Manav Chaudhary
Aneesh Chavan
Narayana Reddy

Introduction

The goal of this project is to automatically extract keyphrases from scientific publications and classify them based on their relevance to the topic. Keyphrases are important for several applications such as summarization, information retrieval, and document classification.

The project uses the **ScienceIE dataset** which consists of **500 journal articles** from the domains of Computer Science, Material Sciences, and Physics. The dataset is divided into training, development, and test sets. The training set consists of **350 documents, 50 are kept for development and 100 for testing.** The dataset is annotated with keyphrases and semantic relations between them.

The project aims to identify and classify keyphrases at mention-level and extract semantic relations between them. We decided to settle down for a BERT based approach, which is a state-of-the-art language model for natural language processing tasks.

Relevant Material Studied

For this project, we primarily followed the paper **"SemEval 2017 Task 10: ScienceIE - Extracting Keyphrases and Relations from Scientific Publications"** as a reference. The paper provides an overview of the ScienceIE task, which involves the identification and classification of keyphrases, as well as the extraction of semantic relations between them, in scientific publications.

The task is framed as a **sequence-to-sequence prediction task**, with three subtasks: mention-level keyphrase identification, mention-level keyphrase classification, and mention-level semantic relation extraction. Keyphrase types include PROCESS (including methods and equipment), TASK, and MATERIAL (including corpora and physical materials), and the relation types used are HYPONYM-OF and SYNONYM-OF. Only paragraphs likely to contain relations are annotated.

While successful systems for this task vary in their approaches, most of them use **RNNs, often in combination with CRFs and CNNs.** However, the system that performed best for evaluation scenario 1 in the paper used an SVM with a well-engineered lexical feature set.

In addition to the paper, we also studied various approaches for natural language processing and sequence labeling tasks, including the **use of pre-trained language models such as BERT** and the use of **attention mechanisms** for improved performance. We decided to incorporate these ideas into our own approach, using **modern BERT-based models** to achieve better accuracy in the task.

Method Outline

Our team found that the previously tried benchmarks for keyphrase extraction employed outdated methods that did not utilize the entire sentence context effectively. Therefore, we decided to explore the use of transformer-based models for keyphrase extraction.

We experimented with two different methods: an unsupervised approach and a supervised approach. The unsupervised method involved training a transformer-based language model on the training data and then extracting keyphrases based on the highest attention scores in the model. However, we found that this method did not perform as well as we had hoped, likely due to the lack of fine-tuning and context-specific training.

Therefore, we moved onto a supervised approach, using the BERT transformer model fine-tuned on our keyphrase extraction dataset. We utilized a modified version of the BERT model, replacing the last two layers with a classification layer to predict the presence or absence of each word in a given sentence as a keyphrase. This approach allowed us to effectively leverage the contextual information provided by the BERT model for keyphrase extraction.

Overall, our experiments with the BERT-based supervised approach proved to be the most effective method for keyphrase extraction, outperforming the previously attempted benchmarks.

Ideas, Experiments, Errors and Outcomes

At the start of our project, we explored various approaches to extract keyphrases and relations from scientific publications. We tried using the **WING-NUS** codebase for data parsing, but it did not have a full code repository available. Additionally, the TTI COIN approach, which used an **LSTM-based entity recognition** and relation extraction approach, was written in C++ and **not compatible** with our systems.

We also tried implementing an automatic keyphrase extraction approach **using GRU and RNN models in Keras.** However, we encountered **compilation errors** and had to spend time fixing them. While we could use this code as a boilerplate for our project, we decided to switch to **PyTorch**, which was more familiar to us, due to the course and assignments.

After researching the state-of-the-art approaches, we decided to use **BERT-based models** for our approach. We followed the SemEval 2017 Task 10 for reference but adopted our own novel approach using modern BERT models to achieve better accuracy.

We trained our model on the training dataset using **unsupervised learning techniques** and evaluated it on the development dataset. The initial results were promising, and we were able to extract keyphrases and relations with reasonable accuracy, with the **classification task achieving an accuracy score of 81%.**

However, we observed that the **choice of the BERT model had a significant impact** on the performance. We found that the "**all-mpnet-base-v2**" model worked best for our task.

