Deliverable 4 - Topic-focused Summarization via Decomposition

Kekoa Riggin, John Greve, Eric Lindberg, Joshua Mathias

Linguistics Department
University of Washington
kekoar@uw.edu, jgreve@uw.edu, lindbe2@uw.edu, emathias@uw.edu

Abstract

D3 results improved by evolving QR matrix ranking from using binary word presence to topic-weighed term frequencies in order to focus on relevance. Also a novel sentence ordering pattern was selected using relative intra-docset cosine similarity scores for test and evaluation data respectively, where the peer-summaries are used to generate the training data.

1 Introduction

Changes from previous work (Riggin et al., 2018) include updates to system architecture and a novel content selection strategy combining term frequencies and term document counts. New results include ROUGE assessments against evaluation data, which was unused by earlier iterations.

2 System Overview

Our system is a full-sentence extractive summarizer that performs content selection and information ordering on multi-article themed Document Sets drawn from the ACQUAINT and ACQUAINT-2 data sets of news articles (Graff, 2002; Vorhees and Graff, 2008). Document Sets are defined by TAC-style Topic Indices, specifically a dev set and an eval set.

Our implementation preserves the arrangement of each Document Set defined in the Topic Index. Each Document Set consists of a collection of Documents (articles) with each Document's text presented as an ordered list of Paragraphs. Extraction of relevant data from each Document Set is handled internally by the Topic Loader module which orchestrates the supporting object model.

Figure 1 shows the Summarizer module which is our main driver. The Summarizer takes a topic index file and asks Topic Loader to retrieve the relevant collection of Document Sets. The Summarizer then applies the content selection and information ordering strategies to each Document Set and saves the resulting summary in the outputs directory, one summary per file per docset.

Summaries generated by our system are capped at 100 words and contain only complete sentences. Each summary is scored according to the ROUGE metric (Lin, 2004).

3 Appproach

In this system we describe how the document sets flow through the system using the Content Selection, Content Ordering, and Content Realization modules.

3.1 Content Selection

Our system relies on a content selection method, adapted from Conroy and O'Leary's QR Matrix Decomposition (QRM) method. We build upon this system with dynamic term and sentence weights to improve selection performance.

3.1.1 Preprocessing

We tokenize sentences and words using NLTK's (Bird et al., 2009) sent_tokenize and word_tokenize functions. We preprocess words by removing stop words, including only tokens containing at least one letter, and lowercasing all words. The preprocessed words are used as input to the QR Matrix (3.1.2) and term weighting (3.1.3). However, the original sentences and words are output to be summaries.

3.1.2 QR Matrix Decomposition

The QR matrix comprises a feature vector for each sentence in a document set where the rows represent sentences and the columns represent words, and indices are populated with binary values in

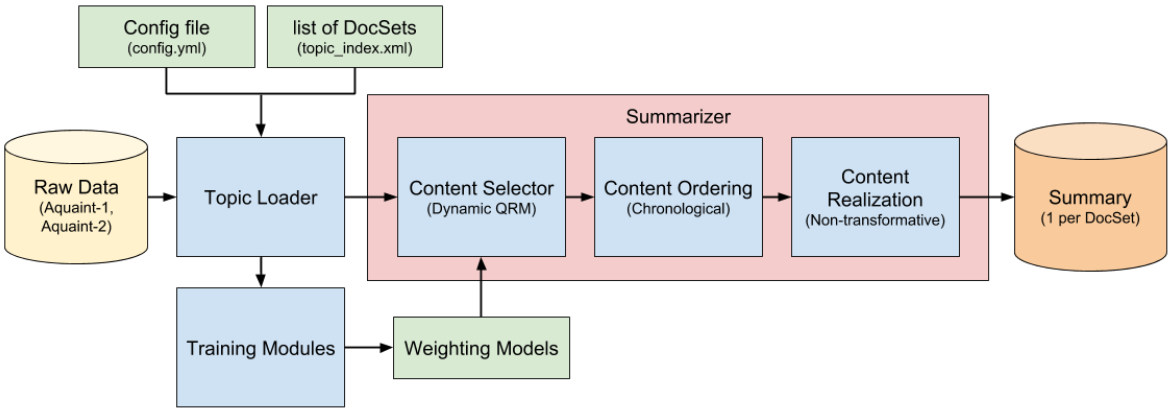

Figure 1: Summarizer Architecture

the original implementation. See below for a description of how we use the QR matrix with real-valued weights in sections 3.1.3 Term Weighting and 3.1.4 Sentence Weighting. The decomposition process proceeds as follows for both binary and real valued weights.

