Name: William Vagharfard StudentID: 860792866

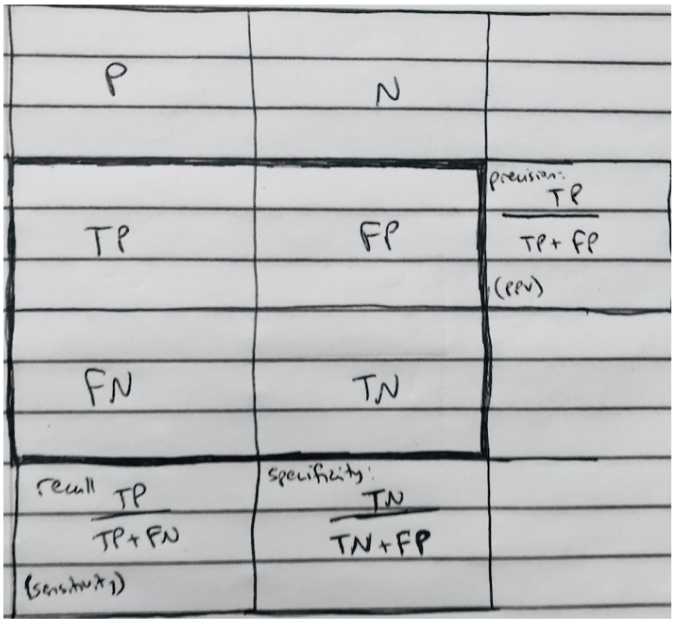
# Assignment 3

CS 173 – 2019 SP [XXX points]

Use the following probabilistic context-free grammar (PCFG) to answer questions below:

| Grammar | | Lexicon |
|---|---|---|
| *S* → *NP VP* | [.80] | *Det* → *that* [.10] \| *a* [.30] \| *the* [.60] |
| *S* → *Aux NP VP* | [.15] | *Noun* → *book* [.10] \| *flight* [.30] |
| *S* → *VP* | [.05] | \| *meal* [.15] \| *money* [.05] |
| *NP* → *Pronoun* | [.35] | \| *flights* [.40] \| *dinner* [.10] |
| *NP* → *Proper-Noun* | [.30] | *Verb* → *book* [.30] \| *include* [.30] |
| *NP* → *Det Nominal* | [.20] | \| *prefer*; [.40] |
| *NP* → *Nominal* | [.15] | *Pronoun* → *I* [.40] \| *she* [.05] |
| *Nominal* → *Noun* | [.75] | \| *me* [.15] \| *you* [.40] |
| *Nominal* → *Nominal Noun* | [.20] | *Proper-Noun* → *Houston* [.60] |
| *Nominal* → *Nominal PP* | [.05] | \| *NWA* [.40] |
| *VP* → *Verb* | [.35] | *Aux* → *does* [.60] \| *can* [40] |
| *VP* → *Verb NP* | [.20] | *Preposition* → *from* [.30] \| *to* [.30] |
| *VP* → *Verb NP PP* | [.10] | \| *on* [.20] \| *near* [.15] |
| *VP* → *Verb PP* | [.15] | \| *through* [.05] |
| *VP* → *Verb NP NP* | [.05] | |
| *VP* → *VP PP* | [.15] | |
| *PP* → *Preposition NP* | [1.0] | |

1. **Confusion Matrix** — go to Wikipedia recreate the confusion matrix below, provide at least: P, N, TP, TN, FP, FN, precision, recall, sensitivity, specificity, PPV.



Name: William Vagharfard StudentID: 860792866

2. **Vocab** — describe each term below and its relevance in your own words as it relates to n-grams, giving examples as needed.

language model –

The probability of a sequence of words happening in a language or corpus (basically a statistical model of word sequences)
The n-gram model is the probability of a certain word given the previous N-1 words.
We don't go past N3, because higher N-grams will be very computationally difficult to compute

smoothing –

modifying the probabilities that we have in order to fix the poor estimates (usually estimates of zero).
We might get poor estimates because of sparse data. With smoothing we can fix this problem by setting the ones that are zero to something else. This makes them less "jagged", and more "smooth".
There are many smoothing algorithms out there.

perplexity –

Perplexity is the probability that a language model that we have gives to the test set.
Higher perplexity is higher probability, and thus the higher perplexity model predicts the data better.

chain rule of probability –

The chain rule shows the link between computing the joint probability of a sequence vs. computing the conditional probability of a word give the previous words. You can use the chain rule to expand the probabilities of a sequence into the separate words it has.

