INTRO TO NLP

Extracting Keyphrases and Relations from Scientific Publications

Team-44

2022201010 - Aman Khandelwal
2022201014 - Divyansh Negi
2022201028 - Sumonto Chatterjee

Objectives

1. Extracting keyphrases and relations from scientific publications is the process of automatically identifying important words or phrases and their relationships to summarize the main topics or concepts discussed in a publication.

2. This task is crucial as it helps researchers and readers to quickly understand the content and contributions of a publication, and enables more efficient information retrieval and literature search

Datasets

There are three benchmark datasets commonly used for automatic keyphrase extraction from scientific articles.

- The SemEval-2010 Task 5 dataset contains annotated keyphrases for 244 scientific articles from various domains.
- The ScienceIE 2017 dataset contains 500 journal articles evenly distributed among three domains and includes plain text, annotated standoff documents, and original full article text.
- The Inspec dataset contains 2,000 abstracts of scientific journal papers from Computer Science, each with two sets of keywords assigned: controlled keywords and uncontrolled keywords. These datasets are used to evaluate the performance of keyphrase extraction models.

Preprocessing

- We have used a preprocessed Inpec Dataset from Hugging face
- For Semval 2010/2017 we have used preprocessed dataset from IIITD's Midas lab's dataset

Models Used

Unsupervised Model

1. Objective: Implement BERT-based keyword extraction for identifying salient keywords from text.

2. Methodology:

- Data Selection: Choose a dataset, e.g., supervised machine learning document.
- Candidate Generation: Generate potential keywords/key phrases using CountVectorizer, customize length, and remove stop words.
- Embeddings: Utilize DistilBERT to convert document and candidates into numerical vectors.
- Cosine Similarity: Calculate similarity scores to identify relevant keywords/keyphrases.

3. Diversification Techniques:

- Max Sum Similarity: Select candidates maximizing similarity to the document while minimizing among candidates.
- MMR (Maximum Marginal Relevance): Choose new candidates similar to the document and dissimilar to already selected keywords, minimizing redundancy.

Unsupervised Model (Input File and Output)

Sample input doc

“

Keyphrase extraction is the task of automatically selecting a small set of phrases that best describe a given free text document. Supervised keyphrase extraction requires large amounts of labeled training data and generalizes very poorly outside the domain of the training data. At the same time, unsupervised systems have poor accuracy, and often do not generalize well, as they require the input document to belong to a larger corpus also given as input. Addressing these drawbacks, in this paper, we tackle keyphrase extraction from single documents with EmbedRank: a novel unsupervised method, that leverages sentence embeddings. EmbedRank achieves higher F-scores than graph-based state of the art systems on standard datasets and is suitable for real-time processing of large amounts of Web data. With EmbedRank, we also explicitly increase coverage and diversity among the selected keyphrases by introducing an embedding-based maximal marginal relevance (MMR) for new phrases. A user study including over 200 votes showed that, although reducing the phrases' semantic overlap leads to no gains in F-score, our high diversity selection is preferred by humans.

”

Output:


```
{
    "topK": [
        "keyphrase extraction requires",
        "paper tackle keyphrase",
        "documents embedrank novel",
        "supervised keyphrase extraction",
        "embedrank novel unsupervised"
    ],
    "MSS": [
        "extraction requires large",
        "novel unsupervised method",
        "200 votes showed",
        "diversity selection preferred",
        "web data embedrank"
    ],
    "MMR": [
        "embedrank novel unsupervised",
        "supervised keyphrase extraction",
        "200 votes showed",
        "paper tackle keyphrase",
        "extraction task automatically"
    ]
}
```

Neural Models (Supervised)

We experimented with BiLSTM-CRF models using GloVe embeddings and without GloVe embeddings, and tested them on different datasets which includes:

- Inspec
- SemEval-2017
- SemEval-2010

Architecture

```
BiLSTMCRF(
  (sent_vocab): Vocab()
  (tag_vocab): Vocab()
  (embedding): Embedding(3937, 300)
  (dropout): Dropout(p=0.5, inplace=False)
  (encoder): LSTM(300, 300, bidirectional=True)
  (hidden2emit_score): Linear(in_features=600, out_features=5, bias=True)
)
```

Inspec

With Glove

{'I-KEY': 4, 'B-KEY': 2, '<PAD>': 1, 'O': 3, '<UNK>': 0}

	precision	recall	f1-score	support
2	0.29	0.56	0.38	2517
3	0.95	0.90	0.92	60526
4	0.43	0.57	0.49	4249
accuracy			0.86	67292
macro avg	0.56	0.67	0.60	67292
weighted avg	0.90	0.86	0.88	67292

Without Glove

{'I-KEY': 4, 'B-KEY': 3, '<PAD>': 1, 'O': 2, '<UNK>': 0}

	precision	recall	f1-score	support
2	0.93	0.91	0.92	58168
3	0.38	0.54	0.44	3418
4	0.56	0.55	0.55	5706
accuracy			0.86	67292
macro avg	0.62	0.67	0.64	67292
weighted avg	0.87	0.86	0.87	67292

