CS4248 Natural Language Processing Tutorial 2: Language Models and Text Classification

In this tutorial, we'll practice four technical topics from our Week 03 and 04 lectures: Language Models, Perplexity, tf×idf and Naïve Bayes.

Please come to your tutorial session, with your attempts to these questions. It's fine to come without a working solution but to get the most out of tutorial, you should attempt them so that you have practice before solutions can be shared. Extension exercises marked with a "**" or "***" are slightly more advanced and you (and your tutorial leader) may not have time to go over them. You're welcomed to discuss these on the forum among yourselves (Let's keep an active forum!)

It is encouraged to try to do the question before the tutorial since we plan to have a Kahoot game at the start of the tutorial.

1. **What is going [MASK]? (Language Models and Smoothing)**

   1. Let's imagine your brain as a super language model (with unlimited capacity for vocabulary and accompanying probabilities). Come up with three predictions for "*What is going* [MASK]" and rank by their approximate likelihood.
   2. Given the corpus below, calculate the bigram MLE $P(w|go)$ for all possible words w. For this exercise only, do some pre-processing and lemmatize the words before constructing the tokens and probabilities (we do this here to densify the probabilities a bit).

      ** *To think about:* In practice, LMs usually do very little or no preprocessing outside of tokenization. Why do you think that is?

      Alice is going to the school. The traffic is going smoothly. She is going to have a meeting with her advisor, Prof Smith. Recently her research is going on very well. When Prof Smith asks "how's it going with your research?", Alice's heart beat does not go up and she goes on with a clear update. After the meeting, they go for a coffee downstairs together.
   3. Let's assume $|V| = 100$. Calculate $P(quickly|go)$ directly using your language model obtained in last step? If this isn't possible, state why and how you could resolve it, and your equivalent probability.

      *** To think about:* What are limitations of a bigram (or even any Markov assumption)? Let's consider some new contexts:

      Hi Alice, what is going [MASK]?
      The microwave is not heating up, what is going [MASK]?
      The tax is going [MASK] *by ten percent.*

      Can we get satisfactory predictions with a bigram LM? How would you solve your identified limitations?

      Hint: Play around the interactive LM demo here: https://huggingface.co/FacebookAI/roberta-base. Note that this is more general language model and not one based strictly on bigrams. It uses a transformer architecture that is at the limit of our syllabus; we'll touch upon it starting in Week 08.

Explanation:MASK might have the following predictions: (1) *on*: 50%, (2) *to*: 25%, (3) *with*: 5%. There could be many other words that could occur, so the percentages here don't add up to 100%.

1. We first count the bigram of “go X”, we have
   - go to: 2
   - go smooth: 1
   - go on: 2
   - go with: 1
   - go up: 1
   - go for: 1

   hence we have:
   - P(to|go) = 0.25
   - P(smooth|go) = 0.125
   - P(on|go) = 0.25
   - P(with|go) = 0.125
   - P(up|go) = 0.125
   - P(for|go) = 0.125
2. No. We need to do smoothing since “quickly” is OOV. We apply simple add-1 (Laplace) smoothing in the below. Specifically, $P_{laplace}(quickly|go) = \frac{C(go, quickly) + 1}{C(go) + V} = \frac{1}{108}$

2. Don’t be perplexed

		w_i				
		<s>	I	am	here	</s>
w_{i-1}	<s>	0	0.1	0.0002	0.01	0
	I	0	0.0003	0.5	0.0001	0.0002
	am	0	0.005	0.0001	0.03	0.0005
	here	0	0.02	0.0001	0.003	0.001
	</s>	0	0	0	0	0

1. Let’s start simply. **Definition:** What is Perplexity? Why do we need the fraction $\frac{1}{N}$ in the definition of perplexity?
2. **Intuition:** Why do we need this measure of perplexity?
3. **vs Probability:** Given a test set W, how does the probability $P(W)$ change when we minimize perplexity?
4. **Practice:** Assume we are using bigram model. Given $W =$ “I am here” and the following table, compute $PP(w)$.

   ** *Word scramble*: What happens with the input $W =$ ”I here am”? Would we obtain a smaller or larger perplexity? Why?

Explanation:

1. Perplexity is the inverse probability of the test set W, normalized by the number of words. Denoted as $PP(W)$.

   For a test set $W = w_1 w_2 \dots w_N$:

   $$PP(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}} \tag{11}$$

   If we use chain rule and bigram model:

   $$PP(W) = (\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})})^{\frac{1}{N}} \tag{12}$$
2. To measure the quality of language model.
3. Minimizing the perplexity is the same as maximizing the probability
4. $PP(W) = ((P(< s >) \times P(I| < s >) \times P(am|I) \times P(here|am) \times P(< /s > |here))^{-\frac{1}{5}}$

   ≈ 14.6 If $W =$ ”I here am”, perplexity would be larger, ≈ 288.5.

