SIG NLL

10/29: Text Summarization

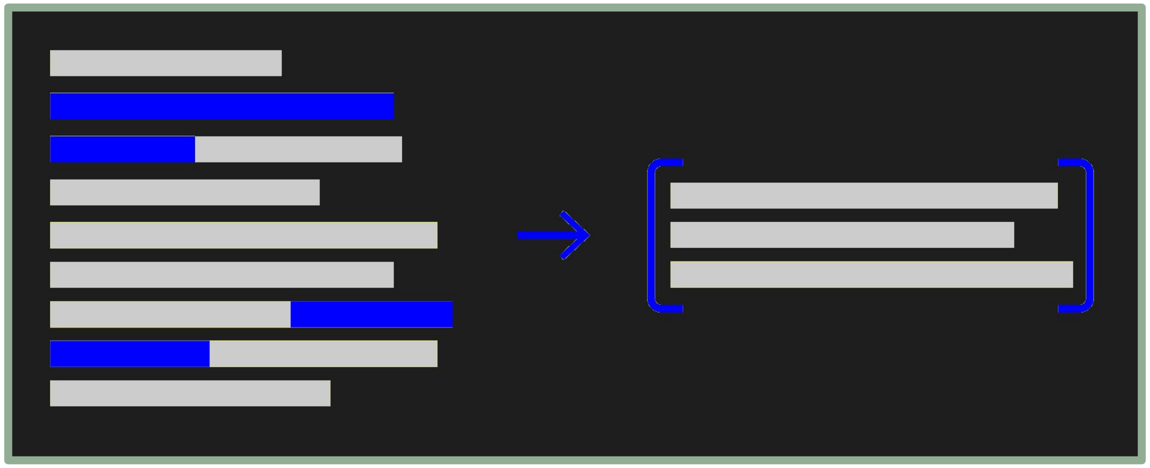
Discord: https://discord.gg/68VpV6

Text Summarization

How can we do it and why is it useful?

What is text summarization?

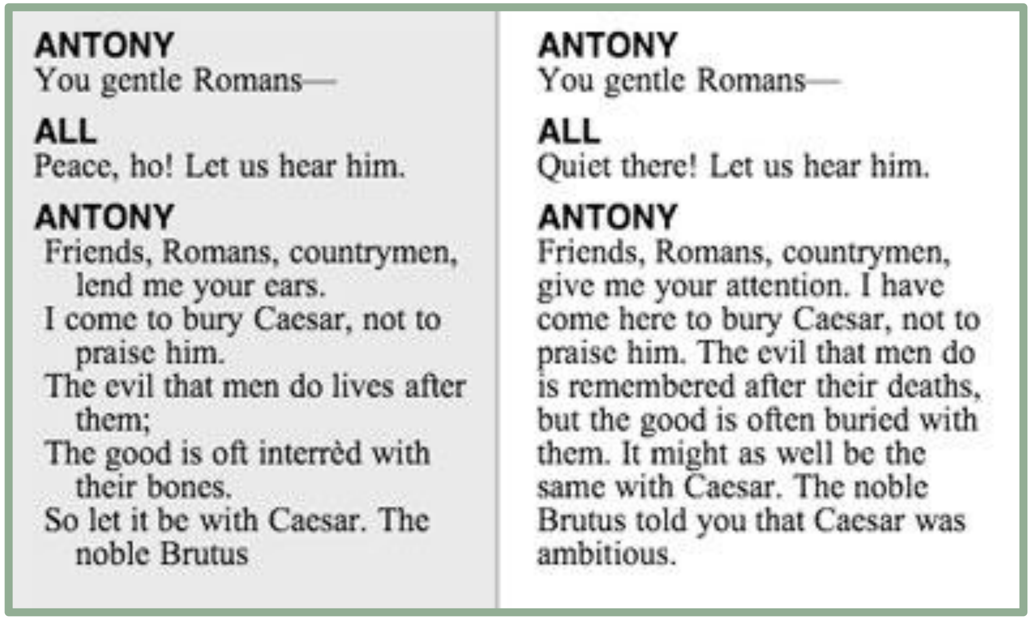
“The process of shortening a set of data, computationally, to create a subset (a summary) that represents the most important or relevant information within the original content”



Where is it used?

