FWD 2021 summer report – Xiuhao Ding

- The Problem

  The ultimate goal is to find out the most efficient way to extract keywords in specific domain from the large corpus.

  According to our observations, there are many resources and methods for extracting keywords or domain relevance, but there is a lack of information for extracting keywords in specific domains.

  Therefore, we think about whether we can extract keywords first, and then combine with the domain relevance method to get the results we want.

- My Approaches

  At first, we evaluate and tried to run many efficient keyword extraction methods, such as Textrank, PhraseScope, TF-DCF, and TeKET. Because our goal is to extract keywords from a large corpus, not from a single article, and then combined with the overall consideration, we finally chose Autophrase as the keyword extraction method.

  For the domain relevance part, we choose the CFL/HiCFL from the paper "Measuring Fine-Grained Domain Relevance of Terms: A Hierarchical Core-Fringe Approach" to measure the relevance to certain domain. (Using query-input)

  After combing these two methods, we use the domain keyword lists (math_keys.csv and Keywords-Springer-83K-20210405.csv) from FWD-slack channel to calculate the precision rate for math/cs domain in order to evaluate the result.

- Progress

  Part 1: Some definitions

  Autophrase: a keyword extraction method (Input: large corpus. Output: key terms.)

  Important score: In the result of Autophrase, each key term has its own score related to its importance in the corpus. Here, we called it important score.

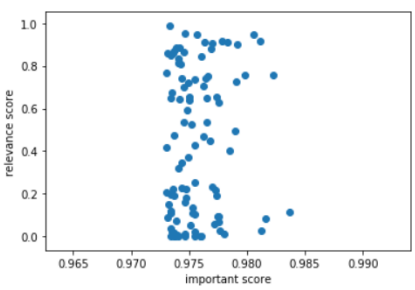
  Domain relevance measure: In this report, the phrases like "domain relevance measure" are refer to the CFL/HiCFL method. (Input: some query terms. Output: each term with the score how it is relevant to a certain domain)

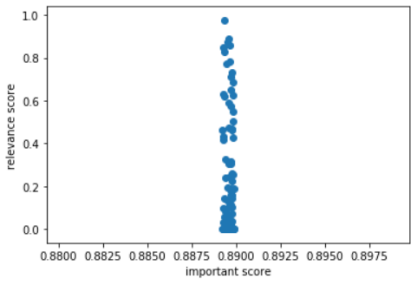
  Relevance score: The score in the result of CFL/HiCFL.

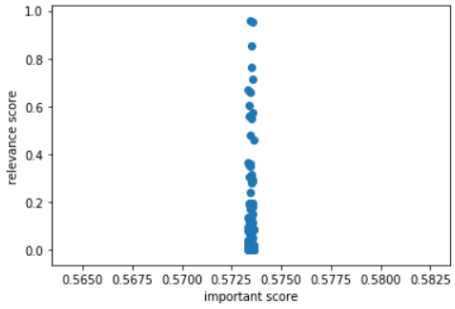
  Part2: The relationship between important score and relevance score

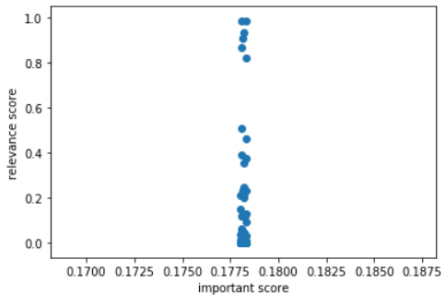
  We extract the CS-corpus from arXiv dataset, using it as the input for Autophrases. From the result of Autophrases, we have a list of important terms and their important scores. We use those important terms as the input query for domain relevance measuring and get the CS-relevance score for those terms.

  The graphs below show the relationship between important score and relevance score in different ranges.









From these four graphs, there are two discussions.

1. The terms has lower important score tends to have lower domain relevance score (the picture in the lower right corner has more terms with low relevance score). This is reasonable because the more critical the keywords in the CS corpus, the closer they are to the CS domain.

2. **It doesn't show a strong correlation between the two scores, so we can combine the two scores.**

Part3: Baseline

Step 1: Extract different topic corpus from arXiv dataset. (cs/math/phy corpus)

Step 2: Using the corpus from Step1 as the input for Autophrases, and get the result, which includes a list of important terms and their important scores.

Step 3: Using the list of important terms from Step2 as the input query for Domain-relevance measurement, choosing the certain domain to measure(cs/math/phy) and get the relevance score for each important term.

Step 4: Mix important scores and relevance scores with different coefficients, and then reorder all the terms based on the new mixing score.

Step 5: Extract terms in different levels (top1000/top5000/1000-2000/…) from the result of step4 and calculate the precision rate based on the keyword list in specific fields (CS/MATH).

Part4: Discussions in Step5 of the baseline

In the step5, we use math_keys.csv and Keywords-Springer-83K-20210405.csv from FWD-slack channel.

Below is using the 83k.csv as the example.

