LING 573 Project Report

Macklin Blackburn, mblac6@gmail.com
Xi Chen, xch@uw.edu
Yuan Zhang, yuanz80@uw.edu

June 4, 2017

Abstract

Multi-document summarization is a longstanding Text Analysis Conference (TAC) shared task aimed at automatically generating one summary for multiple documents on the same topic. This paper tackles the challenge by employing a rich set of statistical and linguistic features and a multi-layer perceptron model in content selection, a clustering approach in information ordering, and rule-based sentence compression in content realization. Our system generates a 100-word extractive summary and produces better ROUGE-2 recall scores on the TAC 2010 data than the MEAD and LEAD baselines.

1 Introduction

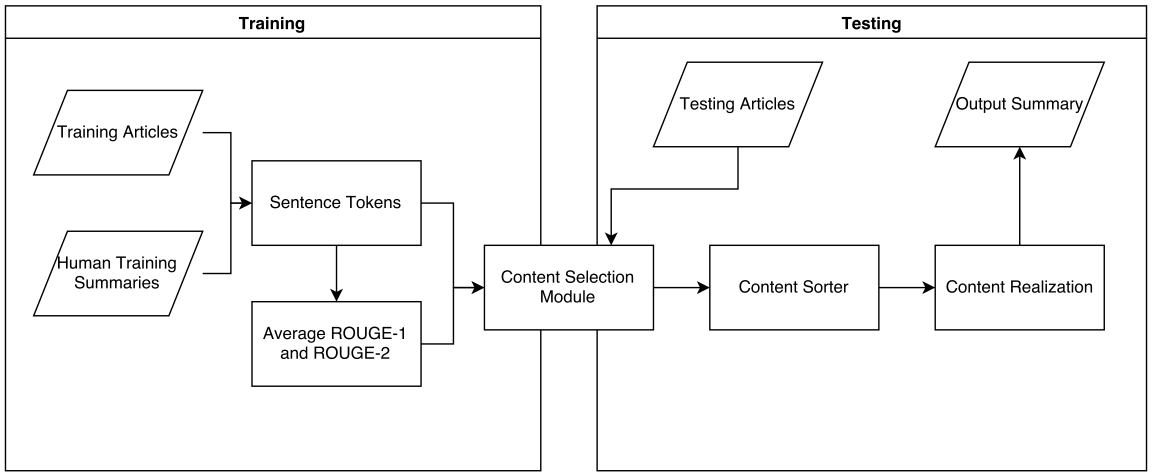
Automatic summarization is an NLP task that turns a set of documents related to the same topic into a summary. This task was first attempted by Luhn (1958). In recent decades, there have been several different approaches to the problem. These approaches fall into two main categories: supervised and unsupervised. Examples of the former category include Schilder(2008), Vanderwende(2007), and Cao(2015). Examples of the latter include Erkan(2004) Otterbacher(2005) and Radev(2004).

This paper demonstrates a system composed of three sub-systems. Firstly, in the content selector module, each sentence from a cluster of documents is assigned a score to indicate its importance in the final summary. Secondly, in the content sorter module, sentences are sorted by scores. Finally, in the content realizer, top-scoring sentences are picked, reordered, and compressed to form the final summary.

We adopt a supervised approach modeling the task as sequential labeling. Given the input of a feature vector of each sentence, the system will output a score between 0 and 1 to indicate the likelihood of each sentence to appear in the final summary:

$$\mathrm{F}(\mathrm{f}_{\mathrm{n1}},\mathrm{f}_{\mathrm{n2}},...\mathrm{f}_{\mathrm{nk}}) \rightarrow \mathrm{Score}_{\mathrm{n}}$$

2 System Architecture



3 Approach

3.1 Data

We carry out our experiments on four datasets from the past Document Understanding Conference (DUC) and TAC competitions. The DUC 2007 and 2009 data are used for training and the TAC 2010 data are used for development. The TAC 2011 data are used for final testing.

There are 45 topic clusters in DUC 2007, 44 in DUC 2009, 46 in TAC 2010, and 44 in TAC 2011. Each cluster contains 10 news articles on the given topic. For each cluster, there are 8 human-written summaries. These human summaries are abstractive with paraphrasing and rewording.

