CIS 430/530 Fall 2012 Final Project

Instructor: Ani Nenkova
TA: Kai Hong, Jessy Li, Achal Shah

Released: December 3, 2012
Due: 11:59PM December 19, 2012
Latest Submission time: December 23 at noon, 2012

Overview

For the final project you will implement several generic multi-document text summarization systems. Your task is to generate a 100-word summary for each input. We will use data from DUC 2004, which is one of the most popular test sets for this task [9]. This dataset consists of 50 inputs, each containing several news articles on the same topic. For evaluation we will use ROUGE to evaluate the produced summaries by comparing to four abstracts written by people on the same input. As before, you may work by yourself or work in group of two.

The project assignment requires implementing three extractive summarization systems as well as one summarizer of your choice.

Score for each task

Full Score: 100 (20% of the final grade).

- Three Basic Summarizers (55%):
  Centrality Summarizer, Topic Word Summarizer, LexRank Summarizer.
- Your Summarizer: 30% (Can be either Extractive or Abstractive, at least two NLP tools or resources must be used).
- Write-up: 15%.
  - Clear and complete write-ups with strong method and analysis sections will receive up to 20 points extra credit.
- The top 5 summarizers among all groups, if with Rouge-2 Recall score higher than 0.088, will get 20 points extra credit for this project. (Up to 40 points in total)

Submission CheckList

- Code for the three basic summarizers.
- Code for your summarizer.
- Summaries generated from the four summarizers (200 files in total).
- Write-up

Data

The Document Understanding Conference (DUC) is an annual evaluation challange for summarization, run by NIST. NIST assessors prepare the inputs for an unseen test set and manually evaluate the submissions of the participating teams. DUC 2004 is the most popular test set for generic multi-document summarization. The data from the evaluation is locally placed in `/home1/c/cis530/final_project`. This folder includes:

- input: The 50 input sets.
- models: Human summaries for the 50 inputs.
- baseline: Summaries generated from a baseline system.
- rouge: The ROUGE automatic evaluation system.
- TopicWords-v1: The topic signature tool, which includes a stopword list inside.
- global: The global TF and IDF information generated from 5000 documents from the Gigaword corpus. Stopwords are excluded.

Rouge

Rouge Score

For evaluation, we will be using the ROUGE system [16]. ROUGE is the most commonly used metric of content selection quality because it is cheap and fast, and the scores it produces are highly correlated with manual evaluation scores for generic multi-document summarization of text. ROUGE is based on the computation of n-gram overlap between a summary and a set of models (human summaries). It is a recall-oriented metric, which is more suitable for the summarization task given the variation in human content selection. ROUGE also has numerous parameters, including word stemming, stopwords removal and n-gram size. Recent research has shown that Rouge-2 Average-R score (Recall) without stopwords removed and with stemming works best for comparing different summarizations generated from machines [8]. Rouge-2 recall with the above parameters is the evaluation we will use in the project.

To score your summaries with Rouge, you simply need to call the script `ROUGE-1.5.5.pl`, which will run Rouge automatically on Eniac. We suggest you to download ROUGE folder to local, because you will need to run often if you want to track your summarizer performance as you work on the project. The default settings which we will be using for comparison is as follows:

```
./ROUGE-1.5.5.pl -c 95 -r 1000 -n 2 -m -a -l 100 -x config.xml
```

The configuration file, `config.xml`, tells Rouge what peers (systems that you are evaluating) and models (human summaries) you're using each time.

Below is an example of a call to Rouge to evaluate the baseline summarizer, and a sample output. The Rouge-2 score, 0.07722, is what we'll be looking at.

```
./ROUGE-1.5.5.pl -c 95 -r 1000 -n 2 -m -a -l 100 -x config.xml
---------------------------------------------
baseline ROUGE-1 Average_R: 0.34263 (95%-conf.int. 0.32904 - 0.35523)
baseline ROUGE-1 Average_P: 0.33855 (95%-conf.int. 0.32486 - 0.35083)
baseline ROUGE-1 Average_F: 0.34053 (95%-conf.int. 0.32711 - 0.35291)
---------------------------------------------
baseline ROUGE-2 Average_R: 0.07722 (95%-conf.int. 0.07044 - 0.08421)
baseline ROUGE-2 Average_P: 0.07630 (95%-conf.int. 0.06949 - 0.08319)
baseline ROUGE-2 Average_F: 0.07675 (95%-conf.int. 0.06998 - 0.08368)
```

If you want to test locally, you need to write your own configuration file. You may find the following link helpful, which tells you how to set up Rouge: `http://kavita-ganesan.com/rouge-howto`.

It is fine to generate summaries more than 100 words, because the Rouge system will automatically truncate the summaries to 100 words with command `-l 100`.