Semval 2017

With Glove

	precision	recall	f1-score	support
2	0.24	0.23	0.24	1285
3	0.00	0.00	0.00	0
4	0.96	0.83	0.89	20324
5	0.00	0.33	0.01	6
6	0.01	0.11	0.01	37
7	0.00	0.00	0.00	0
8	0.09	0.19	0.12	379
accuracy			0.78	22031
macro avg	0.19	0.24	0.18	22031
weighted avg	0.90	0.78	0.84	22031

Without Glove

Training Report (Taken from .ipynb file):

	precision	recall	f1-score	support
2	0.31	0.23	0.26	1670
3	0.14	0.22	0.18	532
4	0.01	0.62	0.01	8
5	0.95	0.84	0.89	19790
6	0.00	0.00	0.00	0
7	0.00	0.00	0.00	0
8	0.01	0.26	0.02	31
accuracy			0.78	22031
macro avg	0.20	0.31	0.19	22031
weighted avg	0.88	0.78	0.82	22031

Semval 2010

With Glove

Training Report (Taken from .ipynb file):

	precision	recall	f1-score	support
2	0.99	0.92	0.96	20420
3	0.06	0.39	0.10	151
4	0.07	0.54	0.13	98
accuracy			0.92	20669
macro avg	0.38	0.62	0.40	20669
weighted avg	0.98	0.92	0.95	20669

Without Glove

Training Report (Taken from .ipynb file):

	precision	recall	f1-score	support
2	0.16	0.34	0.22	476
3	0.16	0.38	0.22	301
4	0.98	0.93	0.95	19892
accuracy			0.91	20669
macro avg	0.43	0.55	0.47	20669
weighted avg	0.95	0.91	0.93	20669

BiLstm with Self Attention

Training Classification Report:

100%| | 1000/1000 [01:04<00:00, 15.61it/s]

	precision	recall	f1-score	support
0	0.87	0.98	0.92	123449
1	0.10	0.03	0.05	9633
2	0.13	0.00	0.01	8426
accuracy			0.86	141508
macro avg	0.37	0.34	0.33	141508
weighted avg	0.78	0.86	0.81	141508

Test Classification Report:

	precision	recall	f1-score	support
0	0.87	1.00	0.93	123449
1	0.00	0.00	0.00	9633
2	0.00	0.00	0.00	8426
accuracy			0.87	141508
macro avg	0.29	0.33	0.31	141508
weighted avg	0.76	0.87	0.81	141508

Pretrained Models (Supervised)

1. Keybert with keyphrase-vectorizers
2. Keyphrase-extraction-kbir
3. Keyphrase-generation-t5-small
4. Keyphrase-extraction-distilbert

Keybert with Keyphrase-Vectorizers

KeyphraseVectorizers package together with KeyBERT. The Keyphrase Vectorizers package extracts keyphrases with part-of-speech patterns from a collection of text documents and converts them into a document-keyphrase matrix.

```
class KeyphraseGenerationPipeline(Text2TextGenerationPipeline):
    def __init__(self, model, keyphrase_sep_token=";", *args, **kwargs):
        super().__init__(
            model=AutoModelForSeq2SeqLM.from_pretrained(model),
            tokenizer=AutoTokenizer.from_pretrained(model),
            *args,
            **kwargs
        )
        self.keyphrase_sep_token = keyphrase_sep_token

    def postprocess(self, model_outputs):
        results = super().postprocess(
            model_outputs=model_outputs
        )
        return [[keyphrase.strip() for keyphrase in result.get("generated_text").split(self.keyphrase_sep_token) if keyphrase != ""] for result in results]
```

Keybert with Keyphrase-Vectorizers Classification Report for Inspec Dataset

Validation Dataset

	precision	recall	f1-score	support
B	0.38	0.00	0.00	3950
I	0.00	0.00	0.00	4522
O	0.87	1.00	0.93	57819
accuracy			0.87	66291
macro avg	0.42	0.33	0.31	66291
weighted avg	0.78	0.87	0.81	66291

Test Dataset

	precision	recall	f1-score	support
B	0.56	0.38	0.46	4207
I	0.63	0.41	0.50	4875
O	0.91	0.96	0.94	58218
accuracy			0.88	67300
macro avg	0.70	0.59	0.63	67300
weighted avg	0.87	0.88	0.87	67300

Keyphrase-extraction kbir

This model uses KBIR as its base model and fine-tunes it on the Inspec dataset. KBIR or Keyphrase Boundary Infilling with Replacement is a pre-trained model which utilizes a multi-task learning setup for optimizing a combined loss of Masked Language Modeling (MLM), Keyphrase Boundary Infilling (KBI) andKeyphrase Replacement Classification (KRC)

```
class KeyphraseExtractionPipeline(TokenClassificationPipeline):
    def __init__(self, model, *args, **kwargs):
        super().__init__(
            model=AutoModelForTokenClassification.from_pretrained(model),
            tokenizer=AutoTokenizer.from_pretrained(model),
            *args,
            **kwargs
        )

    def postprocess(self, model_outputs):
        results = super().postprocess(
            model_outputs,
            aggregation_strategy=AggregationStrategy.SIMPLE,
        )
        return np.unique([result.get("word").strip() for result in results])
```

```
model_name = "ml6team/keyphrase-extraction-kbir-inspec"
extractor = KeyphraseExtractionPipeline(model=model_name)

keyphrases = extractor(test_document)
```