In addition, we extract from the DUC 2009 peer summaries all the human-ordered extractive summaries, marked with the annotator ID 3, for training and evaluating our information ordering module.[5]

3.2 Preprocessing

For preprocessing, we split each document into sentences using the NLTK sentence tokenizer, lowercase each sentence, remove stop words and non-alphanumeric characters, and lemmatize all verbs and nouns using NLTK WordNetLemmatizer package. For each dataset, we produce two parallel dictionaries, one containing the original sentences and the other processed sentences as lists of word tokens, respectively. Finally, we remove from the dictionaries the sentences, which, after preprocessing, have been reduced to empty lists. We use the processed sentences as the input to our system and the original dictionary for system output.

3.3 Topic Orientation

For topic-guided summarization, we adopt the query-based LexRank approach [3]. We combine a continuous LexRank score and a relevance measure into a stochastic graph-based model and we give the two measures different weights in multiple trials. The topic relevance measure is heavily dependent on word token overlaps between a topic or query and a sentence, balanced by an IDF score. The continuous LexRank aims to get at the salience of each sentence in the document set by a cosine similarity measure applied on the tf-idf score of each word in the sentence. The result of this experiment will be presented in Section 5.1.

3.4 Content Selection

Our content selection system uses multi-layer perceptron regression to identify sentences that are likely to have high ROUGE scores based on a number of features. The ranking of the features is shown below.

Feature	Score
Query Lexrank	0.166987
Sentence Length	0.073955
Earliest First Word	0.070324
Count of NN	0.061519
Is First Sentence	0.057551
LLR Sum	0.056517
Sentence Index	0.054131
TF*IDF Average	0.041956
Average First Word	0.040914
Lexrank	0.036588
Reverse KL(Bigrams)	0.030090
KL(Unigrams)	0.029986
KL(Bigrams)	0.027208
Average Word Position	0.026866
LLR	0.023636
TF*IDF sum	0.022663
Reverse KL(Unigrams)	0.021294
Count(JJ)	0.020824
Sentiment	0.017988
P(Number)	0.016580
Count(VBP)	0.013500
Count(VB)	0.013039
Count(RB)	0.012758
Count(IN)	0.012695
Count(NNS)	0.011601
Count(VBD)	0.009724
Count(VBZ)	0.006536
Count(NNP)	0.005466
Count(VBN)	0.004719
Count(PRP)	0.004527
Count(Comma)	0.003646
Count(DT)	0.002395
Count(CC)	0.001817

Since our system took inspiration from the RegSum system, it includes some of the same features, such as the earliest first occurrence of a word or the average position of a word in a document. These features are usually averaged over the words of a sentence, but sometimes a sum was a more effective feature. The sentence intensity score was computed using VADER as included in the NLTK package for python, and was intended to approximate several features in RegSum that contained sentiment information. However, there are several features in this system that are not present in RegSum, such as the KL divergence of bigrams. The content selector uses NLTK's part of speech tagging to count the number of certain parts of speech in the sentence.

During the training stage, the system constructs a vector for each sentence containing each of the features and assigns the sentence a score. The score is computed by averaging all ROUGE scores of the sentence compared to the relevant summaries. Earlier versions of the system assigned each sentence a binary score which represented whether the sentence was in the summary or not, but training with the ROUGE scores yielded better results. After assigning each sentence a score, the sentence vectors are scaled using a function from the SciKit Learn package and the sentence scores are scaled so that the highest score is 1 and the lowest is 0.

3.5 Sentence Ordering

Because the scarcity of training data for sentence ordering, our information ordering procedure is unsupervised one. It is based on sentence cluster adjacency Ji(2008). Given a set of summary sentences $\{S_1,S_2,...S_K\}$, we cluster all sentences in the original documents into K groups $G_1,G_2,...,G_K$ where S_i corresponds to G_i. For each group pairs, the adjacency of G_i and G_j is defined below:

$$C_i, j = \text{pow}(\text{freq}(G_i,G_j),2)/(\text{freq}(G_i)\text{* freq}(G_j))$$

Here f(G_i) denotes the frequency of cluster G_i in source documents and freq(G_i,G_j) denotes the co-occurrence frequency between G_i and G_j in a word-window which has a size of 2 words.

