Summarization

CMSC 473/673 Spring 2016
Bryan Wilkinson
● Paperno, Dennis and Baroni, Marco. 2016. When the Whole is Less Than the Sum of Its Parts
Types of Summaries

● Summarization is producing a shortened version of a text that preserves the most important meanings and information.

● When asked to give a summary we often think of a short text, still written in prose, but other outputs are also considered summaries.

● Abstracts
  ○ Probably the most similar to our intuition, normally for scientific articles, but applies to anything.

● Headlines
  ○ An extreme summary, down to one sentence, or even a phrase.

● Snippets
  ○ Partial sentences grabbed from text to show in a search interface.

● Outlines of documents
What are we Summarizing and Why?

● What
  ○ Single Document or Multi Document
  ○ Should we produce a summary for each book in the Harry Potter series, or one for the series as a whole?

● Why
  ○ Who is our audience? General public or someone looking for an answer?
  ○ If they are seeking the answer to a question, this might change what is considered important
    ■ Should we mention Harry is a wizard in our summary of book 2?
How to Summarize

1. Pick out the most important sentences and phrases
2. Place these sentences in an order
3. Generate the summary from the ordered sentences

- There are two ways produce the summarized text in step 3
  - Extraction - Just paste the extracted sentences together
    - Easy
    - Not good looking
  - Abstraction or generation - Create new sentences that have the same meaning
    - Hard
    - Makes a good summary
Picking the Important Parts of a Document

- Pick the first sentence of every paragraph
  - Paragraph segmentation can actually be pretty tricky
- Use techniques from IR to determine which sentences contain important words
- Use graph based measures, like centrality or PageRank
- Cluster sentences and pick from $x$ from each cluster
- Supervised Machine Learning
Mr. and Mrs. Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. Mr. Dursley was the director of a firm called Grunnings, which made drills. The Dursleys had everything they wanted, but they also had a secret, and their greatest fear was that somebody would discover it. When Mr. and Mrs. Dursley woke up on the dull, gray Tuesday our story starts, there was nothing about the cloudy sky outside to suggest that strange and mysterious things would soon be happening all over the country.
After long wait, D.C. United gets final go-ahead to build new stadium

D.C. United on Thursday received final approval to build a soccer stadium in the District, ending more than a dozen years of frustration to replace RFK Stadium as its home and creating an economic pathway to help the team catch up to the rest of a fast-growing pro league. Two months after giving conditional support, the five-member D.C. Zoning Commission was unanimous in approving the 14-acre project at Buzzard Point in Southwest, adjacent to Fort McNair and three blocks from Nationals Park at the confluence of the Anacostia River and Washington Channel. The city had previously agreed to cover $150 million in land acquisition and infrastructure costs.
Similarly to word-context vectors (may be older) it is common to represent documents with vectors.

A term document matrix calculates how many times each word appears in each document.
- Terms that appear only in a few documents are more “important” in the documents they appear in.
- Documents that have similar terms are similar.

Because certain terms occur so often in almost all documents, we exclude them all together.
- Known as stopwords, examples are *a, the, only, was, for, in*, etc.
Document 1: Mr and Mrs Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much.

Document 2: Not for the first time, an argument had broken out over breakfast at number four, Privet Drive.

Document 3: Harry Potter was a very unusual boy in many ways.
## Sidebar: Term Document Matrices Example

Document 1: Mr Mrs Dursley number four Privet Drive proud say perfectly normal thank much

Document 2: Not first time argument broken breakfast number four Privet Drive

Document 3: Harry Potter unusual boy many ways

<table>
<thead>
<tr>
<th></th>
<th>Mr</th>
<th>Mrs</th>
<th>Dursley</th>
<th>number</th>
<th>four</th>
<th>privet</th>
<th>drive</th>
<th>proud</th>
<th>say</th>
<th>perfectly</th>
<th>normal</th>
<th>thank</th>
<th>much</th>
<th>not</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### Sidebar: Term Document Matrices Example

**Document 1:** Mr Mrs Dursley number four Privet Drive proud say perfectly normal thank much

**Document 2:** Not first time argument broken breakfast number four Privet Drive

**Document 3:** Harry Potter unusual boy many ways

<table>
<thead>
<tr>
<th></th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>D2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>D3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
We want to highlight the terms that are relatively unique to a document

The most common way to do this is Term Frequency - Inverse Document Frequency (TF-IDF)

TF-IDF of word $i$ in document $j = c(w_{ij}) \times \log \left( \frac{D}{d(w_i)} \right)$
Sidebar: Weighting

- We want to highlight the terms that are relatively unique to a document
- The most common way to do this is Term Frequency - Inverse Document Frequency (TF-IDF)

\[
\text{TF-IDF of word } i \text{ in document } j = c(w_{ij}) \times \log\left( \frac{D}{d(w_i)} \right)
\]

<table>
<thead>
<tr>
<th></th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
To determine which sentence has the most important words, we set every term with a TF-IDF greater than some threshold, $t$, to 1, and less than $t$ to 0.

