Summarizing Email Threads

http://www.cs.columbia.edu/nlp/papers/2004/rambow_al_04.pdf

Description of an email thread:

- coherent exchanges of email messages among several participants
- the discourse participants are not physically co-present
- responses need to take special precautions to identify relevant elements of the discourse context (for example, by citing previous messages)

Important Points:

- paradigm used for other genres of summarization, namely sentence extraction: important sentences are extracted from the thread and are composed into a summary
- email-specific features can help in identifying relevant sentences for extraction
- learn sentence extraction strategies using machine learning
- summarizing the whole thread instead of individual emails
- Data: 96 threads of email, Columbia University

Features for Sentence Extraction

1. Basic Features

These features consider the thread as a single text.
The sentence in consideration s, the message in which the sentence appears m, the thread in which the sentence appears t, and the entire corpus c

- *thread_line_num*: The absolute position of s in t.
- *centroid_sim*: Cosine similarity of s's TF-IDF vector (excluding stop words) with t's centroid vector. The centroid vector is the average of the TF-IDF vectors of all the sentences in t. The IDF component is derived from the ACM Corpus.
- *centroid_sim_local*: Same as centroid sim except that the inverse document frequencies are derived from the thread.
- *length*: The number of content terms in s.
- *tfidfsum*: Sum of the TF-IDF weights of content terms in s. IDF weights are derived from c.
- *tfidfavg*: Average TF-IDF weight of the content terms in s. IDF weights are derived from c.

- *t _rel_pos*: Relative position of s in t: the number of sentences preceding s divided by the total number of sentences in t. All messages in a thread are ordered linearly by the time they were sent.
- *is_Question*: Whether s is a question, as determined by punctuation

2. Full Features

Features which address the specific structure of email communication.

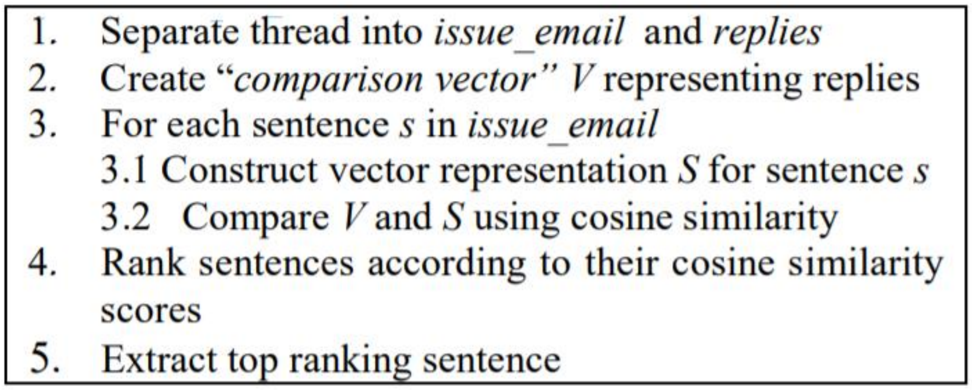
- *subject sim*: Overlap of the content words of the subject of the first message in t with the content words in s.
- *num of res*: Number of direct responses to m.
- *num Of Recipients*: Number of recipients of m. • fol Quote: Whether s follows a quoted portion in m

Generating Overview Summaries of Ongoing Email Thread Discussions

https://www.aclweb.org/anthology/C04-1079.pdf

The focus of this paper is on email discussions supporting a group decision-making process

- uses the structure of the thread dialogue and word vector techniques to determine which sentence in the thread should be extracted as the main issue
- identifies the sentence containing the issue of the thread being discussed, potentially more informative than subject line.
- uses a combination of traditional vector space techniques and Singular Value Decomposition (SVD)
- **Important:** " rely on the premise that the participants of the discussion have implicitly determined which sentence from the initiating email of the thread is most important and that we can see evidence of this inherent in the content of their respective replies"
- create a summary for a single task.
- instead of extracting email texts verbatim, we extract single sentences from particular emails in the thread.
- **Assumptions:** First email describes the issue, does not contain multiple issues, we assume that a text segmentation algorithm has already segmented the threads according to shifts in task



```
1.  Separate thread into issue_email and replies
2.  Create "comparison vector" V representing replies
3.  For each sentence s in issue_email
    3.1 Construct vector representation S for sentence s
    3.2 Compare V and S using cosine similarity
4.  Rank sentences according to their cosine similarity scores
5.  Extract top ranking sentence
```

Figure 3. Framework for extracting discussion Issues.