Configuration File

In the configuration file, you need to specify the location and name of your summary and the location and name of human summary.

```
<PEER-ROOT>
baseline
</PEER-ROOT>
<MODEL-ROOT>
models
</MODEL-ROOT>
<INPUT-FORMAT TYPE="SPL">
</INPUT-FORMAT>
<PEERS>
<P ID="baseline">summary00.txt</P>
</PEERS>
<MODELS>
<M ID="A">D30001.M.100.T.A</M>
<M ID="B">D30001.M.100.T.B</M>
<M ID="C">D30001.M.100.T.C</M>
<M ID="D">D30001.M.100.T.D</M>
</MODELS>
```

`PEER-ROOT` is where your system summaries are located, `MODEL-ROOT` is where model summaries are located. In the example above, the first summary: `summary00.txt` is compared with its corresponding four human summaries.

Comparison Groups

Baseline System

In the baseline system, we simply select the first sentence from each document in the input to form a summary. Selecting the first sentence is a strong baseline for single document summarization. In multi-document summarization, it is a reasonable baseline, with Rouge-2 recall score of 0.07722. Randomly selecting sentences is a much weaker baseline, yielding ROUGE-2 of 0.052.

1 Basic Summarizers (55 points)

First, you will implement three of the summarization approaches described in class: **Centrality**, **Topic-signature** and **LexPageRank**. We provide more detailed instructions below. We have left unspecified some of the details on sentence scoring and redundancy removal. You will make your own choices for these, and will explain your choice in the write-up. Regradless of your choices in these particular details, you should make sure that your implementation of the core algorithm is correct. Reasonable choices with correctly implemented scoring will result in Rouge-2 scores of at least 0.060 and Rouge-1 score of at least 0.320.

1.1 Centrality Summarizer (18 Points)

Centrality

The basic idea of the centrality summarizer is to find the sentences most similar to the original document. Here we define centrality as:

$$Centrality(x) = \frac{1}{N} \sum_{y \neq x} sim(x, y)$$

x and y are vectors corresponding to each sentence in the input. Each word in the input corresponds to a feature and the value of the feature is the weight of this word in vector. For weights you can use either the number of times the word appeared in the input, tf.idf weight, or binary representation (1 if the word appears in the sentence and 0 otherwise). A file with idf values can be found at `global` in project folder.

The centrality of sentence x is equal to the average similarity of the vector representation of this sentence and all other sentences y in the input. To measure similarity, you can use any of the approaches we have discussed in class: cosine similarity, Dice, Jaccard.
After previous step, we have generated the Centrality score corresponding to each sentence. Then we can have various ways of generating the centrality summarizer. Here we show one basic greedy approach here:

Greedy Algorithm

Once all sentences are assigned scores, the greedy algorithm offers a simple way to form the summary: Choosing the sentence with the highest centrality, then the second highest, the third highest, until we've reached the word-limit. The greedy algorithm in Psuedo-Code is below:

```
Greedy(Sentences, threshold)
{
    Centrality = [Sim(Sen, Doc) for Sen in Sentences]
              # Compare the vectors by definition above, omitting details
    Build the Sentence+Centrality Dictionary ----> Diction (we call it Diction)
    Sort Diction according to Centrality in decreasing order.

    Current_Summary = []; length = 0;
    while((len < threshold)and(i <= Diction.size()){
        if (Valid(Diction[i], Current_Summary)) {
            Current_Summary.append(Diction[i].Sentence)
            length += len(Diction[i].Sentence)
        }
        i += 1
    }
    print(CurrentSummary)
}
```

Now the only thing unclear in the algorithm above is in what case can we call a sentence **Valid**. Two issues should be considered:

Sentence length Some work has suggested to have a minimum number of words about 10 and maximum number of words about 35. Think about reasonable restrictions you may want to impose, *none* is an option generally considered a bad one.

Repetition Some of the highly scoring sentences may have overlap in content. One crude way to check for repetition is to compare the vector of the current sentence with the vectors of sentences already in the

summary. We will choose not to include the sentence if the similarity score exceeds some threshold (such as 0.2, 0.4, 0.6, etc), otherwise it is valid by this standard. Obviously, more sophisticated checks for shared content may result in better summaries.

Please record clearly your selection of similarity approach, sentence length limit, redundancy removal approach and any other parameter setting in your final writeup.

1.2 Topic-Word Summarizer (18 Points)

We used the TopicS tool in **Homework-1** and discussed its operation in recent Lectures. For this summarizer, we simply evaluate the importance of sentences by counting the number of topic words in the sentence. The stopword file is in folder `TopicWords-v1`. After generating weights of words, the score of a sentence S according to Topic Signature could be generated and normalized by different ways:

$$TWeight(S) = \#\ of\ topic\ words\ in\ sentence\ x \tag{1}$$

$$TWeight(S) = \frac{\#\ of\ topic\ words\ in\ sentence\ x}{\#\ of\ words\ in\ sentence\ x} \tag{2}$$

$$TWeight(S) = \frac{\#\ of\ topic\ words\ in\ sentence\ x}{\#\ of\ nonstopwords\ in\ sentence\ x} \tag{3}$$

Another parameter you need to select for your topic word summarizer is a cutoff. It is the parameter `topicWordCutoff` defined in the configuration file discussed in Homework 1. While selecting sentence, we select the sentence with the highest $TWeight(S)$, then the second highest, etc. We will also use redundancy removal to exclude potential similar sentences and excluding too long/short sentences.