3. **Which is better? (a.k.a. BOW vs tf×idf and Cosine Similarity)**

Consider following query(q) and documents (*d1* - *d3*):

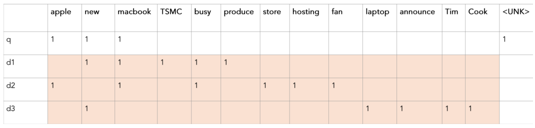
q: *Apple ships new Macbook.*
d1: *TSMC is busy producing new Macbook.*
d2: *Apple Stores are busy hosting Macbook fans.*
d3: *New laptop are announced by Tim Cook.*

1. Show a Bag of Words (BoW) representation and tf×idf representation of all the documents. Preprocess first by removing stop words from present on the NLTK list: (https://gist.github.com/sebleier/554280).

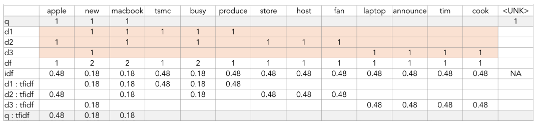
2. Please calculate the Cosine similarity between <*q, d1*>, <*q, d2*>, <*q, d3*> using both BoW and tf×idf representation.

3. Which document shares the highest semantic similarity with the query? Has it been ranked at the highest place? If not, any recommendation to address this problem?
Hint: "Semantic similarity" also measures how similar a set of documents are. But it is based on the likeness of their meaning or semantic content[6] as opposed to lexicographical similarity (which tf×idf is designed for).
*** *From a different perspective (angle)*: Besides Cosine Distance, what other distance metrics are in your radar? Do you think there will be a big impact to the ranking by changing the distance metric?

Explanation: 1. Please see the figures. Click this link to Google Slides for a bigger figure.



	apple	new	macbook	TSMC	busy	produce	store	hosting	fan	laptop	announce	Tim	Cook	<UNK>
q	1	1	1											1
d1		1	1	1	1	1								
d2	1		1		1		1	1	1					
d3		1								1	1	1	1	



	apple	new	macbook	tsmc	busy	produce	store	host	fan	laptop	announce	tim	cook	<UNK>
q	1	1	1											1
d1		1	1	1	1	1								
d2	1		1		1		1	1	1					
d3		1								1	1	1	1	
df	1	2	2	1	2	1	1	1	1	1	1	1	1	
idf	0.48	0.18	0.18	0.48	0.18	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.48	NA
d1 : tfidf		0.18	0.18	0.48	0.18	0.48								
d2 : tfidf	0.48		0.18		0.18		0.48	0.48	0.48					
d3 : tfidf		0.18								0.48	0.48	0.48	0.48	
q : tfidf	0.48	0.18	0.18											

Figure 1: BoW and tfidf

2. (1) For BoW: <*q, d1*>: 0.45; <*q, d2*>: 0.41; <*q, d3*>: 0.22.
(2) For tf×idf: <*q, d1*>: 0.16; <*q, d2*>: 0.49; <*q, d3*>: 0.06.
3. The last one has the highest semantic similarity. Two approaches:
(1) **Distributional word representation:** We hope following word pairs will be aligned between q and d3: Macbook <-> laptop; ship <-> announce; Apple <-> Tim Cook. The idea is that even though the words are not the same, but their distributional representations are similar.
(2) **Knowledge sources:** The motivation is the same. But knowledge sources are more explicit than distributional word representation. Related entities are explicitly linked in the knowledge source.
Perhaps a (recursive) * question:** Again, which is better, distributional word representation OR knowledge source? Can they be even unified in a single view :D ?

4. **Stay Simple, Stay Naïve, (a.k.a. Naïve Bayes)**

[6] https://en.wikipedia.org/wiki/Semantic_similarity

doc	exciting	happy	upset	furious	class
d_1	0	1	2	1	*neg*
d_2	2	1	2	3	*neg*
d_3	0	0	1	1	*neg*
d_4	3	3	0	0	*pos*
d_5	2	0	1	0	*pos*

Table 11: Counts of key sentiment words for each document. The assigned classes are given in the last column (*pos* for positive and *neg* for negative).

doc	text	class
d_6	- I am happy, happy with the exciting result, but furious at your attitude.	*neg*
d_7	- I don't think they'll be happy with the seemly happy news, which makes me upset rather than happy.	*neg*
d_8	- The exciting news is like an exciting gift, dispelling my upset mood.	*pos*

Table 12: Text of testing documents, where the ground-truth classes are displayed in the last column.

Naïve Bayes allows us to define the features we want in text classification. In this question, you will utilize Naïve Bayes (and its variants) to do sentiment analysis. To simplify, we fix the set of key sentiment words as `{exciting, happy, upset, furious}`. And the word counts of each document are provided in Table 11 with the sentiment class as the training corpus. In the following tasks, you need to train Naïve Bayes (and its variants) on it to predict the class of the testing samples in Table 12.

