Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
6.863J/9.611J, Natural Language Processing
Fill In The Gap!
Mid-Course Check-In

Authors: *Mehmet Tugrul Savran, Scott Viteri, William Mitchell*
KerberosID: tugrul, sviteri, williamm

---

1 Problem Statement: Fill In The Gap

Filling in a gap within a sentence is a complex problem, and a prevalent domain of research in Natural Language Processing. It is complex because we can often easily fill in a blank easily, but cannot exactly describe how we do it. An example of a Fill In The Gaps problem is realized below:

The sentence:
The President — the Vice President.
Choices:
a) Cat
b) Glanced
c) Met
d) Ate

Here, the language model should test each of 4 possible choices and pick the most appropriate option C.

Ostensibly, the problem is to pick the most appropriate word from a corpus that will:

1) yield a grammatical sentence -The President cat the Vice President. is not a grammatical sentence - and

2) a natural sentence -The President ate the Vice President. is an unlikely sentence.

Accordingly, our computational pipeline breaks down into two major components: 1) the corpus that will yield candidate words to fill in and 2) the language model that picks the best fit. So far, we used the metric of entropy to determine the best fit. We, hence describe the language models we used and how entropy related to them in the next section.

2 Language Models and Methods

2.1 Corpus Selection and Sentence Generation

We used the corpus from feat0.fcfg[1] from the NLTK book, for both the FCFG and the Penn Tree Bank grammars. We also used the Penn Tree Bank Corpus for initial testing of our method on the PTB grammar. The key point with the corpus selection up to this point was to be just large enough so that we get results that are both interesting and non-trivial.

For each corpus, we generated sentences and removed one word at a time. We picked the best replacement word from the corpus using a few different methods, which we will explain below.

[1] $https://github.com/nltk/nltk_teach/blob/master/examples/grammars/book_grammars/feat0.fcfg$

2.2 Language Models

The brain of the pipeline is the language model that fills in the gap in the sentence. We utilized 3 language models: the skip-gram, the Penn Tree Bank (PTB) CFG, and an NLTK FCFG to find the minimum entropy parse. We tested each of these against the same corpus and discuss our preliminary results below.

2.2.1 Skipgram

Skipgrams are pretty much N-Grams, but expanding on both sides. Hence instead of only considering the previously seen words (the Markov assumption in N-Grams), skip-grams consider the following words that occur after the blank word.

We did not consider N-Grams since they were not suitable for the nature of our problem, since we are removing words within any position of the sentence, an N-Gram model would almost always miss out on information.

We hypothesized that Skipgram would seldom fail because they may fail to capture the compositional structure of the sentence.

For example, in the sentence:

He ate the lunch, – words

So a skip-gram model could very well accept a sentence such as "He ate the lunch, *cat* words." In the time we had before this check in, we prioritized testing our Penn Tree Bank CFG and the NLTK FCFG methods before skip-grams, because they were more likely to yield interesting results.

2.2.2 Penn Tree Bank CFG

Another language model we took on to test was Penn Tree Bank powered by a Context Free Grammar, which is nothing but Penn Tree Bank's own set of rules. Our methodology was as follows:

1. Define corpus as Penn Tree Banks corpus (accessed by treebank.words())
2. Generate valid sentences from Penn Tree Bank
3. Take out (construct the blank) a word randomly from each of these sentences
4. For each of these sentences, replace the blank with each of the words from the corpus
5. Compute the entropy for this candidate sentence
6. Return the minimum entropy candidate sentence

We executed the entire methodology above by a Python script using NLTK tools, and also our lovely parser from 6.863 Software package - we called the parser inside the script (at step 5).

An illustration of this methodology is shown below:
An original sentence generated from PTB:
Pierre Vinken , 61 years old , will join the board as a non-executive director Nov. 29 .
Blanked sentence:
Pierre Vinken , 61 years old , will — the board as a nonexecutive director Nov. 29 .

Minimum entropy sentence (result):
Pierre Vinken , 61 years old , will form the board as a non-executive director Nov. 29 .

We had initially hypothesized that this method was bound to fail since it will seldom honor feature agreement. We found out that such hypothesis held. Some of our results are shown below:

1)

Original sentence: Pierre Vinken , 61 years old , will join the board as a non-executive director Nov. 29 .
Blanked sentence: Pierre Vinken , 61 years old , will — the board as a non-executive director Nov. 29 .
Minimum entropy sentence: Pierre Vinken , 61 years old , will form the board as a non-executive director Nov. 29
Minimum entropy: 11.3109
Runner-up sentence: Pierre Vinken , 61 years old , will join the board as a non-executive director Nov. 29
Runner-up entropy: 11.3111 (very close!)

