601.465/665 — Natural Language Processing
Assignment 2: Probability and Vector Exercises

Prof. Kevin Duh and Jason Eisner — Fall 2019
Due date: Wednesday 18 September, 11am

Only a bit of programming is required for this assignment. Mostly, you'll solve some pencil-and-paper problems.

Collaboration: ***You may discuss each problem with one other student in the class.*** However, please write up your answers separately and hand them in separately. Otherwise it's too easy (for this homework) to get by without understanding. If you discuss a problem with someone else, disclose this in your `README` and mention your partner's name.

Piazza: For questions about the assignment, put the Question Number at the front of the subject header so that it is easier for people to find related posts. E.g., "3a) Why does ...?"

How to hand in your work: Put all notes, documentation, and answers to questions in a `README.pdf` file, and submit it via Gradescope. You will submit your code separately.

Notation: When you are writing `README.pdf` file, you will need some way of typing mathematical symbols. A good option is to use LaTeX. Alternatively, you could try the equation editor in Google Docs or another word processor, or you could embed scans or photos of your neatly handwritten equations.

If you must type plain text, please pick one of the following three notations and use it consistently throughout your assignment. (If you need some additional notation not described here, just describe it clearly and use it.) Use parentheses as needed to disambiguate division and other operators.

	Text	Picts	LaTeX
$p(x \mid y)$	`p(x | y)`	`p(x | y)`	`p(x \mid y)`
$\neg x$	`NOT x`	`~x`	`\neg x`
$\bar{x}$ (set complement)	`COMPL(x)`	`\x`	`\bar{x}`
$x \subseteq y$	`x SUBSET y`	`x {= y`	`x \subseteq y`
$x \supseteq y$	`x SUPERSET y`	`x }= y`	`x \supseteq y`
$x \cup y$	`x UNION y`	`x U y`	`x \cup y`
$x \cap y$	`x INTERSECT y`	`x ^ y`	`x \cap y`
$x \geq y$	`x GREATEREQ y`	`x >= y`	`x \geq y`
$x \leq y$	`x LESSEQ y`	`x <= y`	`x \leq y`
$\emptyset$ (empty set)	`NULL`	`0`	`\emptyset`
$\mathcal{E}$ (event space)	`E`	`E`	`E`

1. These short problems will help you get the hang of manipulating probabilities. Let $\mathcal{E} \neq \emptyset$ denote the event space (it's just a set, also known as the outcome space or sample space),

and p be a function that assigns a real number in $[0, 1]$ to any subset of $\mathcal{E}$. This number is called the probability of the subset.

You are told that p satisfies the following two axioms: $p(\mathcal{E}) = 1$. $p(X \cup Y) = p(X) + p(Y)$ provided that $X \cap Y = \emptyset$.[1]

As a matter of notation, remember that the **conditional probability** $p(X \mid Z) \stackrel{\text{def}}{=} \frac{p(X \cap Z)}{p(Z)}$. For example, singing in the rain is one of my favorite rainy-day activities: so my ratio $p(\text{singing} \mid \text{rainy}) = \frac{p(\text{singing AND rainy})}{p(\text{rainy})}$ is high. Here the predicate "singing" picks out the set of singing events in $\mathcal{E}$, "rainy" picks out the set of rainy events, and the conjoined predicate "singing AND rainy" picks out the intersection of these two sets—that is, all events that are both singing AND rainy.

(a) Prove from the axioms that if $Y \subseteq Z$, then $p(Y) \leq p(Z)$.
You may use any and all set manipulations you like. Remember that $p(A) = 0$ does not imply that $A = \emptyset$ (why not?), and similarly, that $p(B) = p(C)$ does not imply that $B = C$ (even if $B \subseteq C$).

(b) Use the above fact to prove that conditional probabilities $p(X \mid Z)$, just like ordinary probabilities, always fall in the range $[0, 1]$.

(c) Prove from the axioms that $p(\emptyset) = 0$.

(d) Let $\bar{X}$ denote $\mathcal{E} - X$. Prove from the axioms that $p(X) = 1 - p(\bar{X})$. For example, $p(\text{singing}) = 1 - p(\text{NOT singing})$.

