Ling 573 Project: Multi-Document Summarization

Emma Bateman
ebateman@uw.edu

John Dodson
jrdodson@uw.edu

Charlie Guo
qcg@uw.edu

Abstract

We describe an unsupervised multi-document summarization system using TextRank, a graph-based algorithm for ranking salient sentences in a corpus of text documents. The most salient sentences are selected and then re-ordered using a local coherence algorithm for modeling entity distributions across the input texts. The system is divided into four components for selecting content, generating feature representations, ranking, and ordering.

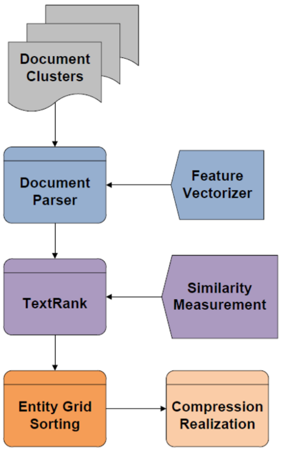

Figure 1: Multi-document summarization system architecture.

1 Introduction

Multi-document summarization systems seek to aggregate collections of text documents and condense their content to provide cohesive summaries. Traditional approaches to summarization fall into two distinct categories: abstractive and extractive. Abstractive techniques are categorized based on their ability to generate novel summary content, in contrast with extractive approaches which select significant content directly from the source documents. In this work, we present an unsupervised system for extractive multi-document summarization over a benchmark dataset. The system is centered around the TextRank algorithm, which is a graph-based technique to identify and rank the most salient sentences in the input documents. We evaluate our system on the AQUAINT and AQUAINT-2 corpora, with average ROUGE-1 and ROUGE-2 F-scores of 0.26093 and 0.06952, respectively.

This report is structured as follows: Section 2 provides an overview of our system and its components, including a description of the AQUAINT dataset; section 3 details our approach to generating summaries using TextRank and entity-grid representations; section 4 presents result metrics; section 5 is a discussion of the approach and details potential improvements to the baseline system; section 6 concludes.

2 System Overview

The system is divided into three primary components: a content selector, a ranker, and a sentence ordering algorithm. The content selector is responsible for ingesting a schema file which details one or more document clusters in the AQUAINT datasets. The selector identifies which documents belong together in a cluster and then ingests those documents as plain text. The system extracts feature vectors from each sentence by computing word frequencies. The feature vectors are given as input to the ranking algorithm which runs TextRank to sort vectors by salience. Ranked sentences undergo deduplication and the system selects the top-K salient sentences as input to an entity-based representation mechanism which is used to model coherence and improve human readability.

2.1 Architecture

Figure **??** provides an illustration of the existing system architecture.

2.2 AQUAINT Dataset

We use the AQUAINT (Advanced Question-Answering for Intelligence) and AQUAINT-2 corpora to provide input to our system. Both datasets are composed of English newswire texts. For the multi-document summarization task, we leverage a corresponding schema file which defines one or more document-topic clusters.

3 Approach

This section details the main components of the working system.

3.1 Content Selection

The ingest processor for the system consumes one or more documents associated with a cluster identifier. These SGML documents are converted to plain text and represented as a single monolithic text string. After ingest is complete, the system will maintain one text representation per document cluster.

The system uses BeautifulSoup to perform most of its SGML and XML parsing. For a given input file, the ingest processor identifies clusters and their associated documents by finding document IDs which correspond to AQUAINT SGML files. The system then loads the corresponding SGML files and pefroms traversal using BeautifulSoup to find the appropriate blocks of text.

3.1.1 Feature Vectors

The feature extraction component takes the plain text representation and tokenizes it into individual sentences using Python's Natural Language Toolkit (NLTK). NLTK is used to remove common stopwords from each sentence. Feature vectors are generated in two steps: firstly, the component identifies a comprehensive vocabulary for the cluster; and secondly, a frequency vector is created based on the token frequencies in the given sentence.

The output of this component is an N x V matrix, where N is the total number of sentences in the cluster and V is the size of the vocabulary.

3.1.2 TextRank

Our system uses TextRank to compute saliency scores for each sentence in the cluster and subsequently rank them according to significance. The TextRank algorithm constructs a graphical representation of textual features, where a given textual feature is represented as a vertex in the graph. The algorithm assigns a significance score to each vertex based on inbound edge weighting.

The authors of the TextRank algorithm formally define a graph as a tuple $G = (V, E)$, where $V = \{v_1, v_2, ...v_n\}$ is the set of vertices in the graph and E is a set of edges and a subset of V x V. TextRank is inspired by PageRank and computes the score for some vertex V_i similarly to the original algorithm:

$$S(V_i) = (1 - d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

$In(V_i)$ represents all inbound connections to vertex V_i, and $Out(V_j)$ represents outbound connections from vertex V_j. The authors use d as a damping factor set between 0 and 1, which effectively biases the calculation. The original PageRank algorithm defaults this factor to 0.85, and in our system we maintain this defaulted value.

In applying TextRank to sentence processing, we compute a pairwise similarity matrix of size N x N, where N is the total number of sentences in the cluster. The similarity between N_i and N_j becomes the edge weight which connects those two vertices in the graph. We use cosine similarity as our distance measurement.

After the TextRank algorithm converges, the algorithm returns the top K sentences as the cluster summary. Each summary contains 100 words maximum. The system maintains a reference to the original sentences, and uses the original text as output.

3.1.3 Redundancy

In our earlier system, we encountered significant amounts of redundancy, including repeated sentences. In one case, a sentence appeared in a summary three times in a row.