In conclusion, using BERT-based models for keyphrase extraction and relation extraction proved to be a logical end to our project. While we faced some initial challenges in choosing the right approach and fixing code errors, we were able to achieve good results with our novel approach.

All the **relevant code files** have been added here:
https://github.com/LainWiredIn/SciTraction

Presentation Link:
https://docs.google.com/presentation/d/1re4AIUrSUDD8uREzULpQFtcGkXmo4ehSLNquF7-bQB0/edit?usp=sharing

Supplementary Information

Unsupervised method (no training)

- Use pre-trained BERT models from huggingfaces
- Extract a “document embedding” storing the “meaning” of the entire document by giving our pretrained BERT the entire query document
- Create a list of n-grams from the document, these are our candidate keyphrases
- Generate candidate embeddings from each n-gram
- Store different length n-grams separately (discussed in problems)
- **Compare each n-gram embedding with the document embedding**
- Ideally, the keyphrases should be the most representative phrases from the sentence
- Sort keyphrases according to their cosine distance between their embedding and the document embedding
- The top `n` keyphrases can be extracted from each list of d-length n-gram to get the best n keyphrases on length d
- We calculated our metrics by parsing each annotation document, and generating as many d-length keyphrases as there d-length ground truth keyphrases in our document.
- We calculated our precision, recall and F1 scores across the entire train and dev set. The results are as follows:

Unsupervised on train:

```
Final results:
Keyphrase length: 1
ic| precision: 0.28884615384615386
            recall: 0.40616549486208764
            f1_score: 0.33760395594515624
            avg_prec: 0.1911090505458214
            avg_reca: 0.3031597311041873

Keyphrase length: 2
ic| precision: 0.20575221238938052
            recall: 0.2590529247910863
            f1_score: 0.22934648581997533
            avg_prec: 0.1517551531225871
            avg_reca: 0.21671395804285717

Keyphrase length: 3
ic| precision: 0.1293800539083558
            recall: 0.14860681114551083
            f1_score: 0.13832853025936603
            avg_prec: 0.09298609483794663
            avg_reca: 0.13530666007517858

Keyphrase length: 4
ic| precision: 0.046169989506820566
            recall: 0.0484048404840484
            f1_score: 0.047261009667024706
            avg_prec: 0.03086773887744761
            avg_reca: 0.04670211126521805
```

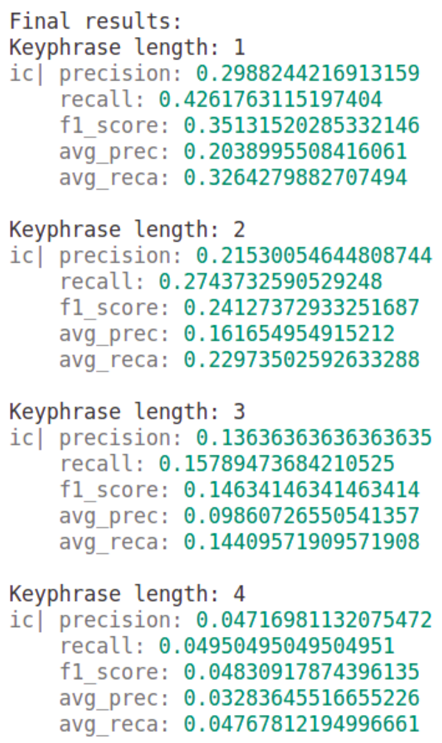
Unsupervised on dev:

```
Final results:
Keyphrase length: 1
ic| precision: 0.3385093167701863
            recall: 0.5117370892018779
            f1_score: 0.40747663551401864
            avg_prec: 0.24831215207028723
            avg_reca: 0.40110394403260496

Keyphrase length: 2
ic| precision: 0.23493975903614459
            recall: 0.30708661417322836
            f1_score: 0.2662116040955631
            avg_prec: 0.19657756207868102
            avg_reca: 0.29122918883948296

Keyphrase length: 3
ic| precision: 0.12658227848101267
            recall: 0.14492753623188406
            f1_score: 0.13513513513513514
            avg_prec: 0.09204623878536922
            avg_reca: 0.13048654244306418

Keyphrase length: 4
ic| precision: 0.07096774193548387
            recall: 0.0763888888888889
```

