Visualizing Hillary Clinton's Emails

Yihe Chen yc3076 | Palmer Lao pol2105 | Daitong Li dl2991 | Ziyue Shuai zs2285 | Eric Zhang ez2232

December 6, 2016

1 Introduction

2 Objectives

3 Data Source

4 Methodology

4.1 An Overview of our Summarization Procedure for one Document

Because our end goal is to create a visual dashboard for our users, we have to be careful about our bookkeeping while we slice and dice our text data. A brief overview of our procedure from end-to-end for one e-mail is as follows:

1. Split the e-mail into sentences
2. Clean each sentence
3. Represent each sentence as a bag-of-words
4. Run TextRank to get a ranking of the bags-of-words
5. Return the top k sentences from the original e-mail associated with the top k ranking bags-of-words

Of particular importance to our implementation is keeping track of how the bags-of-words are associated with the original sentences from the e-mail, as the bags-of-words are not nearly as informative to the user.

4.2 The TextRank Algorithm

We treat each e-mail as a self-contained document and attempt to summarize each document using TextRank [2], which is a derivative of PageRank [3] adapted to units of text instead of web pages.

The PageRank algorithm is a way of ranking web pages based on the way that they link to each other. The algorithm is based on a random walk model of the typical internet user. We assume the user starts on some arbitrary web page, and then either randomly clicks links on the current page or with some re-seeding probability, d, navigates to another random web page that was not necessarily linked to from the current one. The process of such a user's browsing history is clearly Markovian under this assumption. The addition of the re-seeding probability makes this Markov process irreducible and aperiodic, or equivalently, ergodic [4]. This ergodicity property tells us that there exists a so-called stationary distribution over the set of all web pages.

The stationary distribution of an ergodic Markov chain has many interpretations. One is that the distribution assigns probability mass to each state, and this mass is equivalent to the probability of the Markov process to be observed in some state after the chain has mixed or lost its dependence on its initial position. The other is the "time-average" interpretation: the average proportion of time-steps a process will spend in some state tends to this state's mass in the stationary distribution.

Web pages that are assigned a large probability mass are thought to be relatively important in the sense that a random web surfer is more likely to visit it. One can sort the pages by the amount of mass assigned to them by the stationary distribution to get a ranking of pages by this notion of importance.

4.3 Cleaning the e-mails

Many similarity functions [1] as well as the one we implemented rely on metrics defined on bag-of-words representations of text. While the bag-of-words representation is a popular one, it's well

known to have many deficiencies that are exacerbated by carelessly processed text.

One consequence of using a bag-of-words is that words that are not a character-for-character match will not be counted as the same. Much of our text cleaning effort is dedicated to ensuring that words that are indicative of sentence similarity are mapped to the same word, and that words that are not indicative of sentence similarity are removed.

The very first part of cleaning a sentence is simple: we set all of the characters to lowercase, as the capitalization of a word should, in most cases, not change its meaning.

Similarly, we remove any nonalphanumeric symbols from our text, as it seems that the method used to extract the text from the e-mails into a database left many wayward symbols (e.g.`\n`, the newline symbol). Any nonalphanumeric symbols were turned into spaces, and any extraneous whitespace was subsequently deleted.

The next transformation we apply is removing stop words. Stop words are words that almost every valid sentence in the English language contains, such as "the", "a", "as", or "for". Because stop words are so common, leaving them in the sentences would likely inflate the similarity between many bags-of-words. Leaving stop words in our text would effectively dilute our measure of how important a sentence is.

Our last problem is that different conjugations of words such as "run" and "running" will be counted as different although they ostensibly are referring to the same activity. We can attempt to mitigate this problem by applying a popular text cleaning procedure called stemming [5], which attempts to reduce all words to their roots. For example, "running" would be reduced to "run", and "run" would stay the same. This allows us to measure text like "I am running for president" and "a run for office" as similar despite the fact that (after removing stopwords) none of the words are a character-for-character match. The `R` implementation we used is from the `tm` package, which implements the standard Porter stemming algorithm.

5 Results

6 Validation

7 Conclusion

References

[1] Barrios, F., López, F., Argerich, L., and Wachenchauzer, R. Variations of the similarity function of textrank for automated summarization. *arXiv preprint arXiv:1602.03606* (2016).

[2] Mihalcea, R., and Tarau, P. Textrank: Bringing order into texts. Association for Computational Linguistics.

[3] Page, L., Brin, S., Motwani, R., and Winograd, T. The pagerank citation ranking: bringing order to the web.

[4] Ross, S. M. *Introduction to Probability Models, Tenth Edition*, 10 ed. 2009.

[5] Willett, P. The porter stemming algorithm: then and now. *Program 40*, 3 (2006), 219–223.