3. If you wanted to implement what some call "auto-suggest" for query completion (what Google and other search engines do as you type a query), outline a rudimentary process or algorithm you could implement using language modelling:

To do this I would use a radix tree data structure. A radix tree is basically a trie data structure where each node that is an only child is merged with its parent node. A trie data structure is basically a prefix-tree. All the leafs of each node in the trie have common prefix strings, so lookup can be done very quickly. The reason to use a radix tree over a normal trie is because the radix tree will save a lot of space due to the fact that "single child" nodes will be merged with the parent node.

I would search and find the correct path through the tree after every single character that is inputted.
This way you can eliminate very large branches of the tree in every character, and word suggestions will be very quick.
I would personally only start the suggestion process after 3 characters have been entered, because I believe 1-3 character words are so short and quick to type that they do not need to be suggested. If I was implementing this as something that could be saved by user (i.e. on their own smartphone or something), I would keep a radix tree for the general language, using a static dictionary. However I would also keep another radix tree of the top N words by that user, so future searches can be tailored specifically per user and give quicker and more accurate results. If I wanted to implement "fuzzy searching", where the exact word did not need to be inputted to get a suggestion, I would use the Levenstein Distance function along with the radix tree in order to quickly suggest the closest possible match of words that the user is inputting.

Name: William Vagharfard StudentID: 860792866

4. **Vocab** — describe each term below and its relevance in your own words as it relates to statistical parsing, giving examples as needed.

coordination ambiguity –

the ambiguity that comes from when different parts of a sentence can be combined in different ways.
in the book they have "[old men] and [women]" which is different than "old [men and women]".

attachment ambiguity –

the ambiguity that comes when you can put (attach) a part of a sentence to different places in the parse tree.
in the book they have:
"one morning i shot [an elephant] in my pajamas" which is different than "one morning i shot [an elephant in my pajamas]"

PCFG –

probabilistic context-free grammar. In PCFG each rule is associated with a probability.
it is different from regular CFG because each rule has a conditional probability with it

non-terminal expansions –

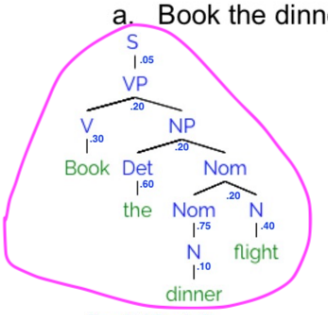
expansion in a CFG is when you make a sentence out of the rules of the CFG. You can not expand the "terminals", however the "non-terminals" (on the left hand side) can be expanded by using the rules.

VP attachment –

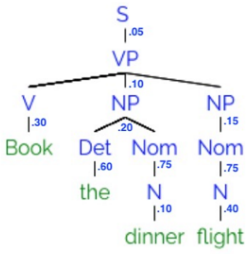
it is difference between attaching a VP (verb) to the sentence vs. attaching an NP (noun) to the sentence.
for example these two sentences are different:
1. [they accused the leader] [of stealing], but nothing came of it. ← VP attachment
2. [they accused the leader of Canada], but nothing came of it. ← NP attachment
The VP attachment is usually easier to parse and understand quickly for regular readers.

5. Give a two parse trees for the following sentences, and calculate their parse tree probabilities. Circle the *correct* parse tree according to common sense. How does it compare with the *most probable* one?

a. Book the dinner flight.



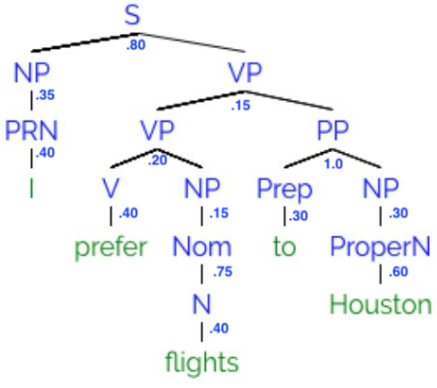

P = 2.16 e^-6



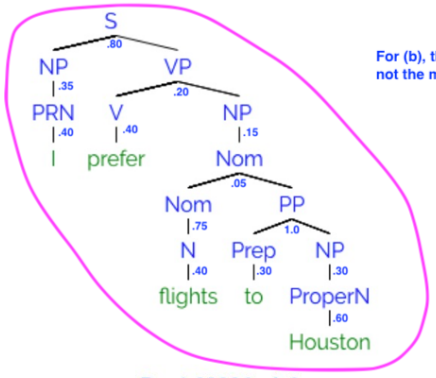

P = 6.075 e^-7

For (a), the common sense tree is also the most probable tree

b. I prefer flights to Houston.




