Probability and Statistics

MA - 202, Project 1

Statistical Language Modelling Using N-grams

MENTOR

Prof. Mayank Singh

I Project Brief

Our aim, using Natural Language Processing, is to predict the most probable next word and analyse the correctness of the input sentence. Further, we would also create a User interface to do the same to make it more interactive and user friendly.

II Main Steps

We divided the task into the following main steps and then integrated it to get the final result:

- Corpus Preparation
- Laplace Smoothing
- Next word prediction
- Perplexity of sentence
- Interface

III Corpus Preparation

Reliable Dataset:

- The model's accuracy depends on the reliability of the dataset used to train the model.
- For the given model, we have sought and utilized the best dataset available in our view

Data preprocessing:

- Firstly, we downloaded a suitable parser to convert the acquired dataset into machine language.
- Then, we installed suitable libraries to enhance preprocessing of data and analyse results.

III Corpus Preparation

Data visualisation and representation:

- We used several analysis methods to analyse the data statistics, which include word frequency analysis, sentence length analysis, and average word length analysis.

- We also plotted a histogram, which is represented as a word cloud for ease of representation, to study the nature of dataset. It turned out to be left-skewed

IV Laplace Smoothing

Defining the TASK

- Building the model that predicts the nth word is not enough as several folds of complexities are involved in it.

- Any dataset is always limited to an extent. So, within it, all possible combinations are not present. Hence, there is a need to build it.

IV Laplace Smoothing

We approached with **two ways** in mind:

1. **Linking two unlinked words**:
Here we thought of assigning zero probability to the words that are not linked.

2. **Completely ignoring unlinked words**:
Another option was to eradicate the possibility of the occurrence of the words together.

Thus, Laplace smoothing makes it better by assigning a frequency to these N-grams

IV Laplace Smoothing

Applying the process to the Bigram:

- Generalization where k is the parameter

$$P^{*}_{Add-k}(w_i|w_{i-1}) = \frac{c(w_i,w_{i-1})+k}{c(w_{i-1})+kV}$$

P* = Probability of the Laplace Smoothed N-grams

Wi = i'th word

c(wi,wi-1) = count of word sequence Wi-1,Wi

V = total number of words in vocabulary

- We found that this process helps rule out instances with zero probabilities and the probabilities change significantly.

Words	Probability without Laplace Smoothing	With LaplaceSmoothing
'not'	0.0707070707070707	0.0003389256058295204
'a'	0.06060606060606061	0.0002965599051008304
'currently'	0.050505050505050504	0.0002541942043721403
'awarded'	0.010101010101010102	8.47314014573801e-05
'stronger'	0.010101010101010102	8.47314014573801e-05

V Predicting Next Word Using N-grams and Markov Chain

Defining the TASK

The task is to code and implement the next word prediction. We are supposed to determine the next word based on the frequency of that word after the previous words. For the same, we will try and use the n-grams approach based on the concept of the Markov chain.

V Predicting Next Word Using N-grams and Markov Chain

Hypothesis 1

N-gram model:

- The task of predicting the next word using an n-gram model is equivalent to the probability of getting a specific word at the nth place given the previous n-1 words. The probability of the same is given by the Bayes formula given by, $P(w_n|w_1,.....,w_{n-1})$

- We will use previous words (the history) to predict the next word
  No. of parameters = (distinct words)^n* (distinct words - 1)

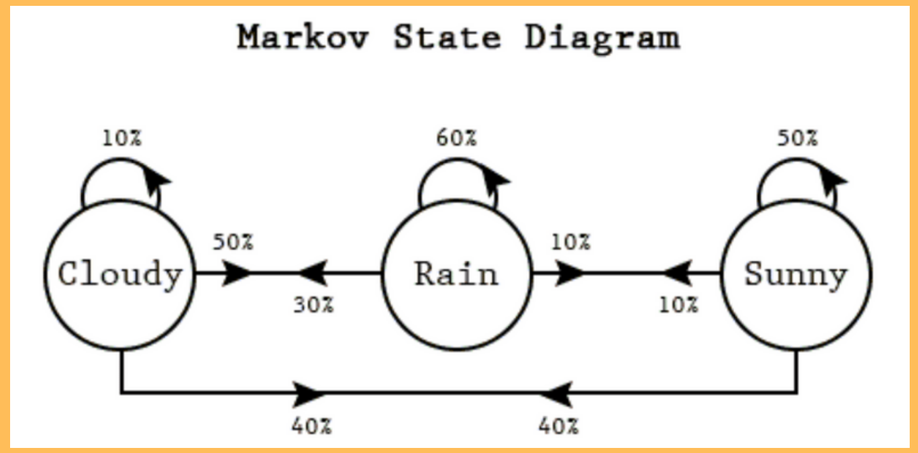
- We neither choose a high value for n nor make our decision based on only parameters. Considering this tradeoff, the most suitable model would be a **trigram model**.

V Predicting Next Word Using N-grams and Markov Chain

Hypothesis 2

N-gram model with Markov chain:

- To reduce the complexity of the n-gram models, we use the Markov property. To implement the Markov property in n-grams we use a Markov chain to store the probability of transitioning from one state to another.



V Predicting Next Word Using N-grams and Markov Chain

Calculating Probabilities:

- Since calculating the probability for the trigram model by hand can be complex, here we will use the bigram model to demonstrate the calculations. It can be easily extended to a trigram using a computer.