(e) Prove from the axioms that $p(\text{singing AND rainy} \mid \text{rainy}) = p(\text{singing} \mid \text{rainy})$.

(f) Prove from the axioms that $p(X \mid Y) = 1 - p(\bar{X} \mid Y)$. For example, $p(\text{singing} \mid \text{rainy}) = 1 - p(\text{NOT singing} \mid \text{rainy})$. This is a generalization of 1d.

(g) Simplify: $\big(p(X \mid Y) \cdot p(Y) + p(X \mid \bar{Y}) \cdot p(\bar{Y})\big) \cdot p(\bar{Z} \mid X)/p(\bar{Z})$

(h) Under what conditions is it true that $p(\text{singing OR rainy}) = p(\text{singing}) + p(\text{rainy})$?

(i) Under what conditions is it true that $p(\text{singing AND rainy}) = p(\text{singing}) \cdot p(\text{rainy})$?

(j) Suppose you know that $p(X \mid Y) = 0$. Prove that $p(X \mid Y, Z) = 0$.[2]

(k) Suppose you know that $p(W \mid Y) = 1$. Prove that $p(W \mid Y, Z) = 1$.

2. All cars are either red or blue. The witness claimed the car that hit the pedestrian was blue. Witnesses are believed to be about 80% reliable in reporting car color (regardless of the actual car color). But only 10% of all cars are blue.

(a) Write an equation relating the following quantities and perhaps other quantities:

$$p(\mathit{Actual} = \text{blue})$$
$$p(\mathit{Actual} = \text{blue} \mid \mathit{Claimed} = \text{blue})$$
$$p(\mathit{Claimed} = \text{blue} \mid \mathit{Actual} = \text{blue})$$

[1] In fact, probability functions p are also required to satisfy a generalization of this second axiom: if $X_1, X_2, X_3, \ldots$ is an infinite sequence of disjoint sets, then $p(\bigcup_{i=1}^{\infty} X_i) = \sum_{i=1}^{\infty} p(X_i)$. But you don't need this for this assignment.

[2] More precisely, $p(X \mid Y, Z)$ could be either 0 or undefined, namely $0/0$. (There do exist advanced ways to redefine conditional probability to avoid this $0/0$ problem. Even then, though, one may want a probability measure p to leave some probabilities or conditional probabilities undefined. This turns out to be important for reasons beyond the scope of this course: e.g. http://en.wikipedia.org/wiki/Vitali_set.)

Reminder: Here, *Claimed* and *Actual* are *random variables*, which means that they are functions over some outcome space. For example, the probability that $Claimed = \text{blue}$ really means the probability of getting an outcome x such that $Claimed(x) = \text{blue}$. We are implicitly assuming that the space of outcomes x is something like the set of witnessed car accidents.

(b) Match the three probabilities above with the following terms: *prior probability*, *likelihood of the evidence*, *posterior probability*.

(c) Give the values of all three probabilities. (Hint: Use Bayes' Theorem.) Which probability should the judge care about?

(d) Let's suppose the numbers 80% and 10% are specific to Baltimore. So in the previous problem, you were implicitly using the following more general version of Bayes' Theorem:

$$p(A \mid B, Y) = \frac{p(B \mid A, Y) \cdot p(A \mid Y)}{p(B \mid Y)}$$

where Y is $city = \text{Baltimore}$. Just as 1f generalized 1d, by adding a "background" condition Y, this version generalizes Bayes' Theorem. Carefully prove it.

(e) Now prove the more detailed version

$$p(A \mid B, Y) = \frac{p(B \mid A, Y) \cdot p(A \mid Y)}{p(B \mid A, Y) \cdot p(A \mid Y) + p(B \mid \bar{A}, Y) \cdot p(\bar{A} \mid Y)}$$

which gives a practical way of finding the denominator in the question 2d.

(f) Write out the equation given in question 2e with A, B, and Y replaced by specific propositions from the red-and-blue car problem. For example, Y is "$city = \text{Baltimore}$" (or just "Baltimore" for short). Now replace the probabilities with actual numbers from the problem, such as 0.8.

Yeah, it's a mickeymouse problem, but I promise that writing out a real case of this important formula won't kill you, and may even be good for you (like, on an exam).