P = 3.26592 e^-6



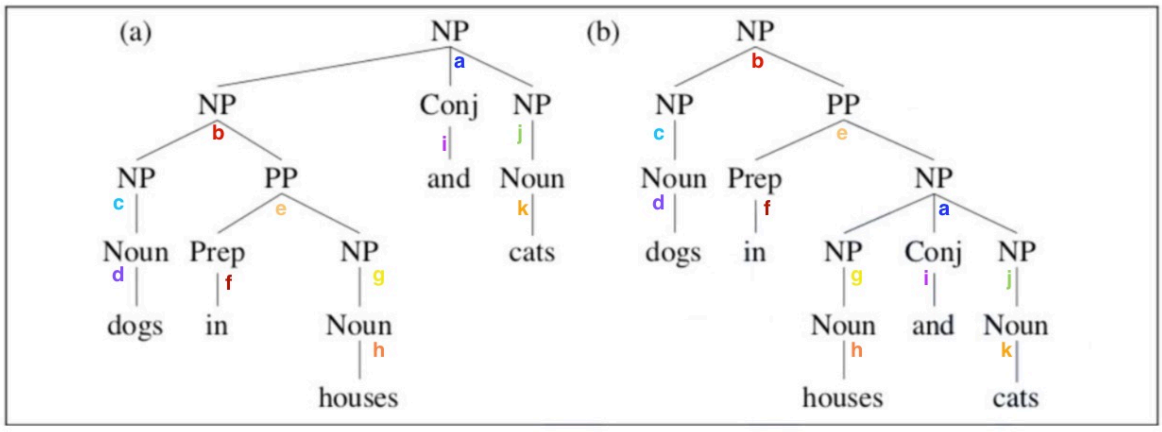

P = 1.08864 e^-6

For (b), the common sense tree not the most probable tree

* part b done using probabilities from figure 14.1

Name: William Vagharfard StudentID: 860792866

6. Statistical parsers have limitations. For example, the following example of *coordination ambiguity* cannot be resolved. Prove it: calculate and explain the parse tree probabilities.




$P(T_a)$ = a*b*c*d*e*f*g*h*i*j*k = P(T_b) $P(T_b)$ = a*b*c*d*e*f*g*h*i*j*k = P(T_a)

Explain what this is telling you:

Both of these trees have exactly the same rules, so they will both be assigned identical probabilities
This is a coordination ambiguity

7. **Vocab** — describe each term below and its relevance in your own words as it relates to vector-based or thesaurus-based models, giving examples as needed.

word-sense disambiguation –

It is the problem of choosing the correct word based on the context of the sentence.
if some words basically mean the same thing, they are said to have the same sense.
in the book they have: vegetarian (food/dish/fare). the food, dish, and fare in this context have the same sense.

naive Bayes classifier –

the nb classifier in word-sense disambiguation is trying to find the best sense for a feature vector.
it does that by saying that if you choose the one best sense out of the whole set of senses for a feature vector, it is the same as choosing the best possible sense for that vector.

KL divergence –

KL divergence shows the difference between two probability distributions. It basically shows you how much info you will lose when you choose one approximation for a data.
The smaller the number (between 0 and 1) the less info you will lose by choosing that approximation.

bootstrapping –

it is a semi-supervised learning algorithm that is used when you want to approximate/estimate a statistic from a whole data. It is basically sampling with replacement. You can create a large training set from only a small set of data points.