- How to integrate 83k.csv with our terms to calculate precision rate.

1. If simply use whether "our terms = the term from 83k", we may miss many terms that contain a part of 83K'terms or is contained by a term from 83k. (e.g. 'cache sizes' != 'cache size'(from 83k))

2. So firstly, we include the terms that are contained by a term from 83k. (e.g. 'architecture search' is contained by 'neural architecture search'(from 83k))

3. Secondly, we include the terms that contain a term from 83k. However, I observe that Null and some single words without too much meanings are included in 83k(e.g. 'learning'). After excluding these two parts, we include the terms that contain a term from 83k. (e.g. 'horn clause logic' contains 'horn clause' (from 83k)).

- Results

Section 1: Conclusion

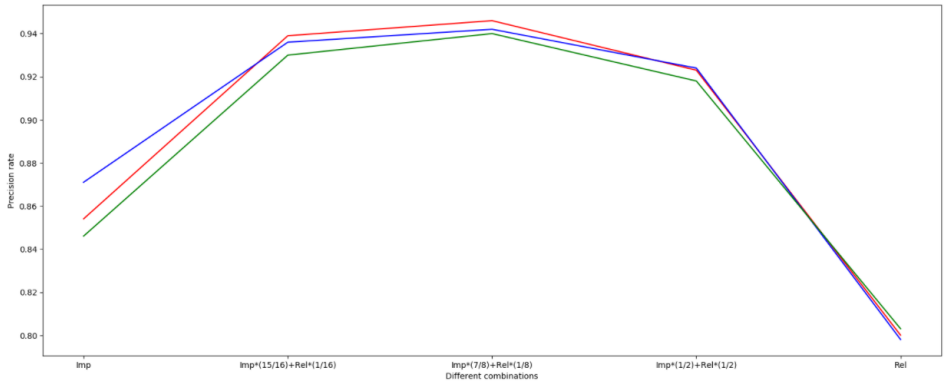
Restate the baseline: Choosing a corpus called **C** as the input for Autophrases, and get the important score called **Imp** for terms, using those terms as the input query for Domain-relevance measurement, and get the relevance score called **Rel**.

1. Only use important score or relevance score to rank terms will give low precision rate. Mixing these two scores will have a significant improvement.

2. If the relevance measuring domain is highly relevant to the topic of the initial corpus C (e.g. C is a CS corpus and the relevant score is calculated for CS domain), we should give more weights for the importance score (Imp), and rank terms using (important_score * (7/8) + relevance_score * (1/8)) will give the highest precision rate. **(Situation 1)**

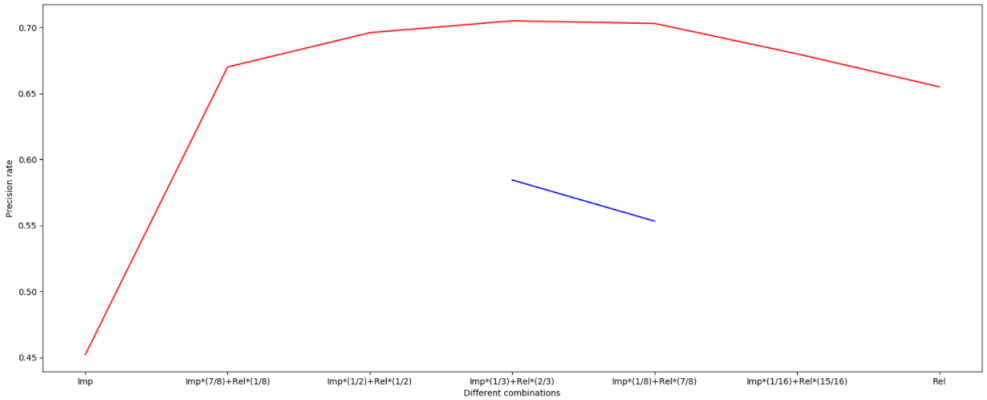
3. Otherwise, if the initial corpus C is not highly related to the relevance-measurement domain (e.g., C is a phy corpus, and the relevant score is calculated for Math or CS domain), we should give more weights for the relevance score (Rel) to earn higher precision rate. When we rank terms using (important_score * (1/3) + relevance_score * (2/3)), we will get overall highest precision rate. (Highest top5000 precision rate and relatively highest top1000 precision rate). **(Situation 2)**

Section 2: Some graph result for Situation 1 mentioned in Section1:

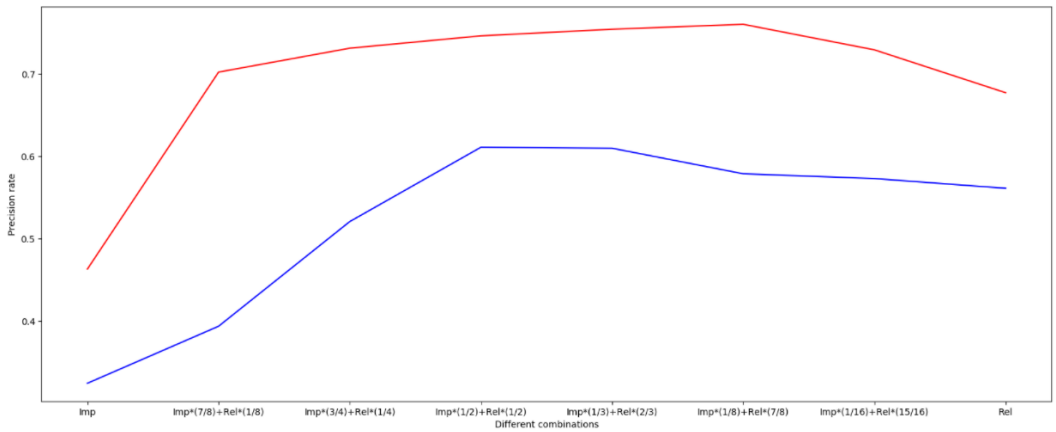


Top1000 precision rate for three instances of cs corpus keyphrases extraction with cs relevance score(precision rate based on CS keyword list)
other intervals have similar rank

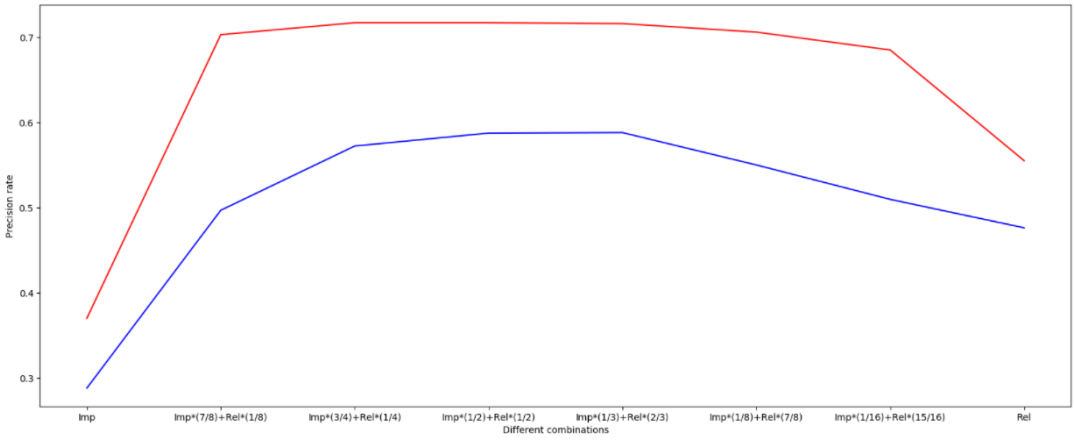
Section 3: Some graph result for Situation 2 mentioned in Section1:



Top1000(red)/5000(blue) precision rate for phy corpus keyphrases extraction with cs relevance score(precision rate based on CS keyword list)
other intervals have similar rank



Top1000(red)/5000(blue) precision rate for phy corpus keyphrases extraction with math relevance score(precision rate based on Math keyword list)
other intervals have similar rank



Top1000(red)/5000(blue) precision rate for cs corpus keyphrases extraction with math relevance score(precision rate based on Math keyword list)
other intervals have similar rank

Section 4: More discussions on Situation 1 mentioned in Section1

For the Math-case in Situation 1: Input: Math corpus – Process: Math domain relevance measurement – Evaluation: Math keyword precison rate.

- Similar results as before

1. Mixing two scores will have a significant improvement. Higher precision rate than only uses one of them.

2. Giving more weights for the important score will have higher precision rate since math-domain is highly related to the math corpus.

3. Using (important_score * (7/8) + relevance_score * (1/8)) to rank terms gives overall best performance.

Some results:

(important_score * (7/8) + relevance_score * (1/8)): top1000precision = 744/ 1000 - top5000 precision = 2939/5000

- Somewhat different

- Math corpus + Math relevance doesn't give high precision rate as CS corpus + CS relevance (precision rate over 90%).

- One reason can be: There are many terms in Math-abstract text include symbols

such as "$(k, \ell) $-pebble game" and "stationary R^d-valued Markov and Feller process (X_t)". Those terms may affect the process of keyword extraction. Also, in the result of keyword extraction, many terms including symbols have no meaning and will cause input errors, so I exclude the terms including symbols from the input of domain relevance measuring.

- Assessment

  Through the results of different situations, it is found that combining these two independent scores can extract keywords in specific fields more effectively.

- Future Plan

  In the above experiment, the keyword list appears in the last step. However, we can also explore the keyword list of other unpopular fields through our baseline. For example, input a corpus of an unpopular fields and extract the keyword list of this field through our baseline. But the premise of this is that our baseline must have high precision rate. The CS corpus-CS relevance gives us a very high precision rate, but in the math-case mentioned in the Section4 above, it does not have a high precision rate. In the future, we can try other corpus/domain that doesn't include many symbols as Math has.