1. Train a multinomial Naïve Bayes model with add-1 smoothing. Use this model to assign a class (*pos* or *neg*) to each of the document in testing set.
2. In practice, whether a word occurs or not usually matters more than its frequency, which motivates a lot of works to clip the word counts (*i.e.*, remove all duplicate words) in each document. And this variant is called **binary multinomial Naïve Bayes**.

   Construct a binary multinomial Naïve Bayes model trained on the same corpus, and use it to predict the labels of the testing set.
3. Use the metrics we've learned from class to evaluate the results. Can you see some difference in the predictions of the two models? Which metric do you think better reflects the models' performance?
4. Negation is an important issue in sentiment analysis. For example, in d_6, it changes the sentiment polarities of the key words.

   Can you think of an idea to address the influence of negation here?

Hints. You can utilize the logpriors and loglikelihoods below for your calculation.

Recall the formulas to calculate the log-prior for a class c (15) and the (add-one) log-likelihood for a word w w.r.t a class c (16):

$$logprior[c] \leftarrow \log \frac{N_c}{N_{doc}} \tag{13}$$

$$loglikelihood[w,c] \leftarrow \log \frac{count(w,c)+1}{\sum_{w' \in V}(count(w',c)+1)} \tag{14}$$

Then for multinomial NB, we have the results as follows:

$$logprior[pos] = \log \frac{2}{5}, \quad logprior[neg] = \log \frac{3}{5}$$

$$loglikelihood[exciting, pos] = \log \frac{5+1}{9+4} = \log \frac{6}{13}, \quad loglikelihood[exciting, neg] = \log \frac{2+1}{14+4} = \log \frac{1}{6}$$

$$loglikelihood[happy, pos] = \log \frac{3+1}{9+4} = \log \frac{4}{13}, \quad loglikelihood[happy, neg] = \log \frac{2+1}{14+4} = \log \frac{1}{6}$$

$$loglikelihood[upset, pos] = \log \frac{1+1}{9+4} = \log \frac{2}{13}, \quad loglikelihood[upset, neg] = \log \frac{5+1}{14+4} = \log \frac{1}{3}$$

$$loglikelihood[furious, pos] = \log \frac{0+1}{9+4} = \log \frac{1}{13}, \quad loglikelihood[furious, neg] = \log \frac{5+1}{14+4} = \log \frac{1}{3}$$

For binary NB, we have the results as follows:

$$logprior[pos] = \log \frac{2}{5}, \quad logprior[neg] = \log \frac{3}{5}$$

$$loglikelihood[exciting, pos] = \log \frac{2+1}{4+4} = \log \frac{3}{8}, \quad loglikelihood[exciting, neg] = \log \frac{1+1}{9+4} = \log \frac{2}{13}$$

$$loglikelihood[happy, pos] = \log \frac{1+1}{4+4} = \log \frac{1}{4}, \quad loglikelihood[happy, neg] = \log \frac{2+1}{9+4} = \log \frac{3}{13}$$

$$loglikelihood[upset, pos] = \log \frac{1+1}{4+4} = \log \frac{1}{4}, \quad loglikelihood[upset, neg] = \log \frac{3+1}{9+4} = \log \frac{4}{13}$$

$$loglikelihood[furious, pos] = \log \frac{0+1}{4+4} = \log \frac{1}{8}, \quad loglikelihood[furious, neg] = \log \frac{3+1}{9+4} = \log \frac{4}{13}$$

Explanation: Below are the answers to Question 4.

1. Recall the formulas to calculate the log-prior for a class c (15) and the (add-one) log-likelihood for a word w w.r.t a class c (16):

$$logprior[c] \leftarrow \log \frac{N_c}{N_{doc}} \tag{15}$$

$$loglikelihood[w, c] \leftarrow \log \frac{count(w, c) + 1}{\sum_{w' \in V}(count(w', c) + 1)} \tag{16}$$

Then we have the results as follows:

$$logprior[pos] = \log \frac{2}{5}, \quad logprior[neg] = \log \frac{3}{5}$$

$$loglikelihood[exciting, pos] = \log \frac{5+1}{9+4} = \log \frac{6}{13}, \quad loglikelihood[exciting, neg] = \log \frac{2+1}{14+4} = \log \frac{1}{6}$$

$$loglikelihood[happy, pos] = \log \frac{3+1}{9+4} = \log \frac{4}{13}, \quad loglikelihood[happy, neg] = \log \frac{2+1}{14+4} = \log \frac{1}{6}$$

$$loglikelihood[upset, pos] = \log \frac{1+1}{9+4} = \log \frac{2}{13}, \quad loglikelihood[upset, neg] = \log \frac{5+1}{14+4} = \log \frac{1}{3}$$

$$loglikelihood[furious, pos] = \log \frac{0+1}{9+4} = \log \frac{1}{13}, \quad loglikelihood[furious, neg] = \log \frac{5+1}{14+4} = \log \frac{1}{3}$$

