Tutorial 2: Language models and Tagging

Status: Completed
Your identity is visible during marking.
Marks: 6 / 10

Submission deadline: 19 Jun 2019 09:00

Responsible for the exercises: Bill Noble (bill.noble@gu.se)

Note: Some exercises ask you to compute values. In all cases, you should be able to make the computation with paper and a calculator, but you are also welcome to use your computer. Either way, please show your work by including a couple intermediary steps in your calculation, or by including the relevant snippet of code you used.

Part I: Language modelling

Q1. Entropy

(See: [Goldsmith], p.17)

Take the following two sentences as a mini-corpus.

> <start> how much wood would a woodchuck chuck if a woodchuck could chuck wood a woodchuck would chuck as much wood as a woodchuck could chuck if a woodchuck could chuck wood <end>

Q1a - Compute the Shannon entropy for the distribution of unigrams and bigrams in this text. Which distribution has more entropy and why?

Q1b - How would the entropy of these distributions change if you add +1 Laplacian smoothing?

Q2. Perplexity

(See: [JMv2] p.95 or [JMv3] p. 59)

> <start> would a woodchuck chuck wood if it could chuck wood <end>

> <start> wood a woodchuck chuck would if it could chuck would <end>

Q2a - Compute the (length-normalized) perplexity of the two sentences above using unigram and bigram language models "trained" on the mini corpus from Q1 (so, four scores total). *Note: you will have to use smoothing to account unforseen ngrams*.

Q2b - What do these perplexity scores tell you about the unigram vs. bigram language models you computed?

Q3. Bigram PFST

(See: Lecture 4 slides)

Q3 - Diagram a probabilistic finite state transducer representing the bigram language model from Q2.

Part II: Part of Speech Tagging

(See: [JMv2] chapter 5)

Q4: Types of taggers

Q4a - One of the simplest kinds of taggers is a lookup tagger that simply maps each word to a single part of speech. It is easy to implement, but what are the disadvantages of such a tagger? Mention 2.

Q4b - Suppose we want to build a better tagger for English which uses word affixes to determine the PoS of a word (e.g. plural nouns generally end in *s*). Pick 3 tags from the Penn tagset and write a regular expression for each. Can you think of any words which will be incorrectly tagged (false positives or false negatives) by your tagger?

Q4c - Neither of the taggers above take context into account. Assume that you then build a bigram HMM tagger from the following two-sentence corpus:

- the/DET cat/N saw/V a/DET fish/N
- a/DET saw/N cannot/MD fish/V

Your tagger is now given the phrase "a fish". How will it tag the word *fish* in this phrase, and why? Show your calculations.

Q5: Playing with the Core NLP tagger

The Stanford CoreNLP tagger uses the Penn Treebank tagset. You can play around with it here.

Q5a - Find a sentence where this (roughly state-of-the-art) tagger tags at least one word incorrectly. What's your explanation for why it got it wrong? For hints, take a look at this paper.

Q5b - Think of a downstream application that might be impacted by the mistake you found. What could go wrong?

Note: In question 2, a previous version gave the hint that you can use Laplacian smoothing to account for unseen bigrams. Of course, this is true for unigrams, too!

Hemanth Kumar Battula , 4 Dec 2018 23:32

File name: Tutorial2.py (5,8 KB)

Status set to: To be marked

Comment: I have been on a vacation for the last week. Honestly, its all my fault as I could not submit the assignment. I would like to revise it and submit in 2 days. Hope you would consider. Thank you.

Hemanth Kumar Battula , 5 Dec 2018 17:19

File name: Tutorial2_solutions.docx Listen (546,0 KB)

Status set to: To be marked

Hemanth Kumar Battula , 5 Dec 2018 17:19

File name: Tutorial2.py (7,7 KB)

Status set to: To be marked

Hemanth Kumar Battula , 5 Dec 2018 17:21

Status set to: To be marked

Comment: The first 'Tutorial2.py' file submitted on 04/12/2018 is not the actual file. Files submitted today i.e on 05/12/2018 are the actual solutions. Thank you.

Bill Noble , 20 Dec 2018 01:17

Status set to: Completed

Mark set to: 6

Grade set to: G

Comment: "Q1a - Unigrams, good! For bigrams, it looks like you might have computed entropy for the conditional distrubition. That's ok though -- for the joint distribution bigrams have more entropy because it is flatter and ""wider"" (has more types) than the unigram distribution.
Q1b - Looks good. You should get higher entropy for both when you add +1 smoothing. This is because distributions are flatter when you add +1 smoothing and flatter distributions have more uncertainty." Q2 - The bigram model should get a lower perplexity than the unigram model. In the bigram model, the first sentence should have somewhat lower perplexity than the first since the model can "tell" that wood and would are switched in the second sentence. Q3 - This is ok -- but I meant for you to represent the model itself. So you would have one node for each word in the vocabulary. The transitions between nodes represent P(w1|w2) in the bigram model (the probability of moving to word2 after seeing word1) Q4 - Good! Q5 - Good!