3 3. Beavers can make three cries, which they use to communicate. `bwa` and `bwee` usually mean something like "come" and "go" respectively, and are used during dam maintenance. `kiki` means "watch out!" The following **conditional probability table** shows the probability of the various cries in different situations.

$p(cry \mid situation)$	Predator!	Timber!	I need help!
`bwa`	0	0.1	0.8
`bwee`	0	0.6	0.1
`kiki`	1.0	0.3	0.1

(a) Notice that each column of the above table sums to 1. Write an equation stating this, in the form $\sum_{variable} p(\cdots) = 1$.

(b) A certain colony of beavers has already cut down all the trees around their dam. As there are no more to chew, $p(timber) = 0$. Getting rid of the trees has also reduced $p(predator)$ to 0.2. These facts are shown in the following **joint probability table**. Fill in the rest of the table, using the previous table and the laws of probability. (Note that the meaning of each table is given in its top left cell.)

$p(cry, situation)$	Predator!	Timber!	I need help!	TOTAL
`bwa`				
`bwee`				
`kiki`				
TOTAL	0.2	0		

(c) A beaver in this colony cries `kiki`. Given this cry, other beavers try to figure out the probability that there is a predator.

i. This probability is written as: $p(\_\_\_\_\_\_\_\_)$

ii. It can be rewritten without the | symbol as: ________

iii. Using the above tables, its value is: ________

iv. Alternatively, Bayes' Theorem allows you to express this probability as:

$$\frac{p(\square) \cdot p(\square)}{p(\square) \cdot p(\square) + p(\square) \cdot p(\square) + p(\square) \cdot p(\square)}$$

v. Using the above tables, the value of this is:

$$\frac{\square \cdot \square}{\square \cdot \square + \square \cdot \square + \square \cdot \square}$$

This should give the same result as in part iii., and it should be clear that they are really the same computation—by constructing table (b) and doing part iii., you were *implicitly* using Bayes' Theorem.

4 4. A **language model** is a probability function p that assigns probabilities to word sequences such as $\vec{w} = (\texttt{arya}, \texttt{loves}, \texttt{soy}, \texttt{lint})$.

Suppose $\vec{w} = w_1 w_2 \cdots w_n$ (a sequence of n words). A **trigram language model** defines

$$p(\vec{w}) \stackrel{\text{def}}{=} \prod_{i=1}^{n+1} p(w_i \mid w_{i-2}, w_{i-1}) \tag{1}$$

where by convention we define $w_{n+1} = \text{EOS}$ ("end of sequence") and $w_0 = w_{-1} = \text{BOS}$ ("beginning of sequence").

For example, the model says that a speaker who says "`arya loves soy lint`" has generated the sequence in left-to-right order, by rolling 5 dice that happened to land on `arya`, `loves`, `soy`, `lint`, EOS. Each roll uses a special die selected by the two previously rolled characters (or BOS).

(a) Explicitly write out $p(w_1 w_2 w_3 w_4)$ when you use naive estimates of the parameters, such as

$$p(w_4 \mid w_2, w_3) \stackrel{\text{def}}{=} \frac{c(w_2 w_3 w_4)}{c(w_2 w_3)} \tag{2}$$

where $c(w_2 w_3 w_4)$ denotes the *count* of times the trigram $w_2 w_3 w_4$ was observed in a training corpus. The following terms will appear in your formula: $c(\textsc{bos bos})$, $c(\textsc{bos bos}\ \texttt{arya})$, $c(\texttt{soy lint}\ \textsc{eos})$. Explain what each of these terms are.

Remark: Naive parameter estimates of this sort are called **maximum-likelihood estimates** (MLE). They have the advantage that they maximize the probability (equivalently, minimize the perplexity) of the training data. But they will generally perform badly on test data, unless the training data were so abundant as to include all possible trigrams many times. This is why we must smooth these estimates in practice.

