Assignment 1 – Getting Started

CS499 - Intro to NLP
Antonis Anastasopoulos

January 2021

Due Date: 2/5/2021 (eod)
Instructions: see at the end of the assignment.

1 Preliminaries (1 credit total)

[1 credit] Answer all four questions as they appear below.

Read in the Hamlet corpus. You can read it easily with

```
with open("hamlet.txt") as inp:
  lines = inp.readlines()
```

Use the NLTK package[1] (which you can install with `pip install nltk`) to split the dataset into sentences. Use the `sent_tokenize()` function. **(a)** How many sentences are there? (ignore empty lines and empty sentences)

Answer: 2355

Now, we want to split into words. First, split the sentences on whitespace (e.g. using the python `.split(' ')` function over each sentence. **(b)** How many words do you find?

Answer: 29605

However, this tokenization approach is not ideal. We are still treating words that are attached to punctuation as different tokens. It's better to do a more informed word tokenization. Use NLTK's `word_tokenize()` function, to split into words. Now the first sentence should have 13 tokens. **(c)** How many tokens are there now?

Answer: 36421

[1] https://www.nltk.org/

Last, lowercase everything, to avoid skewing your counts due to the (arbitrary) rule of uppercasing the first letter of the first word of each sentence.

Compute the number of types and tokens in the corpus. **(d)** How many types and how many tokens appear in Hamlet?

Answer: Tokens: 36421, Types: 4789

2 Herdan's Law(1 credit total)

[**1 credit**] According to Herdan's Law (or Heaps' Law), the relationship between the number of types $|V|$ and number of tokens in N in a corpus should be described by the following equation:

$$|V| = kN^{\beta}.$$

For $\beta = 0.7$ (a standard value typically found for English), what is the value for k, for your lowercased, tokenized Hamlet corpus?

Answer: $k = 3.07$

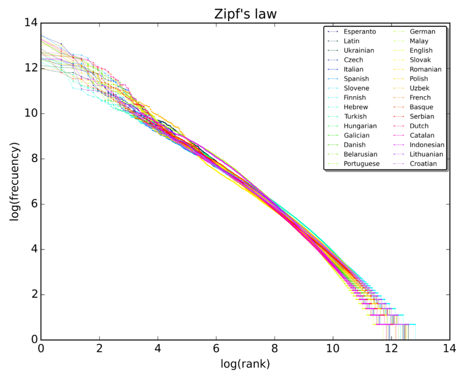
3 Zipf's Law (2 credits total)

Zipf's law[2] states that given a large sample of text, the frequency of any word is inversely proportional to its rank in the frequency table. Zipf's law is most easily observed by plotting the data on a log-log graph, with the axes being the log rank order of a type and log frequency of a type. For example, "the" is the most common type in English, so it should appear at $x = log(1)$.[3] If it appears 70000 times, then its y-axis value shoud be $y = \log(70000)$. The data conform to Zipf's law to the extent that the plot is linear.

For example, look at this chart (from Wikipedia: `https://en.wikipedia.org/wiki/Zipf%27s_law`) that plots the rank versus the frequency for the first 10 million words over 30 wikipedias in a log-log scale:

[2]Named after the linguist George Kingsley Zipf, who first sought to explain it.

[3]Note that "the" is actually the third most common type in the lowercased, tokenized Hamlet corpus, because two other tokens have higher frequencies.



We can describe the law more generally, using a general formula that has parameters that will vary slightly across languages. Formally, let:

- $|V|$ be the number of types (hyperparameter);
- s be the value of the exponent characterizing the distribution (hyperparameter);
- k be the rank of a type.

Zipf's law then predicts that out of a population of N elements, the normalized frequency of the element of rank k, $f(k; s, |V|)$, is:

$$f(k; s, |V|) = \frac{1/k^s}{\sum_{n=1}^{|V|}(1/n^s)}.$$

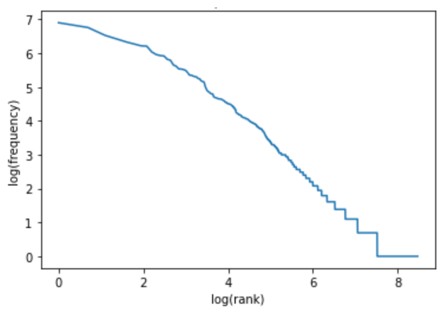
[**1 credit**] Fill the following table based on the (lowercased, tokenized) Hamlet corpus. Use $s = 1$. Report (relative) frequencies up to 5 decimal digits. The observed relative frequency should be the type's frequency as observed in your corpus. The predicted one should be the prediction of the formula above.