2)

Original sentence: Mr. Vinken is chairman of Elsevier N.V. , the Dutch publishing group .
Blanked sentence: Mr. Vinken is — of Elsevier N.V. , the Dutch publishing group .
Minimum entropy sentence: Mr. Vinken is , of Elsevier N.V. , the Dutch publishing group .
Minimum entropy: 10.1262
Runner up sentence: Mr. Vinken is Pierre of Elsevier N.V. , the Dutch publishing group .
Runner up entropy: 12.011

3)

Original sentence: There is no asbestos in our products now .
Blanked sentence: There is no — in our products now .
Minimum entropy sentence: There is no , in our products now .
Minimum entropy: 10.376
Runner up sentence: There is no Pierre in our products now .
Runner up entropy: 11.5218

4)

Original sentence: Compound yields assume reinvestment of dividends and that the current yield continues for a year .
Blanked sentence: Compound yields — reinvestment of dividends and that the current yield continues for a year .
Minimum entropy sentence: Compound yields and reinvestment of dividends and that the current yield continues for a year .
Minimum entropy: 11.478
Runner up sentence: Compound yields , reinvestment of dividends and that the current yield continues for a year .
Runner up entropy: 11.4977

As seen from the results above, the PTB Language Model found a natural fit only for the first sentence. The word form in the sentence Pierre Vinken , 61 years old , will form the board as a non-executive director Nov. 29 is indeed an acceptable fit. It is also comforting that the model didnt just replace back the original word join (it was found in the runner-up sentence!).

The second sentence Mr. Vinken is , of Elsevier N.V. , the Dutch publishing group . is a syntactically acceptable one, albeit semantically unnatural. While the language model could have just put back the word chairman it simply put a comma. This is OK in the sense of less information gain, since putting a comma will emit less information than putting the word chairman however a person cant be a publishing group!

For the third sentence, we see that the minimum-entropy sentence is There is no , in our products now. This sentence parses through the Penn Tree Bank grammar but is ungrammatical.

The fourth sentence Compound yields and reinvestment of dividends and that the current yield continues for a year parses through the Penn Tree Bank grammar, and like the previous sentence is also ungrammatical.

Consequently, we saw that Penn Tree Bank CFG language model showed instances where it failed semantically and syntactically, which is much more important since it is more easily enforced. Penn Tree Banks failure revealed a crucial need for a more solid language model, which we discuss below.

2.2.3 Penn Tree Bank FCFG

The clear option after using CFGs is to add features into the mix! We added tense and plurality features to power the Penn Tree Bank with an FCFG. For preliminary testing, we took a sample feature-CFG from the NLTK book, and picked filler words again using the entropy metric. The sample grammar was as follows:

```
% start S
# ###################
# Grammar Productions
# ###################
# S expansion productions
S -> NP[NUM=?n] VP[NUM=?n]
# NP expansion productions
NP[NUM=?n] -> N[NUM=?n]
NP[NUM=?n] -> PropN[NUM=?n]
NP[NUM=?n] -> Det[NUM=?n] N[NUM=?n]
NP[NUM=pl] -> N[NUM=pl]
# VP expansion productions
VP[TENSE=?t, NUM=?n] -> IV[TENSE=?t, NUM=?n]
VP[TENSE=?t, NUM=?n] -> TV[TENSE=?t, NUM=?n] NP
# ###################
# Lexical Productions
# ###################
Det[NUM=sg] -> 'this' | 'every'
Det[NUM=pl] -> 'these' | 'all'
Det -> 'the' | 'some' | 'several'
PropN[NUM=sg]-> 'Kim' | 'Jody'
N[NUM=sg] -> 'dog' | 'girl' | 'car' | 'child'
N[NUM=pl] -> 'dogs' | 'girls' | 'cars' | 'children'
IV[TENSE=pres,  NUM=sg] -> 'disappears' | 'walks'
TV[TENSE=pres, NUM=sg] -> 'sees' | 'likes'
IV[TENSE=pres,  NUM=pl] -> 'disappear' | 'walk'
TV[TENSE=pres, NUM=pl] -> 'see' | 'like'
IV[TENSE=past] -> 'disappeared' | 'walked'
TV[TENSE=past] -> 'saw' | 'liked'
```