(b) In the data you will use in the *next* assignment, a sequence may be very long. While the sequence is still preceded by BOS and terminated with EOS, each linguistic sentence within the sequence is delimited by <s> at the start and </s> at the end. For example, the following single sequence consists of 3 linguistic sentences:

<s> do you think so </s> <s> yes </s> <s> at least i thought so </s>

Given English training data, the probability of

<s> do you think the </s>

should be extremely low under any good language model. Why? In the case of the trigram model, which parameter or parameters are responsible for making this probability low?

(c) You turn on the radio as it is broadcasting an interview. Assuming a trigram model, match up expressions (A), (B), (C) with descriptions (1), (2), (3):

The expression

(A) $p(\texttt{do}) \cdot p(\texttt{you} \mid \texttt{do}) \cdot p(\texttt{think} \mid \texttt{do}, \texttt{you})$

(B) $p(\texttt{do} \mid \texttt{<s>}) \cdot p(\texttt{you} \mid \texttt{<s>}, \texttt{do}) \cdot p(\texttt{think} \mid \texttt{do}, \texttt{you}) \cdot p(\texttt{</s>} \mid \texttt{you}, \texttt{think})$

(C) $p(\texttt{do} \mid \texttt{<s>}) \cdot p(\texttt{you} \mid \texttt{<s>}, \texttt{do}) \cdot p(\texttt{think} \mid \texttt{do}, \texttt{you})$

represents the probability that

(1) the first complete sentence you hear is do you think

(2) the first 3 words you hear are do you think

(3) the first complete sentence you hear starts with do you think

Explain your answers briefly. *Remark:* The distinctions matter because "do" is more probable at the start of an English sentence than in the middle, and because (3) describes a larger event set than (1) does.

Note: One could also define a kind of reversed trigram language model p_{reversed} that instead assumed the words were generated in reverse order ("from right to left"):

$$p_{\text{reversed}}(\vec{w}) \stackrel{\text{def}}{=} \prod_{i=0}^{n} p(w_i \mid w_{i+1}, w_{i+2}) \tag{3}$$

where by convention we define $w_0 = \textsc{bos}$ and $w_{n+1} = w_{n+2} = \textsc{eos}$. In this case, the generating process stops when we hit BOS, instead of EOS. This idea is used to great success is modern NLP language modeling methods (bidirectional RNNs).

5. The next assignment will involve various kinds of smoothed language models. Often, smoothing involves treating similar events in similar contexts as having similar probabilities. (Backoff is one example.)

   One strategy will be to treat similar *words* as having similar probabilities. But what does "similar words" mean? More concretely, how will we *measure* whether two words are similar?

   Log into the ugrad machines, and look in the directory `/usr/local/data/cs465/hw-lm/lexicons/`. Remember that a *lexicon* lists useful properties of the words of a language. Each of the `words-*.txt` files[3] is a lexicon that lists over 70,000 words of English, with a representation of each word as a d-dimensional vector. In other words, it *embeds* these words into the vector space $\mathbb{R}^d$.

   We can now define "similar words" as words with similar vectors. A word may have many syntactic and semantic properties—some words are transitive verbs, some words are plural, some words refer to animals. These properties can be considered by a log-linear model, and words with similar vectors tend to have similar properties. The larger d is, the more room the vector has to encode interesting properties.

   The vectors in these particular files were produced automatically by Google's `word2vec` program.[4] Their individual dimensions are hard to interpret as natural properties. It would certainly be nice if dimension 2 (for example) represented the "animal" property, so that a word that referred to an animal would be one whose vector $\vec{v} = (v_1, v_2, \ldots v_d)$ had strongly positive v_2. In practice, however, `word2vec` chooses an arbitrary basis for the vector space. So the "animal" direction—to the extent that there is one—is actually represented by some arbitrary vector $\vec{u}$. The animal words tend to be those whose vectors $\vec{v}$ have strongly positive $\vec{v} \cdot \vec{u}$.

   (Geometrically, this dot product measures how far $\vec{v}$ extends in direction $\vec{u}$ (it projects $\vec{v}$ onto the $\vec{u}$ vector). Algebraically, it computes a certain linear combination of $v_1, v_2, \ldots, v_n$. As a special case, if $\vec{u}$ were $(0, 1, 0, 0, \ldots)$, then $\vec{v} \cdot \vec{u}$ would in fact return v_2. But there's no reason to expect that $\vec{u}$ would be that simple.)