Answer:

Type	Rank	Observed Relative Frequency	Predicted Relative Frequency
the	3	0.03683	0.02726
hamlet	54	0.00205	0.00275
royally	3411	0.00003	0.00003
rose	778	0.00014	0.00011
honourable	1003	0.00011	0.00008

[**1 credit**] Create a plot that will illustrate Zipf's Law over your dataset. We want a plot similar to the example above. You can use packages like matplotlib, seaborn, ggplot, etc. Remember to label your axes!

Answer:



4 Edit Distance over Gazetteers (6 credits total)

"Gazetteer" basically means "index". In NLP, we typically refer to them to denote mappings of named entities[4] across languages. The assignment package provides a sample gazetteer with mappings of entities between English and Kinyarwanda. Kinyarwanda is one of the official languages of Rwanda, spoken by at least 12 million people in Rwanda, Congo, and Uganda.

Here's an example of the gazetteer, where each line follows the format `[english] ||| [kinyarwanda]`:

```
Iran ||| Irani
Eggplant ||| Urutoryi
Bible ||| Bibiliya
Flag ||| Ibendera
Rose ||| Iroza
```

Programmatically, you can obtain the contents of the file line by line as follows:

```
with open("gazetteer.eng-kin.txt") as inp:
  lines = inp.readlines()

for l in lines:
  l = l.strip().split(' ||| ')
  eng = l[0].strip()
  kin = l[1].strip()
    ...
```

The goal of this assignment will be to study how similar the English and Kinyarwanda named entities are. To do this, we will rely on the Edit Distance algorithm, as described in Section 3.11 of the SLP book.

[**1 credit**] First, implement a function that computes the minimum edit distance of two strings. Pseudo-code can be found in Figure 3.25 of the book. Print the computation matrix for the pair in the first line of the dataset.

Note: there exist Python packages that implement this function. I highly advise you to implement it from scratch, because you will need to modify it for

[4]A named entity is a real-world object, such as persons, locations, organizations, products, etc., that can be denoted with a proper name.

the following questions. If you are unsure about whether your implementation is correct, you can certainly use such a package to confirm that your outputs are correct.

Answer: From top left to bottom right:

```
   # i r a n i
# [0 1 2 3 4 5]
i [1 0 1 2 3 4]
r [2 1 0 1 2 3]
a [3 2 1 0 1 2]
n [4 3 2 1 0 1]
```

[**1 credit**] What is the average edit distance over the whole dataset? What is the minimum edit distance? What is the maximum edit distance? How many pairs have the minimum, and how many have the maximum edit distance?

Answer: Average edit distace: 2.556, minimum: 0 (396 pairs), maximum: 10 (46 pairs)

[**1 credit**] What is the edit distance for the Kinyarwanda phrases corresponding to the English "Ankara" (the capital of Turkey) and "Ankara Province"? How do you explain the second value?

Answer: 0 and 10. Explanation: kinyarwanda has reverse word order than English for this noun phrase (think of the difference between "Ankara province" and "province of Ankara".

[**1 credit**] Next, implement backtracing (also described in section 3.11 of the book) on top of your function. To do this, follow the instructions of the book:

> In the first step, we augment the minimum edit distance algorithm to store backpointers in each cell. The backpointer from a cell points to the previous cell (or cells) that were extended from in entering the current cell. [...] In the second step, we perform a backtrace. In a backtrace, we start from the last cell (at the final row and column), and follow the pointers back through the dynamic programming matrix. Each complete path between the final cell and the initial cell is a minimum distance alignment.

For the first pair that has a minimum edit distance of 5 (which country is it?), print the edit distance matrix **and** the backtrace (the sequence of cells in the matrix – from top right to bottom left (as shown in Figure 3.27 in the book) – that yields this minimum cost.

Answer:

```
    #  O  s  i  t  a  r  a  l  i  y  a
# [ 0  1  2  3  4  5  6  7  8  9 10 11]
A [ 1  1  2  3  4  5  6  7  8  9 10 11]
```

```
u [ 2  2  2  3  4  5  6  7  8  9 10 11]
s [ 3  3  2  3  4  5  6  7  8  9 10 11]
t [ 4  4  3  3  3  4  5  6  7  8  9 10]
r [ 5  5  4  4  4  4  4  5  6  7  8  9]
a [ 6  6  5  5  5  4  5  4  5  6  7  8]
l [ 7  7  6  6  6  5  5  5  4  5  6  7]
i [ 8  8  7  6  7  6  6  6  5  4  5  6]
a [ 9  9  8  7  7  7  7  6  6  5  5  5]
```

Nice printing of the back-trace:

```
Backtrace: [(9, 11), (8, 10), (8, 9), (7, 8), (6, 7), (5, 6), (4,
    5), (4, 4), (3, 3), (2, 2), (1, 1), (0, 0)]

 | A | u | s | t |   | r | a | l | i |   | a |
 | O | s | i | t | a | r | a | l | i | y | a |
 | s | s | s |   | i |   |   |   |   | i |   |
```

[**1 credit**] Modify your code to also keep track of exactly which characters get *inserted* when translating from English to Kinyarwanda. What are the two most commonly inserted characters and how many times did this insertion occur in the dataset?