- Probability for bi-gram:

$$P(w_{1:n}) \approx \prod_{k=1}^{N} P(w_k|w_{k-1})$$

- Probability for tri-gram:

$$P(w_1^N) = \prod_{i=1}^{N} P(w_n|w_{n-2}, w_{n-1})$$

Defining the TASK

Perplexity is a statistic for evaluating language models. A good model should give valid English sentences a high score and invalid English sentences a low score. Perplexity is a commonly used metric for determining how "excellent" a model is.

$$\mathrm{PP(s)} = p(w1,\ldots\ldots,wn)^{-\frac{1}{n}}$$

VI Perplexity

Formulae:

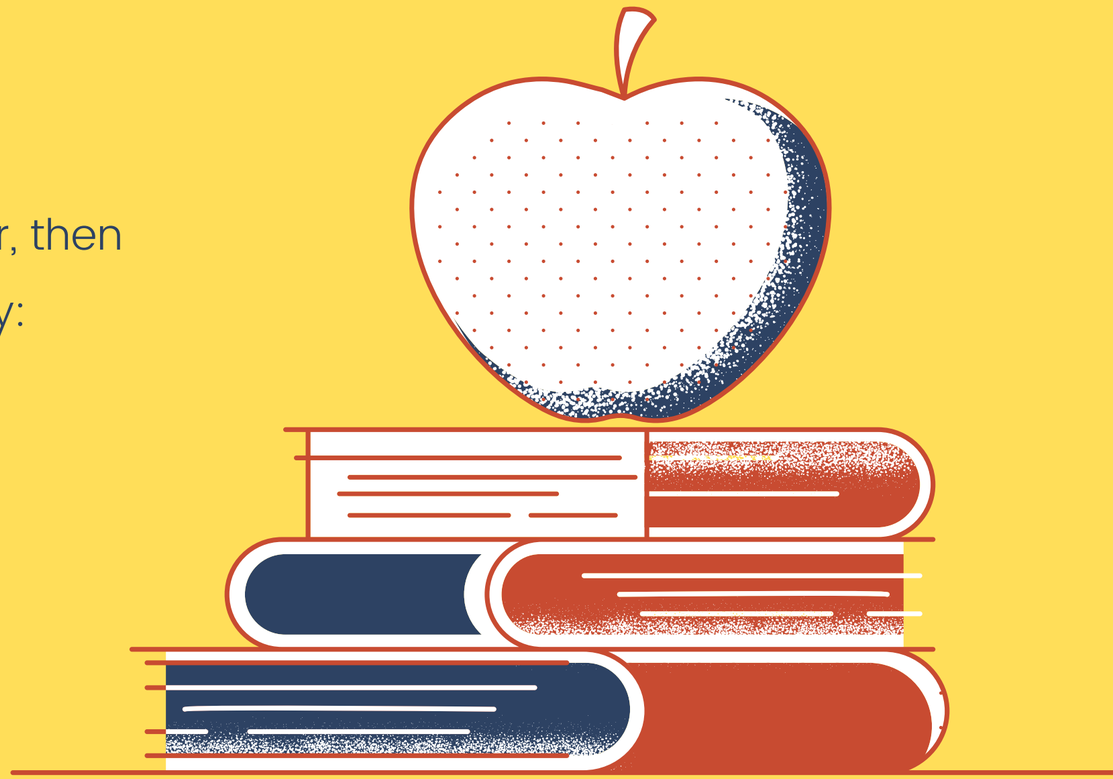
- The perplexity of the corpus, per word, is given by:

$$Perplexity(C) = \frac{1}{P(s_1, s_2, \ldots, s_N)^{\frac{1}{N}}} = \sqrt[N]{\frac{1}{P(s_1, s_2, \ldots, s_N)}}$$

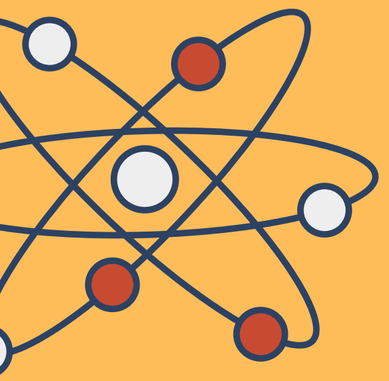
- If the probabilities are considered independent of one another, then the cumulative chance of these phrases being in C is given by:

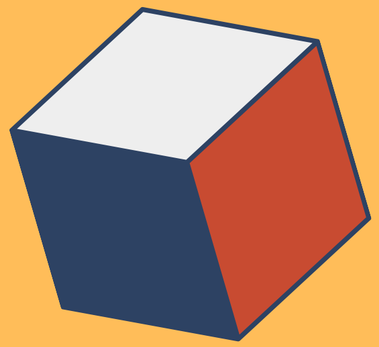
$$P(s_1, \ldots, s_m) = \prod_{i=1}^{m} p(s_i)$$

$$Perplexity(C) = \sqrt[N]{\frac{1}{\prod_{i=1}^{m} p(s_i)}}$$



VI Perplexity





Building the MODEL:

- the probability that our model gives to a whole sentence W made up of the words (w_1,w_2,...,w_N).
- An n-gram model uses the previous (n-1) words to predict the next.

P(w1, w2, ... , wN) = P(w1)*P(w2|w1)*P(w3|w1,w2)*...*P(wk|w1,...,wk-1)

- Normalizing: divide the sum of some terms by the number of words to generate a per-word measure

$$P(W)^{\frac{1}{N}} = \left(\prod_{i=1}^{N} P(w_i)\right)^{\frac{1}{N}}$$

- The perplexity of a single sentence, in which case W would be a single sentence

$$PP(W) = \frac{1}{P(w_1, w_2, \ldots, w_N)^{\frac{1}{N}}} = \sqrt[N]{\frac{1}{P(w_1, w_2, \ldots, w_N)}}$$

Code

Now, let's run through the code to get an insightful view of the process we have discussed so far.