Some common applications of text summarization:

- Newspaper headlines
- Movie previews
- Abridgments (no fear shakespeare)



ANTONY You gentle Romans—	**ANTONY** You gentle Romans—
ALL Peace, ho! Let us hear him.	**ALL** Quiet there! Let us hear him.
ANTONY Friends, Romans, countrymen, lend me your ears. I come to bury Caesar, not to praise him. The evil that men do lives after them; The good is oft interrèd with their bones. So let it be with Caesar. The noble Brutus	**ANTONY** Friends, Romans, countrymen, give me your attention. I have come here to bury Caesar, not to praise him. The evil that men do is remembered after their deaths, but the good is often buried with them. It might as well be the same with Caesar. The noble Brutus told you that Caesar was ambitious.

Summarization Techniques

Extractive Summarization

- ▷ Creates a summary using already existing sentences
- ▷ Relies heavily on statistical models

Abstractive Summarization

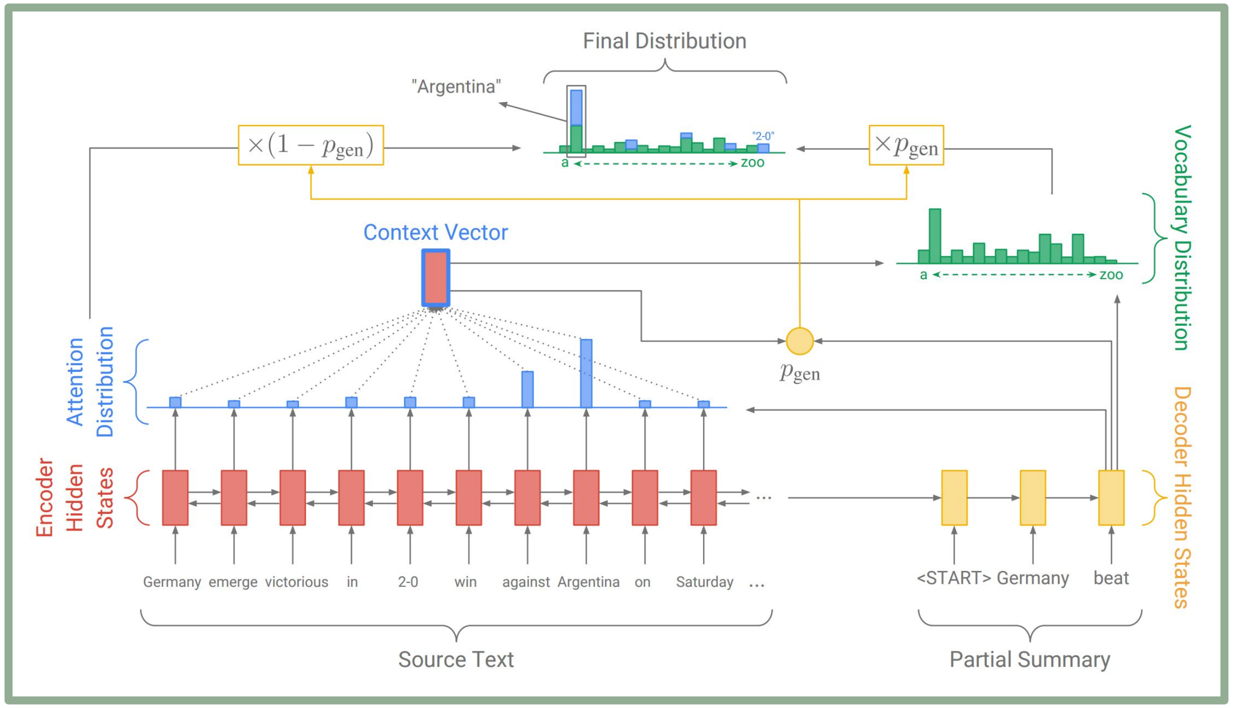
- ▷ Creates a summary using new and unique sentences
- ▷ Relies heavily on artificial intelligence

We'll be focusing on **extractive summarization**

Why not Abstractive Summarization?

Compared to extractive summarization, abstractive summarization is much more complicated and requires a thorough understanding of machine learning and NLP

Architecture for abstractive summarization:



Extractive Summarization

How do we create this model?