The sum of each feature vector is weighted, using a weighting function from the paper, and is used as a score for each sentence. The sentence with the highest score is selected. Then, the features from that sentence are removed from the matrix. This reduces redundancy in future sentences that will be selected, which was our motivation in pursuing this method.

Once all features from the selected sentence have been removed from the remaining feature vectors, the sentence scores are recalculated and this process is repeated until no more sentences will fit in the 100-word summary.

If the best scoring sentence at any recursion of the matrix is too long to add to the summary, the next best scoring sentence that is not too long is used.

3.1.3 Term Weighting

Because Conroy and O'Leary (2001) treat each word as equally informative, generated summaries suffer from an inherent lack of topic orientation in QRM. Additionally, treating all words as equally valuable gives preference to long sentences, regardless of its contribution to a summary.

To leverage the value of words for summarization, we populate the indices of the QR matrix with weighted values based on a word's term frequency, rather than binary values—as outlined by Conroy and O'Leary (2001)—ensuring that words are not treated as equally valuable.

To eliminate the effect of informationless words with high frequencies (pronouns, conjunctions, etc.) we utilized the idea of stop words from Hong and Nenkova (2014) as well as their stop words list.

Conversely, we increased the weight of topic words" by allowing the frequency count of words found in the Topic ID of the document set topic_id file to be weighted more heavily than others. We found that a weighting of 3.5 produced maximal results.

Next, we attempted a more informative measure of word value by weighting word frequency counts with an adapted TF*IDF score (Luhn, 1957).

Schilder and Kondadadi experiment with different simple features for multi-document summarization and find that their proposed document frequency performs best (2008). They define document frequency as the number of documents containing a term, among the documents to be summarized (for one summary). We combine document frequency with the term frequency of our previous work and call this tf-idf.

Unlike our previous work, we also incorporate information from the provided training data (part of the ACQUAINT data sets). After preprocessing, for each term in the training data, we count the number of document sets containing the term as well as the total number of document sets in the training data. These are used to calculate the idf (inverse document frequency) of each word at test time to weight the feature vector (representing words of a sentence) in the QR Matrix. This idf is used in the well-known tf-idf method, where tf is the tf-df previously described. We call this

method tf-df-idf (see below for formal definition). Note that this combination of frequencies is intuitive as it rewards high frequency and coverage within the document set to be summarized and penalizes high coverage in training document sets (allowing the system to determine what words are distinctive in the current document set).

t_s: term frequency in docset
d_s: number of documents term is in, within given docset
D_s: number of documents in docset
t_t: term frequency in training data
d_t: number of docsets term is in, in training data
D_t: number of docsets in training data
base: 10
$\text{tf-df}(t_s, D_s) = t_s * (1 + \frac{D_s}{base} + \log_{base}(\frac{d_s}{D_s}))$
$\text{idf}(d_t, D_t) = 1 + \frac{base}{1+d_t} + \log_{base}(\frac{D_t}{1+d_t})$
$\text{tf-df-idf} = \text{tf-df} * \text{idf}$

In addition, we experimented with normalizing t_s (in the given document set) by t_t (in the training data) in tf-df as $\frac{t_s}{t_t}$. This normalized method is described as "D4 normalized counts" in the results and discussion.

3.1.4 Sentence Weighting

Similar to term weighting, we also experimented with applying weights based exclusively on the location of a sentence within a document set, using the training data for input. The system takes a 4-gram cosine similarity value between each sentence in the peer summaries of the training data against each sentence of each article in the corresponding training document set, averaging the results for the article and sentence location from each docset.

When applied as a weight in QR Matrix, the average cosine similarity for each sentence location in the sentences L_i, and article order A_j, are multiplied along with a weighting factor W, for a score weight of $1 + WL_iA_j$.

3.2 Content Ordering

Our system implements chronological ordering, where selected sentences are ordered in the summary by the date their corresponding article was published. We had intended to use both date and time for ordering as outlined in the Chronological "Expert" (Bollegala et al., 2005); however, we discovered that the articles in the AQUAINT data did not uniformly provide time information. We modified this approach to use the article publication date as the first ordering consideration and the sentence index in its article as the second to serve as a tiebreaker for sentences from articles published on the same date or from the same article.

3.3 Content Realization

Ideally, our system would transform input sentences to maximize the performance of the dynamically-weighted QRM. Due to time constraints, the system produces purely extractive summaries. Our proposed content realization system (included but not integrated in the code release D4.01) uses Stanford Core NLP's (Manning et al., 2014) deterministic coreference system via the pynlp python wrapper.