Keyphrase-extraction kbir Classification Report for Inspec Dataset

Validation Dataset

	precision	recall	f1-score	support
B	0.27	0.00	0.00	3950
I	0.00	0.00	0.00	4522
O	0.87	1.00	0.93	57819
accuracy			0.87	66291
macro avg	0.38	0.33	0.31	66291
weighted avg	0.78	0.87	0.81	66291

Test Dataset

	precision	recall	f1-score	support
B	0.63	0.65	0.64	4207
I	0.70	0.71	0.70	4875
O	0.95	0.95	0.95	58218
accuracy			0.91	67300
macro avg	0.76	0.77	0.76	67300
weighted avg	0.91	0.91	0.91	67300

Keyphrase-generation-T5-small

This model uses T5-small model as its base model and fine-tunes it on the Inspec dataset. Keyphrase generation transformers are fine-tuned as a text-to-text generation problem where the keyphrases are generated.

```
class KeyphraseGenerationPipeline(Text2TextGenerationPipeline):
    def __init__(self, model, keyphrase_sep_token=";", *args, **kwargs):
        super().__init__(
            model=AutoModelForSeq2SeqLM.from_pretrained(model),
            tokenizer=AutoTokenizer.from_pretrained(model),
            *args,
            **kwargs
        )
        self.keyphrase_sep_token = keyphrase_sep_token

    def postprocess(self, model_outputs):
        results = super().postprocess(
            model_outputs=model_outputs
        )
        return [[keyphrase.strip() for keyphrase in result.get("generated_text").split(self.keyphrase_sep_token) if keyphrase != ""] for result in results]


# Load pipeline
model_name = "ml6team/keyphrase-generation-t5-small-inspec"
generator = KeyphraseGenerationPipeline(model=model_name)

keyphrases = generator(test_document)

print(keyphrases)
```

Keyphrase-generation-T5-small Classification Report for Inspec Dataset

Validation Dataset

	precision	recall	f1-score	support
B	0.07	0.02	0.04	3950
I	1.00	0.00	0.00	4522
O	0.87	0.98	0.92	57819
accuracy			0.86	66291
macro avg	0.65	0.33	0.32	66291
weighted avg	0.83	0.86	0.81	66291

Test Dataset

	precision	recall	f1-score	support
B	0.40	0.34	0.37	4207
I	0.62	0.36	0.46	4875
O	0.91	0.95	0.93	58218
accuracy			0.87	67300
macro avg	0.64	0.55	0.58	67300
weighted avg	0.85	0.87	0.86	67300

Keyphrase-extraction-Distilbert

This model uses distilbert as its base model and fine-tunes it on the Inspec dataset.

```
class KeyphraseExtractionPipeline(TokenClassificationPipeline):
    def __init__(self, model, *args, **kwargs):
        super().__init__(
            model=AutoModelForTokenClassification.from_pretrained(model),
            tokenizer=AutoTokenizer.from_pretrained(model),
            *args,
            **kwargs
        )

    def postprocess(self, model_outputs):
        results = super().postprocess(
            model_outputs,
            aggregation_strategy=AggregationStrategy.FIRST,
        )
        return np.unique([result.get("word").strip() for result in results])
```

Keyphrase-extraction-Distilbert Classification Report for Inspec Dataset

Validation Dataset

	precision	recall	f1-score	support
B	0.10	0.00	0.00	3950
I	1.00	0.00	0.00	4522
O	0.87	1.00	0.93	57819
accuracy			0.87	66291
macro avg	0.66	0.33	0.31	66291
weighted avg	0.84	0.87	0.81	66291

Test Dataset

	precision	recall	f1-score	support
B	0.51	0.68	0.58	4207
I	0.62	0.70	0.66	4875
O	0.96	0.92	0.94	58218
accuracy			0.89	67300
macro avg	0.70	0.77	0.73	67300
weighted avg	0.90	0.89	0.90	67300

References

1. Keyphrase Extraction from Scholarly Articles as Sequence Labeling using Contextualized Embeddings (https://arxiv.org/pdf/1910.08840.pdf)
2. Keyphrase Extraction with BERT Transformers and Noun Phrases (https://towardsdatascience.com/enhancing-keybert-keyword-extraction-results with-keyphrasevectorizers-3796fa93f4db)
3. Learning Rich Representation of Keyphrases from Text (https://arxiv.org/pdf/2112.08547.pdf)
4. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (https://jmlr.org/papers/volume21/20-074/20-074.pdf)
5. Automatic Keyphrase Extraction: A Survey of the State of the Art (https://aclanthology.org/P14-1119.pdf)

Thank You