Our current implementation always picks the first sentence in the first published document as the first sentence. After that, a greedy algorithm can be used to order the summary sentences. That is, to find the (i+1)th sentences by maximizing the adjacency:

$$S_i + 1 = \text{arg max}(C_i, j) \text{ where i is the previous sentence in the summary}$$

The implementation uses label spreading routine of scikit-learn to make the clustering and Glove 50D word embedding to measure sentence cosine similarity.

3.6 Sentence Compression

The sentence compression for the system removed dates and times, ages, attributions, adverbs, adjectives and initial conjunctions. Only pre-verbal or pre-nominal adverbs and adjectives were removed. In several tests we found that removing dates/times, ages, and attributions led to a rouge score that was fourteen percent lower. Removing the adverbs, adjectives, and conjunctions dropped the rouge score by twenty six percent. The compression also made some fairly awkward sentences where very important information was omitted. We decided it was best not to include sentence compression in our final submission, but we included our sentence compression code anyway.

3.7 Content Realization

The surface realization is simple: it fetches all sentences with highest possible scores until the final summary length is reached. The realization procedure is integrated with the sentence compression, it calculates summary length according to the length of compressed sentences. To reduce redundancy, when the TF*IDF cosine similarity between a candidate sentence and a chosen sentence in the summary is greater than a threshold, the candidate sentence will be ignored. The current threshold is set to 0.4.

4 Results

4.1 ROUGE Recall Scores

Model	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4
D2	0.18687	0.04579	0.01503	0.00558
D3	0.22459	0.05354	0.01287	0.00424
D4 Devtest	0.23406	0.07048	0.02468	0.01198
D4 Evaltest	0.27357	0.06555	0.01845	0.00766
D4 Devtest Attempt 2	0.24258	0.06551	0.02134	0.00683
D4 Evaltest Attempt 2	0.26533	0.06295	0.01744	0.00724

Table 1: In our first test, the R1 for the evaltest data was much higher than on devtest but all other R scores were lower. This was very surprising given the results of other groups. A second test showed comparable results. The causes for the drop in performance are unclear.

5 Discussion

5.1 Topic Orientation

Our D3 submission added a query-based lexrank score as a feature in the content selection model. Query-based lexrank was ranked the highest of the features by information gain and improved the ROUGE-2 recall of our overall system by roughly 0.008. An experimental feature that measured the KL divergence between a sentence and the topic of the cluster was ranked as the third best feature by information gain but lowered the overall ROUGE-2 recall by roughly 0.007. It's unclear why topic-focused lexrank is so much more effective than topic-focused KL divergence, especially when the information gain of the KL feature is comparable. Our speculation is that the model may rely too heavily on sentence word overlap with the topic, so that it favor sentences that are too similar to each other and contribute roughly the same information to the end summary. Query-based lexrank might be a good feature because it balances relevance to the topic with relevance to the document cluster as a whole.

5.2 Sentence Ordering

Our implementation uses label spreading routine of scikit-learn to make the clustering and word embedding to measure sentence cosine similarity. Because the original paper doesn't contain details about its feature set, We make some numerical experiments to find which word embedding/feature set to use. The metric used is the Kendall's tau, which is defined as:

tau = 1 - 4(number of inversions)/ N(N-1)

Here N is the number of summary sentences. Sentences ordered by the algorithm are compared to 20 passages of human extracted summaries (AQUAINT and AQUAINT-2) to get the score. A baseline of random ordering is used as the baseline. The results are drawn in table 2.

Name	tau	word embedding	window size	memo
Random	0.0	n/a	n/a	baseline
FW2-D50	0.39	glove average d=50	forward windows size=2	
FW2-D100	0.41	glove average d=100	forward windows size=2	
SW2-D50	0.35	glove average d=50	symmetric windows size=2	the one in final delivery
SW2-D100	0.39	glove average d=100	symmetric windows size=2	
FW1-D100	0.33	glove average d=200	forward windows size=2	
FW1-PD100	0.43	paragraph2vec d=100	forward windows size=1	
FW1-PD100	0.32	paragraph2vec d=200	forward windows size=1	
Chronological	0.46	n/a	n/a	published date based

In our experiment, different window size and embedding dimension combinations are tested. It shows adjacency based ordering is far more useful than baseline. Although its tau score is slightly less than chronological sorting, our reading of their results is that they are at least partially independent. In our evaluation dataset (20 human-ordered extractive summaries), word vector based similarity gives better results on 5 passages while the chronology based one gives better results on 8 passages. This suggests that a similarity metric which combines both of them might be better than any single one. This attests to the validity of the main idea of Bollegala et al's paper named "A preference learning approach to sentence ordering for multi-document summarization" [7].