   You'll be using these lexicons in the next assignment. For this assignment, you'll just warm up by writing a short program to get a feel for vector embeddings.

   (a) Your program `findsim` should print the 10 words most similar to `seattle`, other than `seattle` itself, according to the 50-dimensional embeddings, if you run it as

   ```
   ./findsim words-50.txt seattle
   ```

   They should be printed in decreasing order of similarity.

[3] The file format should be easy to figure out. The file `words-`d`.txt` can be regarded as a matrix with about 70,000 rows and d columns. Each row is labeled by a word. The first line of the file is special: it gives the number of rows and columns.

[4] The details are not important for this assignment, but the vectors are optimized by gradient descent so as to arrange that the vector for each word token w_i will be predictable (to the extent possible) from the average vector of nearby word tokens (w_j for $j \neq i$ and $i - 5 \leq j \leq i + 5$). If you're curious, you can find details in Mikolov et al. (2013)'s paper "Distributed Representations of Words and Phrases and their Compositionality," available at http://arxiv.org/abs/1301.3781. We ran the CBOW method over the first 1 billion characters of English Wikipedia. `word2vec` doesn't produce vector representations for rare words ($c(w) < 5$), so we first replaced rare words with the special symbol OOL ("out of lexicon"), forcing `word2vec` to learn a vector representation for OOL.

To measure the similarity of two words with vectors $\vec{v}$ and $\vec{w}$, please use the *cosine similarity*, which is the cosine of the angle θ between the two vectors:

$$\cos\theta = \left(\frac{\vec{v}}{||\vec{v}||}\right) \cdot \left(\frac{\vec{w}}{||\vec{w}||}\right) = \frac{\vec{v}\cdot\vec{w}}{||\vec{v}||\;||\vec{w}||} = \frac{\sum_{i=1}^{d} v_i w_i}{\sqrt{\sum_{i=1}^{d} v_i^2}\sqrt{\sum_{i=1}^{d} w_i^2}} \in [-1, 1]$$

5 What are the most similar words to `seattle`, `dog`, `communist`, `jpg`, `the`, and `google`?

6 Play around some more. What are some examples that work "well" or "poorly"? What patterns do you notice?

7 So far you have been using $d = 50$. What happens for larger or smaller values of d?

Hint: It's probably a good idea to read the lexicon into some data structures.

Hint: Start by making `findsim` find only the single most similar word, which should be easy enough. Then you can generalize this method to find the 10 most similar words. (Since you only want the top 10, it's not necessary to sort the whole lexicon by similarity—that method is acceptable, but inefficient.)

Note: You can copy the lexicon files to your personal machine. But to avoid downloading such large files, it may be easier to run `findsim` directly on the ugrad machines. If you do this, please don't waste space by making fresh copies of the files on the ugrad machines. Just use them at their current location. If you like, create symbolic links (shortcuts) to them by typing "`ln -s /usr/local/data/cs465/hw-lm/lexicons/* .`" in your own working directory.

(b) Now extend your program so that it can also be run as follows:

```
./findsim words-50.txt king man woman
```

This should find the 10 words most similar to the vector `king` − `man` + `woman`, other than those three words themselves.

(The old command `./findsim words-50.txt seattle` should still work. Note that it is equivalent to `./findsim words-50.txt seattle seattle seattle`, and you may want to implement it that way.)

You can regard the above command as completing the analogy

```
man : woman :: king : ?
```

Try some more analogies, this time using the **200-dimensional vectors**. For example:

`king` - `man` + `woman`
`paris` - `france` + `uk`
`hitler` - `germany` + `italy`
`child` - `goose` + `geese`
`goes` - `eats` + `ate`
`car` - `road` + `air`

8 Come up with some analogy questions of your own. Which ones work well or poorly? What happens to the results if you use (say) $d = 10$ instead of $d = 200$?

9 Why does this work at all? Be sure to discuss the role of the vector `king` − `man` before `woman` is added. What does it tell you about the vectors that they can solve analogies?

10 Submit your extended `findsim` program on gradescope.

6. How might you use log-linear models in linguistics? This answer will be submitted only as a ✍11 Piazza note (NOT in your README). Post a new note on Piazza (a note, not a question) with a subject line "[my idea]." and Select the folder "Log-linear-model-for".