<u>Methods for building the comparison vector.</u>

- The Centroid method
- The SVD Centroid method.
- The SVD Key Sentence method
- Combinations of methods: Oracles

<u>The Centroid Method:</u>

In the Centroid method, we first build a term by sentence (t × s) matrix, A, from the reply emails. In this matrix, rows represent sentences and columns represent unique words found in the thread. Thus, the cells of a row store the term frequencies of words in a particular sentence. From this matrix, we form a centroid to represent the content of the replies. This is a matter of summing each row vector and normalizing by the number of rows. This centroid is then what we use as our comparison vector

<u>Dataset</u>

The test data used was a portion of the Columbia ACM Student Chapter corpus. This corpus included a total of 300 threads which were constructed using message-ID information found in the header. On average, there were 190 words per thread and 6.9 sentences in the first email.

A Publicly Available Annotated Corpus for Supervised Email Summarization

https://www.researchgate.net/publication/228675520_A_publicly_available_annotated_corpus_for_supervised_email_summarization#pf5
(Dataset Annotation)

- Abstractive and extractive approaches rely on features of sentences to determine which are the most important sentences to keep in a summary
- Among the two types of summarization, abstractive and extractive, there has been much more work on extractive summarization for email threads
- One approach is to find a good feature and use it to choose the most important sentences
- Another extractive summarization technique uses machine learning to decide among a combination of features for scoring the importance of email sentences. In order to train the machine learning method, annotated email data needs to be available. Although an annotated corpus is necessary for evaluation in both methods, more data is required for machine learning as the evaluation data needs to be different from the training data. The machine learning approach to email thread summarization presented in (Rambow et al. 2004) relied on the Ripper algorithm. They show that using email specific features, such as the number of recipients and the number of responses, improves summarization results compared to just multi-document text features, such as centroid similarity and length.
- Other research on email summarization by (CorstonOliver et al. 2004) has focused on task oriented summaries which combine extractive and abstractive techniques. **Important sentences were extracted using an SVM and then reformulated into a task oriented imperative. In this way tasks could automatically be populated into the Outlook" to-do" list.**

Datasets

- *Enron Corpus*: The dataset contains over 30,000 threads, but they end up being rather short with an average thread size of just over four and a median of two
- The Enron corpus has previously been annotated for summarization (Carenini, Ng, and Zhou 2007). A subset of threads were selected, and important sentences were ranked by annotators in a user study. Each thread was summarized (using extraction) by 5 annotators which were asked to select 30% of the original number of sentences and label them as essential or optional
- 39 threads were annotated in this way to produce a corpus for evaluation.
- This annotation is targeted towards extractive summarization and it is hard to use with an abstractive summarization technique. It also has several arbitrary restrictions

including the requirement of 30% sentence selection as well as only two choices for sentence importance

- The work of (CorstonOliver et al. 2004) and (Rambow et al. 2004) are two examples of using annotated email thread corpora to train summarization algorithms.
- Abstractive summarization is the goal of many researchers since it is what people do naturally. However extractive summarization has been more successful and effective since it is easier to compute. A key goal is to successfully combine extraction with abstraction. In order to do this a corpus is needed that supports such a goal
- The main drawback of both the **W3C** and **CSIRO corpora** for summarization annotation purposes is that the content of the data can be quite technical, posing a problem for annotators unfamiliar with the organizations and the subjects discussed in the data. However, there is a sufficient number of emails in each corpus that the data could be filtered to create corpora subsets comprised of less technical emails