So for the testing cases, we can calculate the log-probability of each class as follows:

- d_6.

$$sum[pos] = \log \frac{2}{5} + \log \frac{6}{13} + 2 \times \log \frac{4}{13} + \log \frac{1}{13}$$

$$sum[neg] = \log \frac{3}{5} + \log \frac{1}{6} + 2 \times \log \frac{1}{6} + \log \frac{1}{3} < sum[pos]$$

$d_6 \rightarrow$ *positive*

- d_7.

$$sum[pos] = \log\frac{2}{5} + 3 \times \log\frac{4}{13} + \log\frac{2}{13}$$

$$sum[neg] = \log\frac{3}{5} + 3 \times \log\frac{1}{6} + \log\frac{1}{3} < sum[pos]$$

  $d_7 \rightarrow$ *positive*

- d_8.

$$sum[pos] = \log\frac{2}{5} + 2 \times \log\frac{6}{13} + \log\frac{2}{13}$$

$$sum[neg] = \log\frac{3}{5} + 2 \times \log\frac{1}{6} + \log\frac{1}{3} < sum[pos]$$

  $d_8 \rightarrow$ *positive*

2. Here in binary NB, we have uniform unit counts for every word. In this case, we can clip the word counts to 1 for calculation. Then we have the results as follows:

$$logprior[pos] = \log\frac{2}{5}, \quad logprior[neg] = \log\frac{3}{5}$$

$$loglikelihood[exciting, pos] = \log\frac{2+1}{4+4} = \log\frac{3}{8}, \quad loglikelihood[exciting, neg] = \log\frac{1+1}{9+4} = \log\frac{2}{13}$$

$$loglikelihood[happy, pos] = \log\frac{1+1}{4+4} = \log\frac{1}{4}, \quad loglikelihood[happy, neg] = \log\frac{2+1}{9+4} = \log\frac{3}{13}$$

$$loglikelihood[upset, pos] = \log\frac{1+1}{4+4} = \log\frac{1}{4}, \quad loglikelihood[upset, neg] = \log\frac{3+1}{9+4} = \log\frac{4}{13}$$

$$loglikelihood[furious, pos] = \log\frac{0+1}{4+4} = \log\frac{1}{8}, \quad loglikelihood[furious, neg] = \log\frac{3+1}{9+4} = \log\frac{4}{13}$$

So for the testing cases, we can calculate the log-probability of each class as follows:

- d_6.

$$sum[pos] = \log\frac{2}{5} + \log\frac{3}{8} + \log\frac{1}{4} + \log\frac{1}{8}$$

$$sum[neg] = \log\frac{3}{5} + \log\frac{2}{13} + \log\frac{3}{13} + \log\frac{4}{13} > sum[pos]$$

  $d_6 \rightarrow$ *negative*

- d_7.

$$sum[pos] = \log\frac{2}{5} + \log\frac{1}{4} + \log\frac{1}{4}$$

$$sum[neg] = \log\frac{3}{5} + \log\frac{4}{13} + \log\frac{3}{13} > sum[pos]$$

  $d_7 \rightarrow$ *negative*

- d_8.

$$sum[pos] = \log\frac{2}{5} + \log\frac{3}{8} + \log\frac{1}{4}$$

$$sum[neg] = \log\frac{3}{5} + \log\frac{2}{13} + \log\frac{4}{13} < sum[pos]$$

  $d_8 \rightarrow$ *positive*

3. We calculate four kinds of metrics. F1 is better as it reflects the model performance without significant bias on classes.

   - **accuracy**. multinomial NB → 33.33%; binary NB → 100.0%
   - **recall**. multinomial NB → 100.0%; binary NB → 100.0%
   - **precision**. multinomial NB → 33.33%; binary NB → 100.0%
   - **F1 score**. multinomial NB → 50.00%; binary NB → 100.0%

   Binary NB performs better as it has uniform unit counts for every word, which highlights the occurrence of the key words rather than the counts of them. Therefore, the effect of `happy` is weaken in d_6 and d_7 while the effects of the negative key words are strengthened.

4. Add a prefix `NOT_` in front of the words which follow a negation in the sentence: `not, n't, rather than`. For example, in d_8, we replace the `happy` with `NOT_happy`, do the document will be like:

   `I don't think they'll be NOT_happy with the seemly NOT_happy news, which makes me upset rather than NOT_happy.`

   Then there aren't key words of `happy` in this sentence, which can help both NB models to make correct predictions.