- The score of the sentence is the average weight of all words in it.
- Other weight schemes like log-likelihood ratio also work very well.
In this set of sentences, the most important sentence is sentence 3,

Harry Potter was a very unusual boy in many ways.

In a real application, the first $n$ sentences are selected, or sentences with scores greater than a second threshold.
● In this set of sentences, the most important sentence is sentence 3,

Harry Potter was a very unusual boy in many ways.

● In a real application, the first $n$ sentences are selected, or sentences with scores greater than a second threshold.

\[ D_1 = \frac{9}{13} = 0.6923 \]

\[ D_2 = \frac{6}{10} = 0.6 \]

\[ D_3 = \frac{6}{6} = 1 \]
Graph Based Extraction Methods

- Another very common method is to let each sentence be a node, and connect sentences according to different features.
- Some common features are:
  - Percent Overlap
  - Coreference (Later Lecture)
  - Discourse Relations (Later Lecture)
In Graph Theory, the centrality of a node is a way to measure its importance.

- There are many different variations
- Today we will consider one very similar to closeness centrality, based on distance between nodes

A simple definition from SLP2 using sentence vectors is

\[
\text{centrality}(x) = \frac{1}{K} \sum_{y=1}^{K} \cos(x, y)
\]
Centrality Example

- Some of the cosine similarities between the first 14 sentences in The Sorcerer's Stone

\[
\begin{array}{cccccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\
0 & 1.00 & 0.00 & 0.11 & 0.00 & 0.07 & 0.00 & 0.00 & 0.13 & 0.10 & 0.00 & 0.00 & 0.10 & 0.14 \\
1 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.10 & 0.03 & 0.00 & 0.00 & 0.05 & 0.13 & 0.00 \\
2 & 0.11 & 0.00 & 1.00 & 0.00 & 0.00 & 0.12 & 0.00 & 0.00 & 0.05 & 0.00 & 0.00 & 0.00 & 0.08 & 0.12 \\
3 & 0.00 & 0.00 & 0.00 & 1.00 & 0.07 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
4 & 0.07 & 0.00 & 0.04 & 0.07 & 1.00 & 0.00 & 0.00 & 0.00 & 0.10 & 0.08 & 0.00 & 0.00 & 0.00 & 0.05 & 0.08 \\
\end{array}
\]

\[
\text{centrality}(x) = \frac{1}{K} \sum_{y=1}^{K} \cos(x, y)
\]
Centrality Example

- The centralities of the first 14 sentences in *The Sorcerer's Stone*

<table>
<thead>
<tr>
<th>S</th>
<th>Centrality</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.12</td>
</tr>
<tr>
<td>1</td>
<td>0.09</td>
</tr>
<tr>
<td>2</td>
<td>0.11</td>
</tr>
<tr>
<td>3</td>
<td>0.08</td>
</tr>
<tr>
<td>4</td>
<td>0.11</td>
</tr>
<tr>
<td>5</td>
<td><strong>0.14</strong></td>
</tr>
<tr>
<td>6</td>
<td>0.09</td>
</tr>
<tr>
<td>7</td>
<td>0.13</td>
</tr>
<tr>
<td>8</td>
<td>0.12</td>
</tr>
<tr>
<td>9</td>
<td>0.13</td>
</tr>
<tr>
<td>10</td>
<td>0.13</td>
</tr>
<tr>
<td>11</td>
<td>0.12</td>
</tr>
<tr>
<td>12</td>
<td>0.11</td>
</tr>
<tr>
<td>13</td>
<td>0.12</td>
</tr>
</tbody>
</table>
The most central sentences are:

The Dursleys had a small son called Dudley and in their opinion there was no finer boy anywhere.

The Dursleys shuddered to think what the neighbors would say if the Potters arrived in the street.

The Dursleys knew that the Potters had a small son, too, but they had never even seen him.
Supervised Sentence Extraction

- Given a corpus of document summary pairs
- Train a classifier to label if a sentence should be part of a summary or not
  - Encode each sentence as a feature vector based on features referring back to the summary examples
  - If a whole sentence is in the summary that is a good indicator
  - If a sentence partially matches on in the summary it is still a pretty good indicator
  - Still can use other features like entities and such
- Has been trained using both Naive Bayes Classifiers, and HMMs
- More recent research work treats summarization like translation
  - Translate full article into summary
Sentence Ordering

- It is quite common to just leave the sentences in the order they were originally
  - Not always a great idea
- Much more studied in the multi document setting
  - With multiple documents, we can study the “average” order of the word
  - We can also take advantage of things like timestamps
- The general idea is to make a text order coherent
  - Entities occurring in adjacent sentences may help this.
Entity Based Ordering

  - Requires Named Entity Recognition
- Create a matrix of size N sentences by E entities
- In each entry of the matrix, place an S if that entity was a subject in that sentence, an O if it was an object, and an E otherwise
  - This means dependency parsing is probably useful too.
- Use this to create a feature vector which is then fed to a trained machine learning algorithm to judge how coherent the text is
- Pick the sentence order that produces the most coherent text.
Summary Generation

