Saarland University

Department of Computational Linguistics

Course: **Computational Linguistics**

Ngram Word Guesser

Author:
Maximilian Wolf
Matriculation: 2539863

Lecturer:
Prof. Dr. Alexander Koller
Tutor:
Antoine Venant

31.03.2017

Abstract

Word prediction is a popular task for Natural Language Processing (NLP). It has recently grown in popularity with the rise of mobile devices that lack a full-size keyboard and thus make text entry cumbersome. One way to solve this issue is to use automated text prediction.

In my project I want to explore the possibilities of n-gram-based word prediction in the context of a game, where the player enters an incomplete sentence and the computer tries to complete it in the way the player intended. For this purpose I employ bigrams, trigrams and fourgrams. They are generated from the Brown corpus and the Webtext corpus of nltk and from the sentences of previous game sessions. The corpora are explored in regular as well as in reversed order, so that the context before as well as after an omitted word can be captured with appropriate n-grams. Function words and non-alphanumeric characters are filtered out before generating an answer.

I show that the proposed method yields many acceptable answers for sentence completion, but exactly guessing the omitted word a player thought of is rarely achieved.

Contents

1 Introduction

The central idea of this project was to create a game where the player omits one word of a sentence that the computer tries to guess with the help of n-grams. The basic interaction is demonstrated in example (1).

(1) **Player:** It's raining _ and dogs.

Computer: It's raining *cats* and dogs.

After the computer presents a solution, the player gives feedback about its correctness. If the guessed answer is wrong, the player is asked to enter the correct solution. Afterwards, the computer tries to find out if the guess was close by looking at the 10 most probable answers that were not chosen (if available). Such an interaction can be seen in (2). The correct sentence with the player's solution is saved in the file *learned_sentences.txt* and part of the game's corpus when it is started the next time.

(2) **Player:** Sharp as a _

Computer: Sharp as a *word*
Was my guess correct?
Please enter "y" or "n":

Player: n

Computer: Oh no! Can you tell me the word you were looking for?

Player: knife

Computer: Thank you. I will try to remember it the next time we play!
Here are other answers I considered:
['labor', 'thousand', 'knife', 'symbolic', 'proper', 'consequence', 'partnership', 'Pandora', 'counterman', 'cap']
I was pretty close!

When the player is done playing, the game can be quit by entering *q* instead of a sentence. When the game is over, some statistics will be displayed as shown in (3).

(3) **Computer:** Please enter a sentence with one word replaced by "_" (or "q" to quit):

Player: q

Computer: In this session I was correct 3 times and wrong 10 times.
1 time I was pretty close.
Goodbye, I hope you had some fun!

2 Resources

To generate n-grams, I employ the Brown corpus and the Webtext corpus from the package nltk.corpus[1]. The Brown corpus consists of roughly one million words and contains text from 500 sources. It was created in 1961 at Brown University. I chose the Brown corpus because it is fairly large and features various domains.

The Webtext corpus contains over 10,000 posts of instant messaging chat sessions. It was created the Naval Postgraduate School and contains texts from 2006. I chose the Webtext corpus to also include a more modern corpus and an additional domain that may be similar to the sentences a player might enter in the player-computer-interaction within the game. An additional bonus of using this corpus is that it can lead to rather funny answers, some of which are presented in section 5.

In addition to the former mentioned corpora, user entered sentences from previous game sessions are also used to generate n-grams. The sentences are saved in the file *learned_sentences.txt* in the *resources* directory. This file grows with each game and therefore enables the computer to learn and improve at word guessing.

The final resource I employ is a list of English function words[2] containing 277 entries. This list is used to filter out function words from potential answers. I do this as I found that players rarely omit function words for the computer to guess. Because function words are very frequent, the decision to filter them out and only consider them when no other options are available, should improve the accuracy of the answers.

3 Implementation

The implementation is done in Python 3. The corpora are obtained using the nltk.corpus library. The Brown, Webtext and the learned sentences corpora are merged into one, forming the complete corpus. This corpus is then used to generate bigrams, trigrams and fourgrams. I do not consider n-gram sequences above fourgrams as they are not very likely to be beneficial due to their low occurrence frequency. Two sets of n-grams are generated for each stage: n-grams for the corpus in regular as well as in reversed order. This is done to enable the generation of answers for omitted words while considering a context before and after the word. Example: *Allow me to _ the door for you.*

The n-grams are produced by calling BasicNgram(), found in *ngram.py* in the *src* directory. The script was provided by Antoine Venant for assignment 1 in the Computational Linguistics course. It defines and trains an n-gram model over the given corpus.