Summarization General Algorithm

In general, our steps are to:

1. Load in a data set
2. Create a frequency dictionary
3. Find weighted frequency of each word
4. Calculate weighted sentence frequencies
5. Select the most important sentences

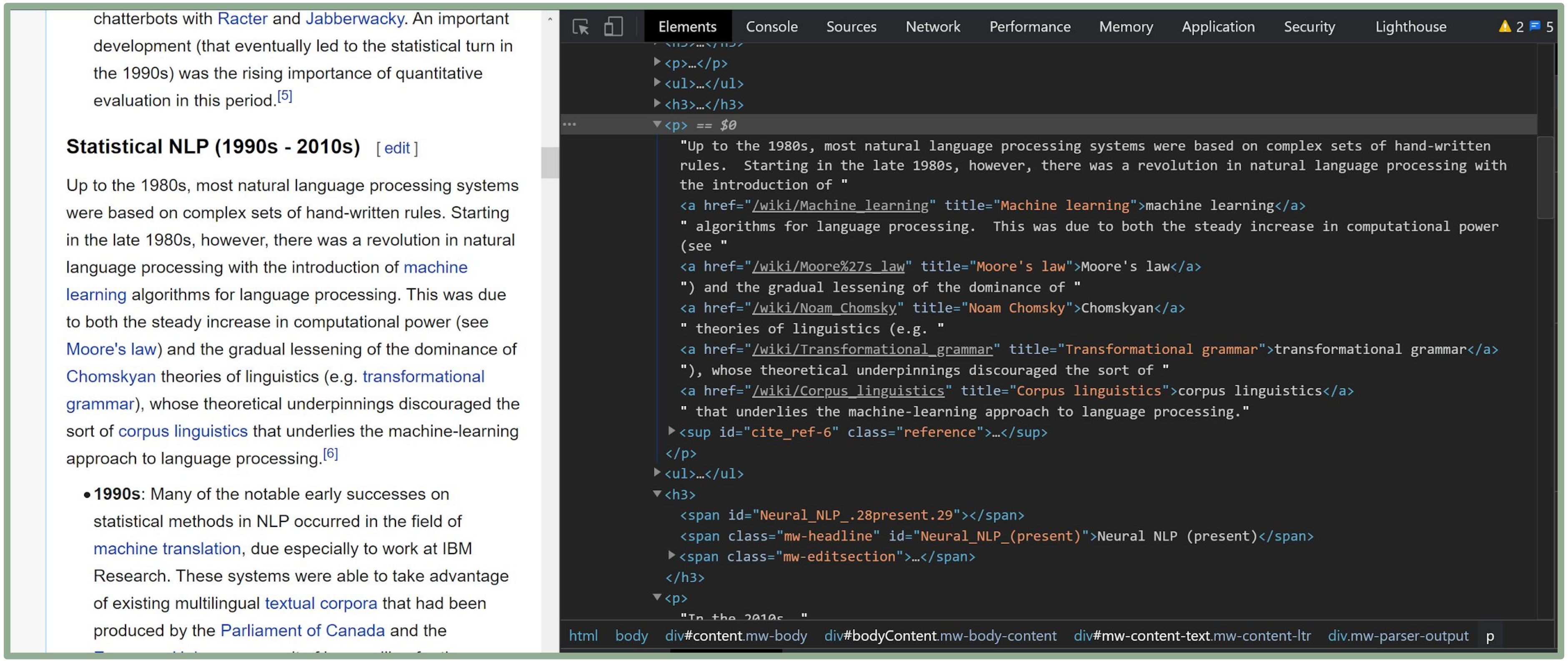
Loading Data: Web Scraping

In the notebook we extract our data from Wikipedia

To make this work for any Wikipedia article, we'll use a technique called **web scraping**

Loading Data: Web Scraping

Essentially, we'll use a library to grab the HTML of a web page, and then parse the data we need from the article



chatterbots with Racter and Jabberwacky. An important development (that eventually led to the statistical turn in the 1990s) was the rising importance of quantitative evaluation in this period.[5]

Statistical NLP (1990s - 2010s) [edit]