In the writeup, please clarify in which approach you get scores for sentences. Also record your parameter settings and redundancy removal approach and settings as well.

1.3 LexPageRank Summarizer (19 points)

LexRank is a graph-based method of computing sentence importance. It represents each sentence as a node, then computes cosine-similarity between sentences to generate edges. If the similarity between two sentences is above some pre-defined threshold, then there is an edge between the two nodes, otherwise there isn't. Again while computing sentence similarities, it is preferred to represent your vector in TF-IDF rather than frequency approach. You may find more details about LexRank in this paper:

`http://acl.ldc.upenn.edu/acl2004/emnlp/pdf/Erkan.pdf`

For the writeup, please include your choice of threshold when selecting edges, your criteria for stopping the iterative LexRank approximation, and again your redundancy removal approach.

2 Your Summarization System (30 Points)

Now, you get to design your own summarizers. It can be either Extractive or Abstractive, designed in anyway you want. A requirement is that you must use **at least two tools or resources** in your summarization systems, for example, MPQA, WordNet, or NLP processing modules (coreference, named entity recognition, POS tagging syntactic parsing) from CoreNLP, Stanford-PosTagging, etc. You may find this project resource page helpful:

`http://www.cis.upenn.edu/~cis530/Project.html`.

Note that using SVM, WEKA or the topicS tool does not count.

3 Relative Ranking According to Rouge Score (For Extra Credit)

The performance of your system will be evaluated according to the best performance of your four systems (Ideally, it will be your summarizer). If your result is among the best five groups and has a Rouge-2 score of more than 0.088, you will get 20 points extra credits for this project. When submitting, make sure that your code can successfully generate the summaries you submit.

4 Write-up Requirement (15 points)

Your write-up should include the following sections:

1. Parameters for the basic systems:
   (1) For each summarizer, give a list of parameters. Explain how you set the parameter and explain the motivation for your choice.
   (2) Performance: Rouge-2 recall of the three basic systems.

2. For your own system, you need to show:
   (1) General idea/method of your system.
   (2) What resources or tools you have used and how are they included in your implementations.
   (3) The result of your summarization system.

3. Discussion and Analysis. Here include any analysis you have done on the output of your system or comparisons with the basic systems.

The write-up should be no more than 3 pages in PDF. You may find the templates on class project resource page useful. Of course you can have your own format, as long as your report includes all the things required. Clear and complete write-ups with strong method and analysis sections will receive up to 20 points extra credit.

You may find many useful papers about summarization in reference.

References

[1] John M. Conroy, Judith D. Schlesinger, Dianne P. O'Leary, and Dianne P. O'Leary. Topic-focused multi-document summarization using an approximate oracle score. In *ACL*, 2006.

[2] Gnes Erkan, Dragomir R. Radev, and Dragomir R. Radev. Lexpagerank: Prestige in multi-document text summarization. In *EMNLP*, pages 365–371, 2004.

[3] Yihong Gong. Generic text summarization using relevance measure and latent semantic analysis. In *in Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2001.

[4] Aria Haghighi, Lucy Vanderwende, and Lucy Vanderwende. Exploring content models for multi-document summarization. In *HLT-NAACL*, pages 362–370, 2009.

[5] A. Kulesza and B. Taskar. Learning determinantal point processes. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, 2011.

[6] Ani Nenkova, Kathleen McKeown, and Kathleen McKeown. Automatic summarization. pages 103–233, 2011.

[7] Ani Nenkova and Lucy Vanderwende. The impact of frequency on summarization.

[8] Karolina Owczarzak, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. An assessment of the accuracy of automatic evaluation in summarization. In *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*, pages 1–9, Montréal, Canada, June 2012. Association for Computational Linguistics.

[9] James Yen Paul Over. An introduction to duc-2004: Intrinsic evaluation of generic news text summarization systems, 2004.

[10] Dragomir R. Radev, Hongyan Jing, and Malgorzata Budzikowska. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *ANLP/NAACL Workshop on Summarization*, Seattle, WA, April 2000.

[11] Dragomir R. Radev, Hongyan Jing, Magorzata Sty, Daniel Tam, and Daniel Tam. Centroid-based summarization of multiple documents. pages 919–938, 2004.

[12] Lucy Vanderwende, Hisami Suzuki, Chris Brockett, Ani Nenkova, and Ani Nenkova. Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. pages 1606–1618, 2007.

[13] Xiaojun Wan, Jianwu Yang, and Jianwu Yang. Improved affinity graph based multi-document summarization. In *HLT-NAACL*, 2006.

[14] Kam F. Wong, Mingli Wu, and Wenjie Li. Extractive summarization using supervised and semi-supervised learning. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 985–992. Association for Computational Linguistics, 2008.

[15] Shasha Xie and Yang Liu. Improving supervised learning for meeting summarization using sampling and regression. *Comput. Speech Lang.*, 24(3):495–514, July 2010.

[16] Chin yew Lin. Rouge: a package for automatic evaluation of summaries. pages 25–26, 2004.