information content –

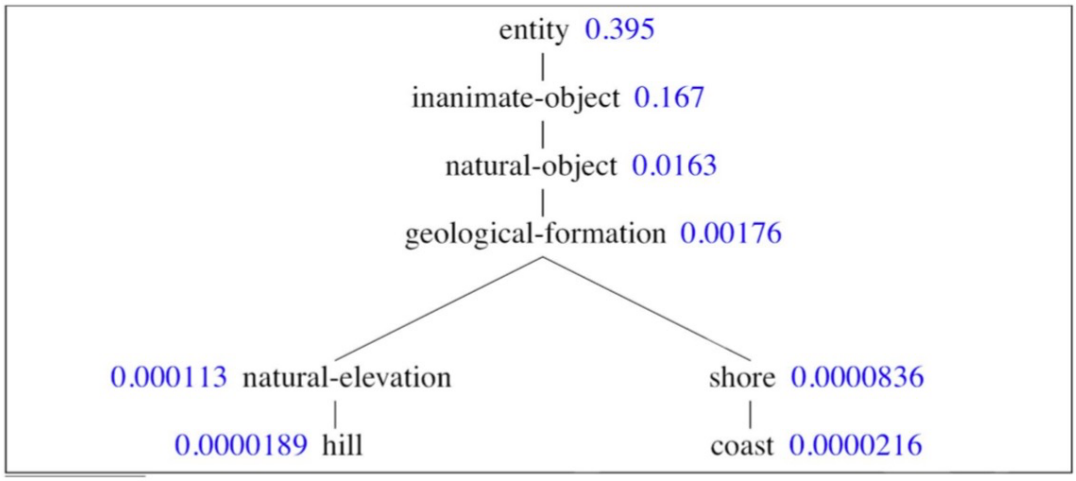
information content is the measure of the specificity of a concept. higher values of IC means a more specific concept (like: pitch-fork), while lower values mean the concept is more general (like: idea). IC is computed based on the freq count of concepts in a dataset.

LCS –

LCS is the lowest common subsumer. The LCS(x,y) is the lowest node in a hierarchy that is a hypernym of x and y.
i.e. the most specific concept which is an ancestor of both x and y.
it represents the commonality of the pair of concepts x and y.

Name: William Vagharfard StudentID: 860792866

8. Use the following WordNet hierarchy with probabilities P(c) to calculate items below:



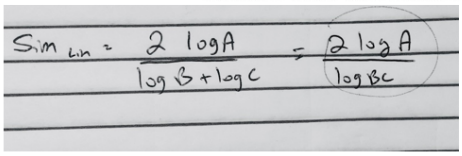

(2log(.00176)) / (log(.0000189) + log(0.0000216))

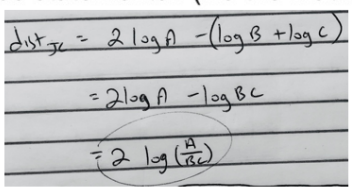
| | | |
|---|---|---|
| $sim_{path}$(hill, coast): 1/5 = 0.20 | $sim_{Resnik}$(hill, coast): -log(.00176) = 6.34 | $sim_{Lin}$(hill, coast): 0.59 |
| $sim_{path}$(hill, shore): 1/4 = 0.25 | $sim_{Resnik}$(hill, shore): -log(.00176) = 6.34 | $sim_{Lin}$(hill, shore): 0.63 |

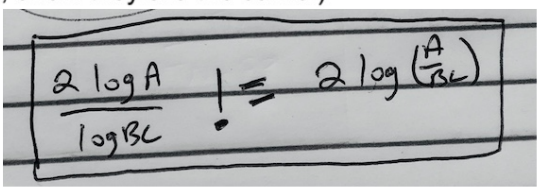
(2log(.00176)) / (log(.0000189) + log(0.0000836))

Which method makes the most sense to you? I think sim_lin is the most sense because it also calculates the probability of the word as well (how common the words are affect the calculation)

9. Examine the formula for (20.27) $sim_{Lin}$ and (20.29) $dist_{JC}$. The book states these are "similar" and that the Jiang-Conrath distance can be converted by taking the reciprocal. What's odd about these statements? (Do the math; show they are the same.)




10. **Vocab** — describe each term below and its relevance in your own words as it relates to vector-based models, giving examples as needed.

Manhattan distance –
measures the distance between vectors, showing vector similarity.
Manhattan distance is the abs. value of distance of x and y. It is not used much in NLP because it is sensitive to long vectors.

Euclidean distance –
Also measures the distance between vectors, showing vector similarity.
Euclidean distance is called the squared distance because of how it's computed.
Also not used much in NLP because of sensitivity to long vectors.

Jaccard similarity –
similarity measure that is called the min/max similarity because of how it is computed.
it is used more often because it is not sensitive to long vectors (for high freq words)

Jensen-Shannon divergence –
shows the divergence of each distribution from the mean of both distributions.
unlike the KL-divergence, it has no problem dealing with a 0 in the denominator, so it is used more often.