Up to the 1980s, most natural language processing systems were based on complex sets of hand-written rules. Starting in the late 1980s, however, there was a revolution in natural language processing with the introduction of machine learning algorithms for language processing. This was due to both the steady increase in computational power (see Moore's law) and the gradual lessening of the dominance of Chomskyan theories of linguistics (e.g. transformational grammar), whose theoretical underpinnings discouraged the sort of corpus linguistics that underlies the machine-learning approach to language processing.[6]

- **1990s**: Many of the notable early successes on statistical methods in NLP occurred in the field of machine translation, due especially to work at IBM Research. These systems were able to take advantage of existing multilingual textual corpora that had been produced by the Parliament of Canada and the

```
▸<p>…</p>
▸<ul>…</ul>
▸<h3>…</h3>
▾<p> == $0
   "Up to the 1980s, most natural language processing systems were based on complex sets of hand-written
   rules.  Starting in the late 1980s, however, there was a revolution in natural language processing with
   the introduction of "
   <a href="/wiki/Machine_learning" title="Machine learning">machine learning</a>
   " algorithms for language processing.  This was due to both the steady increase in computational power
   (see "
   <a href="/wiki/Moore%27s_law" title="Moore's law">Moore's law</a>
   ") and the gradual lessening of the dominance of "
   <a href="/wiki/Noam_Chomsky" title="Noam Chomsky">Chomskyan</a>
   " theories of linguistics (e.g. "
   <a href="/wiki/Transformational_grammar" title="Transformational grammar">transformational grammar</a>
   "), whose theoretical underpinnings discouraged the sort of "
   <a href="/wiki/Corpus_linguistics" title="Corpus linguistics">corpus linguistics</a>
   " that underlies the machine-learning approach to language processing."
  ▸<sup id="cite_ref-6" class="reference">…</sup>
 </p>
▸<ul>…</ul>
▾<h3>
   <span id="Neural_NLP_.28present.29"></span>
   <span class="mw-headline" id="Neural_NLP_(present)">Neural NLP (present)</span>
  ▸<span class="mw-editsection">…</span>
 </h3>
▾<p>
```

html
body
div#content.mw-body
div#bodyContent.mw-body-content
div#mw-content-text.mw-content-ltr
div.mw-parser-output
p

Creating a Frequency Dictionary

Using the same dictionary techniques we've discussed, we'll create a dictionary where the keys are the words and the values are the number of occurrences of the word

```
"Sans teeth. Sans eyes. Sans taste.
 Sans everything."
```

```
{"sans" : 4, "teeth" : 1, "eyes" : 1,
 "taste" : 1, "everything" : 1}
```

Word Weighted Frequencies

The weighted frequency of a word is a measure of how occurrent a word is compared to all other words

We use the following formula to calculate the weighted frequency of each word w:

$$weight_w = \frac{frequency_w}{max(frequency)}$$

Word Weighted Frequencies Example

Using our dictionary from earlier:

```
{"sans" : 4, "teeth" : 1, "eyes" : 1,
 "taste" : 1, "everything" : 1}
```

```
max_frequency = 4 #(from sans)
```

Our new frequency dictionary will be:

```
{"sans" : 1, "teeth" : 0.25, "eyes" :
 0.25, "taste" : 0.25, "everything" : 0.25}
```

Sentence Frequencies

To get the weighted frequency of each sentence, we just need to take the sum of the weight of the words in the sentence

```
sentence_freq = 0
for word in sentence:
    sentence_freq += weight[word]
```

Note: We still need to clean/tokenize each word like we have been doing to make sure we get the correct match in the dictionary