Email Classification

In the original work, emails were classified as *Propose, Request, Commit, Deliver, Meeting*, and *deliveredData*. These categories are not mutually exclusive. We propose to use a subset of these speech acts which can not be computed automatically and are information rich. The annotation would consist of the following categories: *Propose, Request, Commit, Agreement/Disagreement* and *Meeting*. A *Propose message* proposes a joint activity; a *Request* asks the recipient to perform an activity; a *Commit message* commits the sender to some future course of action; an *Agreement/Disagreement* is an agreement or disagreement with a joint activity; and a *Meeting message* is regarding a joint activity in time or space. *Deliver* is excluded because most emails deliver some sort of information and *deliveredData* is excluded because *attachments or hyperlinks* can be automatically parsed from an email message.

Discovering and Summarizing Email Conversations

https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.112.99&rep=rep1&type=pdf

Summarizing Email Conversations with Clue Words

- We propose using the **fragment quotation graph** to capture conversations. Based on an analysis of the quotations embedded in emails, the graph provides a fine represen
- We propose an email summarization method, called **ClueWordSummarizer** (CWS), based on a novel concept called clue words. A clue word from a node is a word (modulo stemming) that appears also in its parent node(s) and/or child node(s) in the quotation graph. It is important to note that a clue word takes into account simultaneously (part of) the content and the structure of the quotation graph. Moreover, CWS is unsupervised and can produce summaries of any size as requested by the user.
- A **clue word** in node (fragment) F is a word which also appears in a semantically similar form in a parent or a child node of F in the fragment quotation graph.
- Other algorithms used: MEAD and RIPPER

GIST-IT: Summarizing Email Using Linguistic Knowledge and Machine Learning

https://www.aclweb.org/anthology/W01-1011.pdf

- Our novel approach first extracts simple noun phrases as candidate units for representing document meaning and then uses machine learning algorithms to select the most prominent ones. This combined method allows us to generate an informative, generic, "at-a-glance" summary.
- In this paper, we show: (a) the efficiency of the linguistic approach for phrase extraction in comparing results with and without filtering techniques, (b) the usefulness of vector representation in determining proper features to identify contentful information, (c) the benefit of using a new measure of TF*IDF for the noun phrase and its constituents, (d) the power of machine learning systems in evaluating several classifiers in order to select the one performing the best for this task.
- In this paper, we propose a novel technique for summarization that combines the linguistic approach of extracting simple noun phrases as possible candidates for document extracts, and the use of machine learning algorithms to automatically select the most salient ones.

Dataset

we used three different data collections: Brown corpus, Wall Street Journal, and a set of 4000 email messages (most of which were collected during a conference organization). Our algorithm was a simple one: we extracted all the nouns that appear in front of the preposition "of" and then sorted them by frequency of appearance in all three corpora and used a threshold to select the final list. We generated a set of 141 empty nouns that we used in this forth step of filtering process.

Features

We select six features that we consider relevant in association with the whole NP:

- *np_tfidf* – the TF*IDF measure associated with the whole NP.
- *np_focc* - The first occurrence of the noun phrase in the document.
- *np_length_words* - Noun phrase length measured in number of words, normalized by dividing it with the total numbers of words in the candidate NPs list.
- *np_length_chars* - Noun phrase length measured in number of characters, normalized by dividing it with the total numbers of characters in the candidate NPs list.
- *sent_pos* - Position of the noun phrase in sentence: the number of words that precede the noun phrase, divided by the sentence length. For noun phrases in the subject line and headlines (which are usually short and will be affected by this measure), we consider the maximum length of sentence in document as the normalization factor.
- *par_pos* - Position of noun phrase in paragraph, same as sent_pos, but at the paragraph level.

Other Resources

- Evaluation metric
  https://en.wikipedia.org/wiki/ROUGE_(metric)
- Encoder-Decoder with Attention
  https://towardsdatascience.com/text-summarization-using-deep-learning-6e379ed2e89c