Language Models
Language Technology

Miquel Puig i Mena

September 2019

1 Objective

The report explains how to compute and analyze several metrics representing the acceptance of a general sentence given a language model backed by a big text corpora. Note that both the sentence to be computed and the dataset have to share language.

2 Motivation

Language text corrector, text suggestions or even identity validation processes must numerically qualify the probability of a sentence against a big text corpora. In other words, such innovative operations have to generalize a language by modelling it with a big dataset in order to provide analytic conclusions regarding inputted sentences. This report presents a way to create a model of a language plus a way to provide analytical results concerning a generic sentence to process.

3 Implementation

Implementation can be split in two independent section.

Firstly, document focuses in how to process a text dataset to acquire a language representation. Note that modelling a language brings infinity of possibilities but only unigram and bigram extraction will be performed in this report. **Future work might suggest to find best N for Ngram extraction in order to have better results when testing the system**.

Secondly, big importance must be given to what metrics can be used to define whether a sentence is suitable for an application. Since no application will be studied, an overview on each metric is provided with it's pros and cons.

3.1 Modelling a language

Before counting ngrams in the text used as full representation of the language, a sense of what we call sentences has to be defined in the system.

3.1.1 Sentence definition

Following expression has been used to spot sentences in a general text:

```
def normalize_corpus(...):
    ...
    # Text comma filtered:
    #   Remove all commas in it's usual use from text.
    text_comma_filtered = re.sub(r'\,(\s\p{L})',
                                 r'\1',
                                 text)
    # Sentenced text:
    #   Find all punctuation symbols followed by N spaces and a Capital letter.
    #   Replace by </s>\n<s>[FoundCapitalLetter]. Note that punctuation symbol is dropped.
    sentenced_text = re.sub(r'[\p{P}][\p{P}|\s]*(\p{Lu})',
                            r' {end} {start} \1'.format(end=end_tag, start=start_tag),
                            text_comma_filtered)
    # Add tags at beginning and end of file respectively
    sentenced_text_lower = start_tag + sentenced_text.lower() + end_tag
```

Once the original text has been sentenced marked with sentence separator tags, the system proceeds to extract all tokens. Note that it's important to denote that there's and end and start in each sentence in order to find combinations of word that might be more frequent than others at some positions of the sentence.

3.1.2 Token extraction

Cutting tokens from the text is done using tokenize function:

```
def tokenize(...):
    """
    Splits string text by non-letter symbols except for [<,>,/].
    Those symbols are reserved for sentence splitting meaning
  that they will from a token by them self.

    :param text: String. Text to split.
    :return: String[].
    """
    return re.split(r'[^\p{L}\<\>\/]+', text)
```

Finally, N value will define mode of counting tokens. An Ngram study makes X groups of N coliding elements where X is number of tokens in the processed corpus minus N plus 1. Such amount of groups comes from a sliding window of length N through all tokens.

3.2 Metric Analysis

After the system has been populated and processed with it's ngram dictionary containing frequencies and probabilities of {N-tuples, N-1-tuples, ..., tuples, singular words}, the model is ready to take sentences as input and provide a set of metrics to provide an analytical feedback.

3.2.1 Metric 0: Probability of a sentence

Even though it can be an abstract concept, a probability of a sentence gives an idea about how frequent a combination of words can be seen in the language modeled beforehand (See eq. 1). **It's important to note that if a Ntuple has never been seen in the model, an approximation is taken as seen in eq. 2**.

$$P(S) = \prod_{Ntuple_i = S_0}^{S_n} P(Ntuple_i) \text{ where } S = \{Ntuple_0, ..., Ntuple_n\} \tag{1}$$

$$P(Ntuple) \approx P(Mtuple) \text{ where } Ntuple = Nthword : Mtuple \tag{2}$$

3.2.2 Metric 1: Geometric mean probability

Indicates the central tendency or typical value of a set of numbers by using the product of their values[Wikb] (See eq. 3). Note that P is a serie of N products, therefore, G(P) defines an average among each word property comfronting a sentence.

$$G(P) = \sqrt[N]{P} \tag{3}$$

3.2.3 Metric 2: Entropy

The entropy rate can be defined as the time density of the average information in a stochastic process [Wika] (See eq. 4).

$$H(P) = -\frac{1}{N} \log P \tag{4}$$

3.2.4 Metric 3: Perplexity

Perplexity is a measurement of how well a probability model predicts a sample. A low perplexity indicates the probability model is good at predicting the sample [Wikc] (See eq. 5).