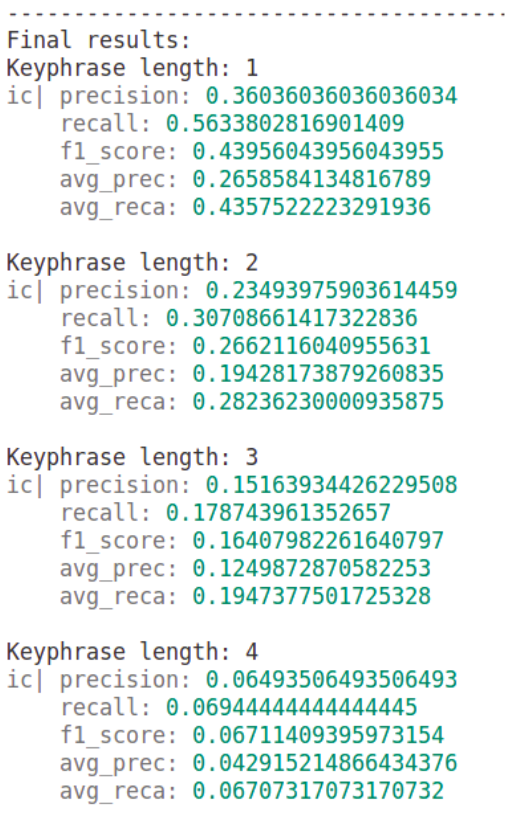
f1_score: 0.07357859531772575
avg_prec: 0.05000000000000001
avg_reca: 0.07398373983739837

While the above results are obtained using with "all-MiniLM-L12-v2" model The following results are obtained using the best model "all-mpnet-base-v2"

Unsupervised on **train**



```
Final results:
Keyphrase length: 1
ic| precision: 0.2988244216913159
    recall: 0.4261763115197404
    f1_score: 0.35131520285332146
    avg_prec: 0.2038995508416061
    avg_reca: 0.3264279882707494

Keyphrase length: 2
ic| precision: 0.21530054644808744
    recall: 0.2743732590529248
    f1_score: 0.24127372933251687
    avg_prec: 0.161654954915212
    avg_reca: 0.22973502592633288

Keyphrase length: 3
ic| precision: 0.13636363636363635
    recall: 0.15789473684210525
    f1_score: 0.14634146341463414
    avg_prec: 0.09860726550541357
    avg_reca: 0.14409571909571908

Keyphrase length: 4
ic| precision: 0.04716981132075472
    recall: 0.04950495049504951
    f1_score: 0.04830917874396135
    avg_prec: 0.03283645516655226
    avg_reca: 0.04767812194996661
```



```
--------------------------------------
Final results:
Keyphrase length: 1
ic| precision: 0.36036036036036034
    recall: 0.5633802816901409
    f1_score: 0.43956043956043955
    avg_prec: 0.2658584134816789
    avg_reca: 0.4357522223291936

Keyphrase length: 2
ic| precision: 0.23493975903614459
    recall: 0.30708661417322836
    f1_score: 0.2662116040955631
    avg_prec: 0.19428173879260835
    avg_reca: 0.28236230000935875

Keyphrase length: 3
ic| precision: 0.15163934426229508
    recall: 0.178743961352657
    f1_score: 0.16407982261640797
    avg_prec: 0.1249872870582253
    avg_reca: 0.1947377501725328

Keyphrase length: 4
ic| precision: 0.06493506493506493
    recall: 0.06944444444444445
    f1_score: 0.06711409395973154
    avg_prec: 0.042915214866434376
    avg_reca: 0.06707317073170732
```

Unsupervised on **dev**

Problems:

- Some keyphrases in the ann files are very large ~14-16 words. Extracting keyphrases of this length is meaningless as these are only included for later tasks in the same shared task
- Some special characters (greek alphabets/symbols) and named entities are very scientific and specific in nature. It is not likely that our model has embeddings for these terms available. Thus, our method may not work for sentences with any of these unseen characters. We do not have enough training data per document to compensate for this as well
- With only 350 training “documents” that are each just one paragraph long, we do not have a large enough amount of data to train a model generally enough to identify keywords well