- The simplest method is to output the ordered sentences as the summary
  - This doesn’t always look the nicest
- Numerous other techniques exist to massage a list of sentences into something more natural
  - This isn’t necessary in the summarization as translation paradigm because entirely new sentences may be generated.
- Compress the sentences
  - Remove extraneous or redundant information from the individual sentences
- Fuse the sentences
  - Combine two or more sentences into something more natural
Sentence Compression

● Rule Based
  ○ One method is to find the main verb of the sentence as well as the subject and object
    ■ These have to be kept
    ■ Also keep arguments of the verb
      ● Found in a Semantic Lexicon
    ■ Delete phrases based on corpus statistics as well as number of overlaps to other parts of the document
  ○ Can also make up rules that involve always deleting certain words or their arguments
    ■ Appositives
    ■ Attributions

● Statistically Based
  ○ Generate a parse tree for the sentence
    ■ Drop out phrases and recalculate the probability of the tree.
Sentence Fusion

- **Cut-and-paste**
  - Based on observations on how professional summarizers work
  - Given two sentences parse them
    - Substitute a subtree from one into another parse tree
    - Use some heuristics to figure out where

- **More Sophisticated Methods**
  - Align of a group of sentences
  - Select one sentence to be the base sentence
  - Combine similar phrases in the other sentences and insert into the base sentence
Multi-Document Summarization

- Summarize a group of documents
  - Assume the group of documents has already been selected and thus all relevant

- Same basic parts
  - Select important sentences
  - Order sentences
  - Generate summary

- Some specific challenges
  - Avoiding selecting the same information from all documents
  - Possibly a lot more data to summarize
Avoiding Redundancy

- Keep track of the sentences extracted so far
  - Down weight sentences that are very similar to ones already extracted

- Clustering
  - Cluster sentences based on vectors
  - Select one representative sentence of that cluster
  - Or use sentence fusion to combine them together

- Reduce redundancy post extraction
  - Extract all sentences as before
  - Use a second pass of selection just over those sentences
The primary evaluation metric for summarization is ROUGE

- Inspired by Machine Translation metric BLEU
- Calculate the overlap in n-grams between an automatically produced summary and a set of reference summaries

$$\text{ROUGE-2} = \frac{\sum_{S \in \text{RefSummaries}} \sum_{\text{bigram} \in S} \text{Count}_{\text{match}}(\text{bigram})}{\sum_{S \in \text{RefSummaries}} \sum_{\text{bigram} \in S} \text{Count}(\text{bigram})}$$
Evaluation Example

- Automatic Summary: Harry potter was an unusual boy.
- Reference 1: Harry Potter was weird.
- Reference 2: The unusual boy’s name was Harry Potter.

\[
\text{ROUGE-2} = \frac{\sum_{S \in \text{RefSummaries}} \sum_{\text{bigram} \in S} \text{Count}_{\text{match}}(\text{bigram})}{\sum_{S \in \text{RefSummaries}} \sum_{\text{bigram} \in S} \text{Count}(\text{bigram})}
\]
Evaluation Example

- Automatic Summary: Harry Potter was an unusual boy.
- Reference 1: Harry Potter was weird.
- Reference 2: The unusual boy was named Harry Potter.

\[(\text{Harry, Potter}) (\text{Potter, was}) (\text{was, an}) (\text{an, unusual}) (\text{unusual, boy}) (\text{boy, .})\]
\[(\text{Harry, Potter}) (\text{Potter, was}) (\text{was, weird}) (\text{weird, .})\]
\[(\text{The, unusual}) (\text{unusual, boy}) (\text{boy, was}) (\text{was, named}) (\text{named, Harry}) (\text{Harry, Potter}) (\text{Potter, .})\]

Count\textsubscript{match} = 2, 2
Count = 4, 8

ROUGE-2 = \(\frac{4}{12} = 0.33333\)

\[
\text{ROUGE-2} = \frac{\sum_{S \in \text{RefSummaries}} \sum_{\text{bigram} \in S} \text{Count}_{\text{match}}(\text{bigram})}{\sum_{S \in \text{RefSummaries}} \sum_{\text{bigram} \in S} \text{Count}(\text{bigram})}
\]
Summarization Examples

- Many taken from the appendix of [Automatic Text Summarization](http://autosummarizer.com/index.php)
- [http://www.extractorlive.com/upload_demo.html](http://www.extractorlive.com/upload_demo.html)
- [http://freesummarizer.com/](http://freesummarizer.com/)
- [http://smmry.com/](http://smmry.com/)
- [http://textcompactor.com/](http://textcompactor.com/)

\[
\text{ROUGE-2} = \frac{\sum_{S \in \text{RefSummaries}} \sum_{\text{bigram} \in S} \text{Count}_{\text{match}}(\text{bigram})}{\sum_{S \in \text{RefSummaries}} \sum_{\text{bigram} \in S} \text{Count}(\text{bigram})}
\]