   - Think over your own interests in linguistics, NLP, or a related field.

     In your note, give an example of a conditional distribution $p(y \mid x)$ that would be interesting to model. Your goal here might be to predict y from x, or to understand the properties of x that are predictive of y.

     Be clear about both your real-world problem and your model of it. In particular, be precise about what the formal objects x and y are—numbers? strings? arrays? trees?—*and* what x and y represent in the real world. **We will post an "official example" on Piazza** to illustrate the form of a good answer.

   - ✍11 **Alternatively**, respond to someone else's Piazza post that you found interesting. Suggest some new features $f_k(x, y)$ that would be useful for their problem. Or suggest changes to the problem setup (e.g., the choice of x and y) or to other people's features.

   Be precise when describing your feature functions. Don't just say "the amount of pseudoscience words used in the text." That's a certainly a good starting thought—but how should your program identify "pseudoscience words"? (Would you need to supply it with a list of such words, or could you somehow pluck them automatically from a corpus of pseudoscience articles?) And how exactly would your feature function measure this "amount"? (Should it return an integer count? A fraction? The log of a fraction?)

   Consider designing a large, systematic set of feature functions—hundreds or thousands that *might* help prediction, not just 3 or 4 that *should* help. Hopefully, optimizing the weights will find the features that *do* help and give weight ≈ 0 to the others. For example, instead of a single feature that fires whenever it sees a pseudoscience word, you could create separate features that fire for each word from "aardvark" to "zebra," and hope that the system will learn to put a positive weight on "megalithic" and a negative weight on "t-test."

   Warning: If you're trying to predict different probabilities for different y values, then your feature vector $\vec{f}(x, y)$ needs to be different for different y values, too. Suppose you are given a text x and you're trying to predict its author's age y. If (for a given x) your features fire in the same way on all ages y, then all ages will have the same probability: $p(0 \mid x) = p(1 \mid x) = p(2 \mid x) = \ldots = p(99 \mid x) = \frac{1}{100}$. So don't use a feature like "$f_3(x, y) = 1$ if x contains an emoji." You will need a set of more specific features that depend on y, such as "$f_3(x, y) = 1$ if x contains an emoji <u>and $y < 18$</u>."[5]

[5] If this feature has positive weight, then an emoji raises the probability that the author is a minor. Specifically, it raises the probabilities that $y = 0, y = 1, \ldots, y = 17$, which means lowering the probabilities that $y = 18, y = 19, \ldots$. The bad version that doesn't test y tries to raise the probabilities of all ages at once—but this has no effect after normalization: doubling all of the unnormalized probabilities doesn't change their ratios.

7. **Extra credit:** This problem is supposed to convince you that logical reasoning is is just a special case of probabilistic reasoning.[6] That is, probability theory allows you to draw a conclusion from some premises like "**If** there's no nail, **then** there's definitely no shoe."

(a)	$p(\neg\text{shoe} \mid \neg\text{nail}) = 1$	*For want of a nail the shoe was lost,*
(b)	$p(\neg\text{horse} \mid \neg\text{shoe}) = 1$	*For want of a shoe the horse was lost,*
(c)	$p(\neg\text{race} \mid \neg\text{horse}) = 1$	*For want of a horse the race was lost,*
(d)	$p(\neg\text{fortune} \mid \neg\text{race}) = 1$	*For want of a race the fortune was lost,*
(e)	$p(\neg\text{fortune} \mid \neg\text{nail}) = 1$	*And all for the want of a horseshoe nail.*

✌12 Show carefully that (e) follows from (a)–(d). *Hint:* Consider

$$p(\neg\text{fortune}, \neg\text{race}, \neg\text{horse}, \neg\text{shoe} \mid \neg\text{nail}),$$

as well as the "chain rule" and problems 1a, 1b, and 1k.

Note: The $\neg$ symbol denotes the boolean operator NOT.

[6] An excellent and wide-ranging book developing this theme is *Probability Theory: The Logic of Science*, by the influential statistician E. T. Jaynes. See `http://bayes.wustl.edu/` for more readings.