Via Core NLP (Manning et al., 2014), we intended to implement the coreference portion of the preprocessing method outlined by Conroy and O'Leary (2004). This method replaces all referents in a chain with the most informative form of the entity (i.e. the referent with the most words) in the source text. Once sentences are selected for content and if any referent is repeated in the selection, the repetition of the entity is replaced with the original version for that instance from the source text to reduce redundancy.

Our experiments with coreference resolution and sentence compression are briefly outlined in the Discussion section.

4 Results

First we present a summary of our empirical evaluation of different versions of our system on the dev set. We found that tf-df improves ROUGE 1 through 4 recall scores compared to only tf (term frequency). We also tried df (document frequency) without tf, which improved R-2, R-3, and R-4, but lowered R-1. tf-df-idf further improves all ROUGE recall. "D4 normalized counts" represents our attempt to normalize tf over trained word counts while still applying tf-df-idf. As normalizing word counts worsened ROUGE scores, we removed it from the weighting. Improved scores for D4 topic words" demonstrates that weighting words higher for being in the title further improves results on top of tf-df-idf.

The D3 vs D4 results are shown in Table 1. Here D4 is the final version which includes tf-df-idf and topic word weighting. As noted in our D3 paper, we found a stopword related bug during post-hoc

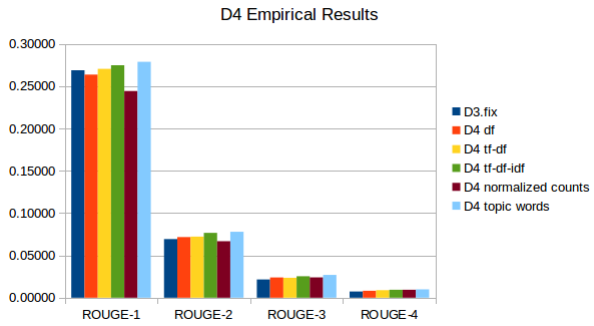

Figure 2: D4 Rouge Scores

analysis. The D3.fix results in Table 1 show the full potential of the D3 strategies and are the relevant point of comparison for judging D4 improvements (which also include D3.fix), though we do show original D3 scores for consistency with our previous paper.

Table 2 shows percentage changes, where you can see a lift from D3.fix to D4 devtest with an increase of 14.5% and 24.7% in average recall for Rouge-1 and Rouge-2, respectively. Comparing D4 devtest to evaltest, we observed increases of 9.1% and 6% in average recall for Rouge-1 and Rouge-2, respectively.

Table 3 shows the results of the sentence location experiment. Initially, the results of applying sentence location alone proved surprisingly effective. However, applying the weights, as described above, did not have much effect on the resulting score, no matter what weight factor was applied. The effect of the weighting seemed to cease changing the results in any way with a weight factor of 3.0 or more. Results differ by less than a tenth of a percentage point, being slightly better in the eval data, and slightly worse for dev.

5 Discussion

As we tested each of our improvements separately on the dev set and in different combinations (not all trials are presented), it appears that df, tf, and topic title weighting can all independently improve results, and they work even better by combining them using simple multiplication or division (tf-df, tf-idf). Other researchers performing multi-document summarization may similarly be able to successfully incorporate these weights in their system.

In analyzing how to increase the information density of our summaries, we discovered that, because the QRM method prefers longer sentences, we sacrificed highly informative sentences that would not fit in the remaining space for our 100-word summaries.

It became apparent to us that if we could reduce the size of our selected sentences, or even phrases within them, we could potentially increase the amount of information packed into our summaries.

Coreference resolution was a promising technique we attempted, because shorter and more informative phrases might be used to identify entities in our summaries. However we were unable to progress our coreference module from a prototype to full functionality.

Additionally, because our QRM implementation is topic-oriented, sentence compression and coreference are promising content realization approaches since removing words with low informedness from the source sentences would theoretically have little impact on sentence scores.

The sentence weighting results suggest that sentence location is a surprisingly good predictor of salient sentence for summary. This is probably because the structure of the news stories are quite similar, making simple sentence location a plausible measure of usefulness. Despite this, the resulting summaries from location only selection were not good matches for human readers.

We manually evaluated our output summaries for our final system and found them to contain topic-focused sentences, and sentences within a summary are diverse. We also found that our information ordering scheme was effective, perhaps because the most central information is usually described in earlier articles and earlier in the document (especially the first sentence). Later sentences are informative but supplementary.