First, we took this grammar and expanded it into CFG form with Regex matches and a for loop. We picked the sentence *this dog walked*, and removed *walked*. We substituted in every possible terminal from the above fcfg, and checked the parses entropy using the provided parse file from lab 2. The sentence with the lowest entropy was *this dog disappears*, which is semantically funky, but syntactically fine. We tried the same sentence using the PTB grammar, and got *this dog like*. Both of these behaviors were as expected.

```
PTB CFG
    6.863$ python3 max_ent_treebank.py
Found an improved sentence:  this dog this
Current entropy is:  15.817
Found an improved sentence:  this dog all
Current entropy is:  15.1721
Found an improved sentence:  this dog the
Current entropy is:  14.6765
Found an improved sentence:  this dog disappears
Current entropy is:  14.0972
Found an improved sentence:  this dog walk
Current entropy is:  14.0485
Found an improved sentence:  this dog see
Current entropy is:  13.8712
Found an improved sentence:  this dog like
Current entropy is:  13.8155
Minimum entropy found is:  13.8155
Minimum entropy sentence is:  this dog like

NLTK FCFG
6.863$ python3 max_ent_treebank.py
Found an improved sentence:  this dog disappears
Current entropy is:  2.86165
Minimum entropy found is:  2.86165
Minimum entropy sentence is:  this dog disappears
```

We could continue making these proofs of inadequacy more rigorous, but maybe more promising direction is to augment the feature grammar with semantic information. Logical forms can be embedded into the NLP feature grammars, and this seems like a good way to encode a notion that *he ate a hamburger* is better than *he pick-pocketed a hamburger*. We intend to try a few different methods of adding semantics into the picture. We hope that through working on lab 3, we will start gaining a better grasp of how to use semantics to help with this problem.

3 Goals/Extensions

We have proposed exploring four different methods for the fill-in-the-blank-problem: by using skip-gram, CFGs, and FCFGs augmented with semantic information. We have already tried a few preliminary methods to add semantics. We created a distance metric between sentences using wordnet, but this metric was useless because it only encoded pairwise relationships between individual words in each sentence – so for our purposes would fill in *tigers — zebras* with *tigers leopards zebras* rather than *tigers eat zebras*, for example. We even tried using combinatory categorial grammars to parse possible filled-in sentences, which we then mapped to semantic higher-order logic representations of those sentences. We picked the sentence fills whose parses generated the shortest lambda description – a sort of minimum description length method. Preliminary results did not look too promising, but it is possible that we have not explored this path with enough

rigor, and we plan testing this idea further.

A fortunate part of our task is that validation is fairly simple – one can always take sentences, remove a word, and check if the method guessed correctly. But this may be overly strict. We could potentially take into account a few metrics – did the method choose the correct word? Did the method choose the correct word in its top five choices? Did the method choose a fill-in word that was in the wordnet synonym set of the correct word? We can also do testing on the small set of words that can be generated by the NLTK book FCFG from above. I think this has just enough structure to be both useful and easily interpretable. If we get to the point where we have a few separate feasible algorithms to fill in the blank, we can test on a corpus with greater structure as needed.

4 References

[1] S. Bird, E. Loper, and E. Klein. NLTK 3.2.5 Documentation. https://www.nltk.org/

[2] G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. Introduction to WordNet: An On-Line Lexical Database. 1993. http://wordnetcode.princeton.edu/5papers.pdf

[3] ccg2lambda: Composing Semantic Representations Guided by CCG Derivations. https://github.com/mynlp/ccg2lambda

[4] The Penn Treebank: An Overview
https://pdfs.semanticscholar.org/182c/4a4074e8577c7ba5cbbc52249e41270c8d64.pdf

[5] Feature Grammar Parsing. http://www.nltk.org/howto/featgram.html

[6] S. Bird, E. Loper, and E. Klein. Building Feature Based Grammars. 2014. http://www.nltk.org/book/ch09.html

[7] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean. Distributed Representations of Words and Phrases and their Compositionality. 2013. https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

[8] M. Marcus, B. Santorini, M. Marcinkiewicz, and A. Taylor. Penn Treebank Sample. Philadelphia: Linguistic Data Consortium. 1999. https://catalog.ldc.upenn.edu/ldc99t42

[9] S. Bird, E. Loper, and E. Klein. Source Code for nltk.grammar. 2017. http://www.nltk.org/_modules/nltk/grammar.html