$$Perp(H) = 2^H \tag{5}$$

4 Conclusion

To finalize the report, the tool is tested against a custom sentence input cropped from Selma's text: "Han kunde inte förstå, att de blevo så glada". With similar results as previous tests: Bigram model manages to keep more information from both the inputted and stored information (See ouptut next page). In that sense, it is straight forward to say that Bigram models fit better to a language representation. Note that a challenging work could be to define N from Ngram to achieve optimal results in model's statistics.

```
----------------------------------------------------------------------------------------------------
Unigram Model
====================================================================================================
Word            Freq            Size            Prob
====================================================================================================
han             22743           1087122         0.0209203750820975
kunde           3397            1087122         0.0031247642858851167
inte            13826           1087122         0.012717983814144134
f rst             395             1087122         0.00036334468440524616
att             28914           1087122         0.026596830898464017
de              12599           1087122         0.011589315642586572
blevo           265             1087122         0.00024376288953769677
s                9558            1087122         0.008792021502646437
glada           177             1087122         0.00016281521301197106
</s>            62459           1087122         0.05745353327409435
====================================================================================================
Prob. Unigram: 1.8667215710968224e-24
Geometric mean prob.: 0.0042374828604601865
Entropy rate: 7.882576751735607
Perplexity: 235.98915510218757
----------------------------------------------------------------------------------------------------


----------------------------------------------------------------------------------------------------
Bigram Model
====================================================================================================
Word0           Word1           CountTuple      Count_1         P(wi+1|wi)
====================================================================================================
<s>             han             5481            62459           0.08775356633951872
han             kunde           411             22743           0.018071494525788153
kunde           inte            468             3397            0.13776861937003237
inte            f rst             54              13826           0.0039056849414147257
f rst             att             168             395             0.4253164556962025
att             de              1365            28914           0.04720896451545964
de              blevo           50              12599           0.003968568934042385
blevo           s                9               265             0.033962264150943396
s                glada           17              9558            0.0017786147729650554
glada           </s>            6               177             0.03389830508474576
====================================================================================================
Prob. Bigram: 1.3923020523018452e-16
Geometric mean prob.: 0.02596410437594034
Entropy rate: 5.267337728827146
Perplexity: 38.51471190843965
----------------------------------------------------------------------------------------------------
```

5 Reading

Concluding, Norvig's notebook provides a way to cut a long no-separated string in order to locate each word in the text. As this report, the experiment is executed both with unigrams and bigrams. After experimenting with a custom input, the notebook backs this report's idea where bigram processing accomplishes better results to model a language and, therefore, provides better results. Experiment shown below, shows how bigram model is able to find deeper words, specially when processing compund words.

```
string_to_compute = "mysleepingbagwentforahaircutinthedresssaloonmeanwhilein \
                        highschoolpeoplemidagedwerehavingkingsizebedsinevery \
                        roomnotethattheywereeatingpancakes"

seg1 = segment(string_to_compute)
seg2 = segment2(string_to_compute)
print(seg1 == seg2)
print(seg1)
print(seg2)

False
['my', 'sleeping', 'bag', 'went', 'for', 'a', 'haircut', 'in', 'the', 'dress',
'saloon', 'meanwhile', 'in', 'highschool', 'people', 'mid', 'aged', 'were',
'having', 'kingsize', 'beds', 'in', 'every', 'room', 'note', 'that', 'they',
'were', 'eating', 'pancakes']
['my', 'sleeping', 'bag', 'went', 'for', 'a', 'haircut', 'in', 'the', 'dress',
'saloon', 'meanwhile', 'in', 'high', 'school', 'people', 'mid', 'aged', 'were',
'having', 'king', 'size', 'beds', 'in', 'every', 'room', 'note', 'that', 'they',
'were', 'eating', 'pancakes']
```

References

[Wika] Wikipedia. *Entropy rate*. URL: https://en.wikipedia.org/wiki/Entropy_rate. (accessed: 29.09.2019).

[Wikb] Wikipedia. *Geometric Mean*. URL: https://en.wikipedia.org/wiki/Geometric%5C_mean. (accessed: 29.09.2019).

[Wikc] Wikipedia. *Perplexity*. URL: https://en.wikipedia.org/wiki/Perplexity. (accessed: 29.09.2019).