6 Conclusion

Our experiments demonstrate the effectiveness of combining various frequency counts and topic-focused information, as well as incorporating these weightings in a QR Matrix which removes redundancy. We also showed that using sentence position information for content selection can give decent ROUGE scores but may result in poor summaries. Future work may successfully combine our tf-df-idf method with content realization strategies and with an HMM approach as described by Conroy and O'Leary (2004) and de-

Table 1: D3 vs D4 Rouge 1 and 2 Average Recall Results

Recall Score	D3.devtest	D3.fix	D4.devtest	D4.evaltest
ROUGE-1	0.2226	0.2484	0.2787	0.3067
ROUGE-2	0.0522	0.0634	0.0779	0.0831

Table 2: % Change in Rouge 1 and 2 Average Recall Results

% Change	D3 to D3.fix	D3.fix to D4	D4 dev to eval
ROUGE-1	11.56%	12.22%	10.03%
ROUGE-2	21.59%	22.83%	6.66%

Table 3: Sentence Location Summary Results

Recall Score	Location.dev	Location.eval	WeightedQR.dev	WeightedQR.eval
ROUGE-1	0.2713	0.3067	0.2784	0.3075
ROUGE-2	0.0763	0.0831	0.0777	0.0833

scribed in our D3 work (Riggin et al., 2018).

References

S. Bird, E. Klein, and E. Loper. 2009. *Natural Language Processing with Python*, 1st edition. O'Reilly Media, Inc.

D. Bollegala, N. Okazaki, and M. Ishizuka. 2005. A preference learning approach to sentence ordering for multi-document summarization. *In Proceedings of the 2nd international joint conference on natural language processing*.

J. M. Conroy and D. P. O'Leary. 2001. Text summarization via hidden markov models. *In Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 406–407.

J. M. Conroy and D. P. O'Leary. 2004. Left-brain right-brain multi-document summarization. *In Proceedigns of DUC 04 conference*.

D. Graff. 2002. The aquaint corpus of english news text. *Web Download*.

K. Hong and A. Nenkova. 2014. Improving the estimation of word importance for news multi-document summarization. *In Proceedings of the 14th conference of the European chapter of the association for computational linguistics*, pages 712 – 721.

C. Lin. 2004. Rouge: a package for automatic evaluation of summaries. *In Proceedings of Document Understanding Conference (DUC) 2004*.

H. P. Luhn. 1957. A statistical approach to mechanized encoding and searching of literary information. *IBM journal of research and development*, pages 309 – 317.

C. D. Manning, M. Surdean, J. Bauer, J. Finkel, S. J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55 – 60.

K. Riggin, J. Greve, E. Lindberg, and J. Mathias. 2018. Topic-focused summarization via decomposition, d3.

F. Schilder and R. Kondadadi. 2008. Fastsum: fast and accurate query-based multi-document summarization. *In Proceedings of the 46th annual meeting of the association of computational linguistics on human language technologies: short papers*, pages 205 – 208.

E. Vorhees and D. Graff. 2008. Aquaint-2 information-retrieval text research collection. *Web Download*.

Appendix

Technical Notes: Running on the D4.01 System

A full D4.cmd condor run on an empty document cache (no "shelve" files) takes about 7 hours vs about 20 seconds with a populated cache. Please use the D4.01 tag, which corrects modest oversights to D4 including 1) not "git adding" all of the D4 deliverables (outputs/D4_⋆ folders and result files), 2) ensuring both devtest and evaltest are run in D4.cmd's driver shell script, bin/summarizer_patas, and 3) adding a config specification for 2+ ROUGE calculations (the original D4 tag was writing ROUGE results to a hardwired path so evaltest would overwrite devtest). Note that these are all D4 configuration changes, none of the core logic in src/⋆.py was modified.

Optional

The shelve cache files are not in git, but a suitable D4 version is available in case you don't want to wait for the cache initialization process. Using the pre-filled cache files drops the run time to about 20 seconds. Before running D4.cmd, do:

```
$ cp /home2/jgreve/tools/d4_shelve/* .
```

Troubleshooting

An error message we have seen during development.

```
FileNotFoundError: [Errno 2]
No such file or directory:
'outputs/position_data/sample_article_data.csv
```

D4.cmd requires some $\star$.csv training data files in outputs/position_data, which is in git. Potential fix: Try doing $ git checkout – outputs and verify the outputs/position_data folder has csv files.