Sentence Frequencies Example

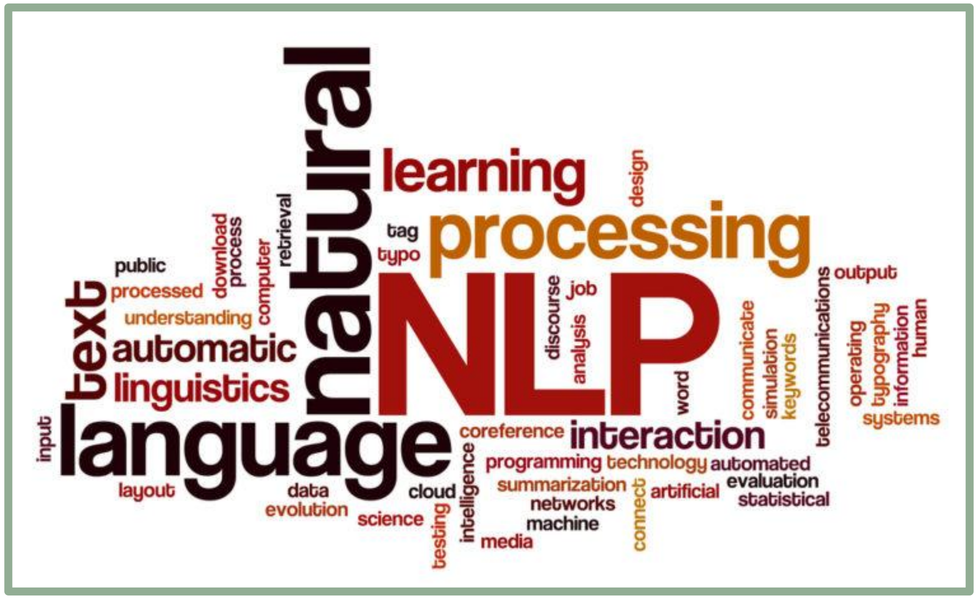
```
{"sans" : 1, "teeth" : 0.25, "eyes" :
 0.25, "taste" : 0.25, "everything" : 0.25}
```

```
["Sans teeth",                  [1 + 0.25,
 "Sans eyes",           →        1 + 0.25,
 "Sans taste",                   1 + 0.25,
 "Sans everything"]              1 + 0.25]
                                     ↓
      frequencies = [1.25, 1.25, 1.25, 1.25]
```

Creating a Summary

Now that we have a metric to determine how good our sentences are, how do we create our summary?

The most intuitive thing to do is to sort by our sentence frequencies and choose the top couple of sentences



Creating a Summary Example

A taco can be made with a variety of fillings, including beef, pork, chicken, seafood, beans, vegetables, and cheese, allowing for great versatility and variety. A taco is a traditional Mexican dish consisting of a small hand-sized corn or wheat tortilla topped with a filling. The tortilla is then folded around the filling and eaten by hand.

```
sentence_weights = [1.5, 3, 0.6]
```

If we only chose one sentence as our summary, we would chose the second one, with the highest weight of 3

A taco is a traditional Mexican dish consisting of a small hand-sized corn or wheat tortilla topped with a filling.

Flaws of Our Model

What problems could our model run into?

Really Long Sentences

Suppose we’re looking at the Wikipedia article for machine learning and we come across this sentence:

Tom M. Mitchell provided a widely quoted, more formal definition of the algorithms studied in the machine learning field: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E." This definition of the tasks in which machine learning is concerned offers a fundamentally operational definition rather than defining the field in cognitive terms

There’s a good chance that this sentence will have a high frequency because its so long

Really Long Sentences: Solution

Since we’re creating a summary, it doesn’t really make sense to pick really long sentences anyway

We can reduce the effects of this problem with a constant that limits the length of our sentences

```
length_limit = 30
if (num_words  > 30):
  sentence_weight = 0
else:
  [other calculations]
```

Non-Unique Sentences

Imagine we're trying to create a summary of the Gucci Wikipedia article, and this infamous Lil Pump lyric appears:

```
"Gucci gang, Gucci gang, Gucci gang,
Gucci gang (Gucci gang). Gucci gang,
Gucci gang, Gucci gang (Gucci gang)"
```

This will undoubtedly give us a very high sentence frequency, even though it's not a very good summary

Non-Unique Sentences: Solution

We can fix this problem by taking a weighted average of our weight and our fraction of unique words:

$$ratio_{unique} = \frac{num_{unique}}{num_{words}}$$

$$freq = \alpha(freq)+(1-\alpha)(ratio_{unique})$$

Just like the learning rate in ML, we need to figure out the right value for alpha

Fortunately, this won't happen on Wikipedia, so we don't need to take it into account

Tasks to Complete

1) Work on the notebook (**text_summary.zip)** in the google drive folder https://drive.google.com/drive/folders/1Qr4AaFRgTn8kYkQxHkz2edwGO26MQm4Z?usp=sharing

Try to work on it collaboratively! You might meet some people you could do a project with in the future

As always, let us know if you need any help!

SIG
NLL