Note: this question is specifically about insertions, not substitutions. Although you could consider a substitution as a sequence of two operations (one deletion, one insertion), here we do not take this approach.

Answer: The most common are i: 192, y: 95

[**1 credit**] These two most commonly inserted characters often have very "short" pronunciation in Kinyarwanda. As a result, one could argue that they don't really impact the understanding of these terms, so inserting them shouldn't have such a high cost when computing the edit distance.

Modify your code so that when you compute the edit distance, the cost of inserting either of these two characters is 0.5 instead of 1. What is the average edit distance over the dataset with this modified scoring scheme?

Answer: Average edit distance with modified scoring scheme: 2.12987

[BONUS] Mongo Morphology (3 credits total)

Mongo (also known as Nkundo or Lomongo) is a Bantu language spoken by the Mongo Peoples of the central Democratic Republic of the Congo. Presently, there are around 400,000 native speakers spread out over a large area around the Congo River. Below is a table showing a few verb conjugations in Lomongo. d͡ʒ is a consonant pronounced like the **j** in the English word **jump**. ŋ is a consonant pronounced like the **ng** at the end of the English word **sing**.[5]

[5]The symbols d͡ʒ and ŋ are part of the International Phonetic Alphabet (IPA; `https://en.wikipedia.org/wiki/International_Phonetic_Alphabet`). This is the alphabet phoneticians

Imperative	2nd Singular	3rd Singular	3rd Plural	English
bota	oota	aota	baota	‘beget’
kamba	okamba	akamba	bakamba	‘work’
imed͡ʒa	wimed͡ʒa	imed͡ʒa	bimed͡ʒa	‘consent’
usa	wusa	usa	busa	‘throw’
bata	oata	aata	baata	‘get’
ɛna	wɛna	ɛna	bɛna	‘see’
isa	wisa	isa	bisa	‘hide’
d͡ʒila	od͡ʒila	ad͡ʒila	bad͡ʒila	‘wait’
ina	wina	ina	bina	‘hate’
bina	oina	aina	baina	‘dance’
asa	wasa	asa	basa	‘search’
saŋga	osaŋga	asaŋga	basaŋga	‘say’

[**2 credits**] Explain how these Mongo verb forms work by filling in the blanks below:

Each Mongo verb has a root form. The 4 verb forms shown here are formed by adding a prefix before the root form. The prefix for the imperative form is ∅−, the prefix for the 2nd singular form is **o-**, the prefix for the 3rd singular form is **a-**, and the prefix for the 3rd plural form is **ba-**.

However, we are not done yet: to get the final verb form, we must apply some sound change rules. The relevant rules are:

1. If there are two **vowels** in a row, delete **the first one**.
2. Delete **b** when it appears between two **vowels**.
3. Change **o** to **w** when it appears before a **vowel**.

There is one final wrinkle: the order that these rules are applied in maters. The rules must be applied in this order: First apply rule **3**, then rule **1**, then rule **2**.

[**1 credit**] Fill in the blanks in the table below.

Submission Instructions

This assignment must be done individually. Discussing the assignment on the class forum is absolutely ok – posting the solutions is not. If you use someone

use to describe the *sounds* of a language, and the symbols correspond to phonemes and phones. You can see the whole IPA and hear the correspondeing sounds for the symbols in this interactive chart: `https://www.ipachart.com/`.

Imperative	2nd Singular	3rd Singular	3rd Plural	English
bakisa	**oakisa**	**aakisa**	**baakisa**	‘add’
anda	wanda	**anda**	**banda**	‘begin’
solola	**osolola**	**asolola**	basolola	‘chat’
ponama	**oponama**	aponama	**baponama**	‘elect’
bowa	oowa	**aowa**	**baowa**	‘cure’
balusa	**oalusa**	aalusa	**baalusa**	‘turn’
loŋga	**oloŋga**	**aloŋga**	**baloŋga**	‘win’

else’s code (e.g. taken from a book or a website), make sure to properly cite it in your report.

Please submit all of the following in a gzipped tar archive (.tgz; not .zip or .rar) via Blackboard. If you’re making a full submission, please name your file `gmuid-hw1.tgz` (for example, the instructor’s submission would be `antonis-hw1.tgz`). If you’re making a partial submission, please name your file `gmuid-hw1-part.tgz` where part is the part (1, 2, 3, or 4) that you’re submitting. Note that submitting two files with the same name will overwrite one of them!

Your submission should contain:

- A PDF file (not .doc or .docx) with your responses to the instructions/questions above.
- All of the code that you wrote.
- A brief README file with instructions on how to build/run your code.