> "The nation deserves and I will select a Supreme Court justice that Americans can be proud of," Bush said.
> "The nation deserves and I will select a Supreme Court Justice that Americans can be proud of," Bush said.
> "The nation deserves and I will select a Supreme Court justice that Americans can be proud of," Bush said.
> Sandra Day O'Connor, the first woman ever appointed to the US Supreme

> Court, said Friday that she is retiring, giving US president George W. Bush his first opportunity to appoint a justice.

We fixed this issue by modifying the TextRank algorithm. At each iteration, until enough sentences have been selected to complete the summary, the most salient sentence is extracted. Then, the corresponding feature vector is subtracted from the rest of the matrix, disadvantaging sentences with a high level to similarity to the selected sentence. Finally, the similarity algorithm is rerun before the next sentence is selected.

3.2 Information Ordering

The system uses an entity-based modeling approach to improve summary coherence and provide a more logical ordering of the sentences in each summary. Our approach is inspired by the entity grid mechanism introduced in Barzilay and Lapata 2008, wherein the authors describe a local coherence algorithm centered around entity transitions. Our algorithm models the distribution of entities across the top outputs of TextRank, which means the input data is fundamentally small. We make the assumption that the input sentences naturally contain local coherence after applying TextRank.

3.2.1 Extracting Grammatical Roles

The authors of the original algorithm exploit several factors in their entity representations, namely, coreference resolution, grammatical roles, and saliency. Our system only uses grammatical function in determining entity types. We use the Stanford CoreNLP toolkit to generate dependency parses over each input sentence. Subject and object relations are extracted from the parse tree and used to model entity transitions. We provide weights for entities with subject, object, and alternate relation categories.

3.2.2 Entity Grid Representation

Following the conventions established in Barzilay and Lapata 2008, we use the grammatical roles described in Section 3.2.1 to denote types of discourse entities. After generating dependency parses and compiling the set of unique entities in the input corpus, we compile a table representation of size N x M, where N is the total number of sentences in the input corpus and M is the total number of unique entities extracted from that corpus. In our system N is reasonably small.

The entity grid is populated with weight values corresponding to the grammatical function of an entity in a given sentence, where a table entry $E_{i,j}$ is weighted depending on the grammatical role of entity j in sentence i.

3.3 Baselines

In addition to the entity grid algorithm, we implemented three baseline ordering algorithms.

3.3.1 Saliency Order

The saliency baseline outputs sentences in the same order that they were returned by TextRank.

3.3.2 Original Document Order

For the original order baseline system, the position of a sentence in a summary is determined by its placement in its original document.

During the feature extraction stage, a number is generated for each sentence to represent the position of the sentence in the document. This number is calculated by dividing the number of preceding sentences by the total number of sentences in the document.

After TextRank is run, the selected sentences are sorted by their position values to form the final summary.

3.3.3 Traveling Salesman Order

The "traveling salesman" baseline system orders the sentences by solving the traveling salesman problem for the feature vectors of the selected sentences.

The distances between vectors are calculated using cosine similarity.

The sentences are output in the order of the shortest path, starting with the most salient of the two endpoints.

4 Results

Table 1 and **Table 2** present ROUGE-1 and ROUGE-2 metrics for development and evaluation data.

Development set ROUGE-1 and ROUGE-2 scores						
Algorithm	**ROUGE-1 Recall**	**ROUGE-1 Precision**	**ROUGE-1 F1**	**ROUGE-2 Recall**	**ROUGE-2 Precision**	**ROUGE-2 F1**
Entity grid, rule-based	0.21643	0.23487	0.22322	0.05036	0.05311	0.05133
Shortest path, rule-based	0.21865	0.23280	0.22400	**0.05054**	0.05282	0.05136
Entity grid, model-based	0.20716	0.27003	0.23264	0.04610	0.05970	**0.05169**
Shortest path, model-based	**0.20730**	**0.27023**	**0.23280**	0.04586	**0.06043**	0.05173

Table 1: Evaluation metrics on AQUAINT data

Evaluation set ROUGE-1 and ROUGE-2 scores						
Algorithm	**ROUGE-1 Recall**	**ROUGE-1 Precision**	**ROUGE-1 F1**	**ROUGE-2 Recall**	**ROUGE-2 Precision**	**ROUGE-2 F1**
Entity grid, rule-based	0.25371	0.27168	0.26009	0.06173	0.06545	0.06281
Shortest path, rule-based	0.25628	0.26949	0.26154	0.06235	0.06441	0.06305
Entity grid, model-based	**0.25656**	**0.32682**	**0.28385**	**0.06695**	**0.08390**	**0.07365**
Shortest path, model-based	0.24993	0.31408	0.27512	0.06276	0.07798	0.06882

Table 2: Evaluation metrics on Gigaword data

5 Discussion

Problematically, there was little difference between our various sorting algorithms. Part of the problem is that the algorithms had to work with what they were given; often, there was no good way to organize the selected sentences.

A big issue with our current system is that it lacks sufficient preprocessing and postprocessing. By incorporating more linguistic information in our feature vectors and transforming the selected sentences during content realization, we hope to make significant improvements in our next iteration.

6 Conclusion

We have described an unsupervised system for generating single summaries over multiple documents. The system generates feature vectors per sentence by computing word frequencies, and then provides those frequency vectors as input to the TextRank algorithm. The algorithm is able to identify significant lines of text in a single pass. We take the top K sentences as the corresponding summary, up to 100 words max per summary.