The n-grams are generated each time the game is started. They are not saved to a file. The benefits of this are firstly, the file size of the game is kept to a minimum. This was important to me as the project should be submitted via e-mail. Secondly, corpora

[1] `http://www.nltk.org/api/nltk.corpus.html`

[2] `https://semanticsimilarity.wordpress.com/function-word-lists/`

can be exchanged easily. The drawback is that it may take up to a minute to start the game (this is of course also dependent on the hardware the game is run on). To alleviate this issue, the user is gradually updated on the current progress.

After the generation is done, the player is prompted to enter a sentence with one word left out, denoted by an underscore. The input string is tokenized and verified (the sentence needs to contain an underscore and at least one word). After that the index of the omitted word is ascertained. The words before and after represent the context for the n-grams. A prediction using fourgrams is always preferred over trigram predictions, which are preferred over bigram predictions. The 50 most probable predictions for the missing word that were made with the context to the left are compared with the predictions generated with the context to the right of the word. If the intersection of theses two lists is not empty and contains content words, the first entry of the intersection is chosen as the guessed answer. The order of the intersection list is determined by the order of the prediction list using the left side context, which in turn is ordered by most probable to least probable answer (based on n-gram frequency distributions). Function words and non-alphanumeric characters are only chosen as an answer if no other option exists.

After the answer is presented, the user can confirm whether it was correct, or give the right solution. The counts of correct, wrong and close answers (if the answer was within the 10 most probable answers) are counted and displayed when the player quits the game.

3.1 Requirements

Requirements to run the game:

- Python 3
- The Brown corpus
- The Webtext corpus

If a corpus is missing, the player will be notified and given the option to run nltk.download() in order to install the corpora.

3.2 How to Play

To start the game run *word_guesser.py* in the *src* directory.
After the game is started, follow the instructions on screen.

The game is available for download here:
`https://github.com/artificial-max/ngram_word_guesser`

Player	Correct	Wrong	Close	Total
1	5	17	0	22
2	4	19	0	23
3	3	7	0	10
4	3	13	1	17
5	1	28	2	31
6	1	5	0	6
7	1	6	0	7
8	0	5	2	7
9	0	12	1	13
10	0	6	0	6
Total:	18	100	5	123

Table 1: Player results.

4 Evaluation

To evaluate the word guessing approach, I asked 10 people to play the game. The results are shown in Table 1. Around 15% of the guessed answers were exactly what the player thought of, 81% of the answers were wrong and in 4% of the cases the right answer was at least in the top 10 almost guessed answers.

Often times there are countless possible answers and guessing the one the player thought of is highly unlikely. For example, the input sentence *I am very _* is a low constraint sentence that has a huge amount of possible answers. A player might have thought of the word *tired*, while the game might answer *proud* instead.

Some mistakes may be avoided by also taking part of speech into account. This could help to avoid mistakes like *I am planning to *sleeping**.

Another area for improvement is the selection of corpora. Bigger and more modern corpora including recent pop culture references could improve the accuracy. A corpus consisting of idioms would also be highly likely to be beneficial, as many players entered partial idioms as their sentences.

While the results show that guessing the exact answer that a player has in mind is difficult, I would still consider many of the wrong answers given by the game as acceptable answers for sentence completion. Some interesting and funny examples are presented in the next section.

5 Highlights

Below is a selection of answers generated by game. The guessed answer is marked by two stars, the expected answer is featured in brackets.

- Bacon and *science* (eggs)

  Here are other answers I considered: ['stared', 'gulped', 'grade', 'altogether', 'cousins', 'sodium', 'thrashed', 'Chalidale', 'Eve', 'binoculars']

- Walk a mile in their *entirety* (shoes)

  Here are other answers I considered: ['personal', 'saddles', 'ideas', 'persistence', 'scientific', 'development', 'religious', 'studies', 'previous', 'footsteps']

- Beat around the *rear* (bush)

  Here are other answers I considered: ['wheel', 'ring', 'Atlantic', 'photograph', 'heart', 'Merchandise', 'ranch', 'camps', 'schoolhouse', 'corner']

- Struck by a smooth *fit* (criminal)

  Here are other answers I considered: ['humility', 'finish', 'file', 'lawn']

- Kill two birds with one *stone* (stone)

- Kill two *proposed* with one stone (birds)

  Here are other answers I considered: ['thousand', 'drivers', 'excellent', 'sheets', 'Mexicans', 'small', 'misconceptions', 'trees', 'professional', 'crackers']

- I've lost my *virginity* shoes. (new)

  Here are other answers I considered: ['job', 'fake', 'wallet', 'trust', 'GameBoy', 'favorite', 'cell']

- I am playing this *saxophone* instead of working. (game)

- sugar and spice and everything *tastes* . (nice)

  Here are other answers I considered: ['blows']

- This *terminology* tastes very good (wine)

  Here are other answers I considered: ['agency', 'wine', 'meant', 'chick', 'magnificent', 'expectation', 'harbor', 'da y', 'catalogue', 'throwed']
  I was pretty close!

- this corpus gives weird *results* . (results)