5.3 LLR & Devtest

For D4, we used a larger background corpus, half of the New York Times corpus, to calculate the LLR. This seems to have positive effects on the ROUGE scores on the development data while simultaneously leading to a drop in the information gain of LLR. Rouge-2 increased from 0.053 to 0.070. An empirical survey of the output summaries for D4 compared with those for D3 shows an increasing presence of the topic in sentences picked for the final summary in D4 and a general lack of overlaps between D3 and D4 summaries. It would seem that somehow the modified LLR has augmented the status of query-based lexrank as a feature and changed the overall scoring for the candidate sentences. The cause of this may be the lower info gain of the modified LLR. In D3, LLR was the second highest feature by info gain behind query based lexrank. After the info gain of LLR dropped, the D4 system may have increasingly relied on query-based lexrank. In some cases, the modified LLR managed to produce more coherent summaries and identify good first sentences. In others, the inclusion of more topic-focused sentences shows no correlation with readability.

5.4 Devtest & Evaltest

It is puzzling that our D4 system performed better on the devtest data than the evaltest data in ROUGE-2, 3, 4, the opposite of how our peers' systems did. We have checked the devtest data to rule out errors in data extraction and preprocessing. We have tested our D3 system on the dev and evaltest data and received lower scores in the evaltest, as well. This means that the modified LLR is not the cause of our problem.

One thing is certain. There must be a substantive difference between either the human summaries or the documents in the evaltest set and the devtest, which results in general higher scores on the evaltest data. Since we feed the human summaries as training instances into our model and our model is trained with average ROUGE-1 to 4 scores as targets, we hypothesize that it may have been overfit to the training data.

5.5 Problems with Variation in System Scores

Our system uses MLP regression, which uses pseudo-randomness so that each run yields slightly different results. This is the only source of variation in content selection. However, even after fixing the seed of the pseudo-random number generator, or changing the regression to linear, there is still variation in the scores of the system. We believe that the source of the randomness is information ordering. Our system uses word embeddings during information ordering, and the order of sentences varies between system runs. Since our system makes summaries over the 100 word limit and truncation is done by ROUGE eval, different sections are truncated in different runs. This means that final ROUGE scores are slightly variable.

6 Conclusion

We have significantly improved our overall evaluation scores from D2 to D4. Preprocessing, expanding training data, and feature engineering in content selection seems to have largely accounted for the improvement in our system. While the RegSum approach to feature engineering shows promise, it also makes diagnostics more difficult and sometimes produces unexpected results. There is much room for further exploration with regard to information ordering to improve the readability of our summaries. Our experiments have led us to the conclusion that a good information ordering system must include chronological ordering in some measure. While our first experiment with rule-based sentence compression failed to boost our scores, we may yet hit upon a good recipe with further experiments.

References

[1] Ji Donghong and Nie Yu. Sentence ordering based on cluster adjacency in multi-document summarization. *The third international joint conference on natural language processing*, 2008.

[2] G. Erkan and D. Radev. Lexrank: graph-based lexical centrality as salience in text summarization. *J. Artificial Intelligence Research, 22(1):457-479*, 2004.

[3] J. Otterbacher G. Erkan and D. R. Radev. Using random walks for question-focused sentence retrieval. *Proceedings of Human Languages Technology Conference and Conference on Empirical Methods in Natural Language Processing, pp.915-922*, 2005.

[4] K. Hong and A. Nenkova. Improving the estimation of word importance for news multi- document summarization. *in Proceedings of EACL*, 2014.

[5] P.E. Genest G. Lapalme and M. Yousfi-Monod. Hextac: The creation of a manual extractive run. *Proceedings of the Second Text Analysis Conference (TAC 2009)*, 2009.

[6] A. Nenkova and K. McKeown. Automatic summarization. *Foundations and Trends in Information Retrieval*, 2011.