CPSC 477/577

Natural Language Processing Yale University

Spring 2021 Practice Questions

Part 1: Multiple Choice

**Q7.** Which of the following words start(s) with a voiced consonant? tear, cat, three, gate, part, fuse, deer

**Q1.** The following sentences show examples of what linguistic phenomenon? I had a coffee this morning (meaning “I had one cup of coffee”)

I tried two wines last night (meaning “I had two types of wine”)

I had fish for dinner (meaning “I had some fish”, not “I had a fish”)

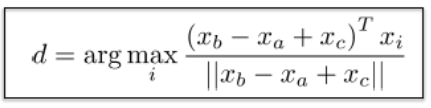
1. type coercion
2. selectional restriction
3. non-projectivity
4. backoff

**Q2.** Consider the tweet below. It is funny, partially because it involves an ambiguity. What type of ambiguity is it?

1. part of speech
2. referential
3. morphological
4. syntactic



**Q4.** What is this formula used for?



1. continuous bag of words (CBOW)
2. wordnet-based semantic similarity
3. dimensionality reduction
4. word analogy computation
5. none of the above

**Q6.** A dependency tree for a sentence with N words includes this many dependencies:

1. N/2
2. N-1
3. N
4. 2N

e. N(N-1)/2

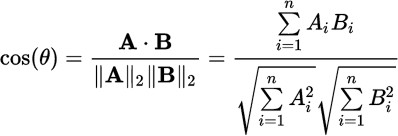
**Q7.** If f(x) is coffee beans and f’(x) is ground coffee, what is f’’(x)?

1. cup of water
2. cup of green tea
3. cup of black coffee
4. cup of venti non-fat ice caramel macchiato with extra foam, to go

**Q8.** Which of the following areas of linguistics deals with the derivation of the word "computer" from the word "compute"?

1. Inflectional morphology.
2. Derivational morphology.
3. Lexical semantics.
4. Compositional semantics.
5. None of the above.

**Q9.** The cosine similarity between the vectors (1,2,0) and (1,1,2) is: 0.55



**Q10.** A collection includes 1,000,000 documents. The word "saline" appears in 1,000 of these documents: in 200 of them it appears once in each document, in the next 500 of them it appears twice in each document, and in the final 300 of them it appears nine times in each document. What is the IDF (inverse document frequency) of the word "saline", rounded to the nearest integer? 3

**Q11.** How would you represent the sentence "Exactly one student passed the test" in First Order Logic (FOL)?

1. ∃x: student(x) 𝖠 passed (x, test) 𝖠 [∀y: (student (y) 𝖠 passed(y, test)) ==> x=y]
2. ∀x : student ( x ) ==> ¬ passed ( x, test )
3. ∃x: student (x) 𝖠 ¬ passed (x, test)
4. ¬ [ ∃x : student ( x ) 𝖠 ¬ passed ( x, test )]
5. ∃x : student ( x ) 𝖠 passed ( x, test ) 𝖠 [ ∃y : student ( y ) 𝖠 passed ( y, test ) 𝖠 x = y ]

**Q12.** Consider the following two sequences of parts of speech (each sequence corresponds to a sentence).

DT JJ NN PRP VBP TO VB DT NN IN VB IN DT NNS

DT NN IN DT NN NN RB VBZ PRP VB RP NNS

What is the maximum likelihood estimate (MLE) for the probability of bigram "NN IN"?

1. 0
2. .6
3. .25
4. .4
5. 0.1

**Q13.** What is the complex type for a preposition in CCG?

* 1. (NP\NP)/NP
  2. (S\NP)/NP
  3. NP/N
  4. NP\NP
  5. S/NP

**Q14.** Consider the following segment of a Wordnet-like ontology, augmented with subtree probabilities. What is the "Lin" similarity between chicken and dog?

| entity (1.00)

/ \

animal (0.50) building (0.50)

/ \

mammal (0.375) bird (0.125)

/ \ \

cat (0.125) dog (0.25) chicken (0.125)

(a) 0.125

(b) 0.200

(c) 0.250

(d) 0.400

(e) 0.500

**Q15.** One of the lectures used sentences containing the word "acerola" to illustrate a concept. What is that concept? Distributional Similarity

What is "acerola"?

1. a fast train connecting New York to Boston
2. a fruit with soft pulp
3. a roughly straight-line configuration of three or more celestial bodies in a gravitational system
4. a deep learning library
5. a Pokemon character

**Q16.** According to the rules of ITG (Inversion Transduction Grammar), how many reorderings are allowed for the production

NP -> ART CARD JJ NN?

1. 1
2. 2
3. 4
4. 16
5. 24

**Q17.** True or False: Given the feature structures FS1 and FS2,

FS1

[ agr = [ number = 'singular' ] ] [ [ person = 1 ] ]

[ ]

[ type = 'NP' ]

FS2

[ agr = [ number = ?n ] ] [ ]

[ subj = [ number = ?n ] ]

the output of their unification Unify (FS1, FS2) is correctly shown below.

FS3

[ agr = [ number = 'singular' ] ] [ [ person = 1 ] ]

[ ]

[ subj = [ number = 'singular' ] ] [ ]

[ type = 'NP' ]

1. TRUE
2. FALSE
3. no idea

**Q18.** IBM Models 1, 2, and 3 are used for:

1. statistical machine translation
2. neural machine translation
3. constituent parsing
4. dependency parsing
5. none of the above

**Q19.** In order to prevent the possibility of an artificially high machine translation score, BLEU includes the following component:

1. counting crossing brackets
2. finetuning
3. “forget” gates
4. brevity penalty
5. subcategorization

**Q20.** What grammatical formalism is specifically designed to handle sentences like this one: “I bought a hat for my son and a book for my daughter”?

1. Head Driven Phrase Structure Grammar (HPSG)
2. ~~Dependency Grammar~~
3. Regular Grammar
4. Context Sensitive Grammar (CSG)
5. Combinatory Categorial Grammar (CCG)

**Q21.** Given the sentence “When you get home, I will have fixed the sink”, in what order do the utterance (U), reference point (R), and event (E) occur? The “<” symbol below means “precedes”.

1. E < R < U
2. U < E < R
3. U < (R,E)
4. (U,R) < E
5. E < U < R

**Q22.** Which of the following techniques is \*not\* used by BERT:

1. bidirectional encoding
2. attention
3. parallel corpora
4. stacked encoders
5. position embeddings

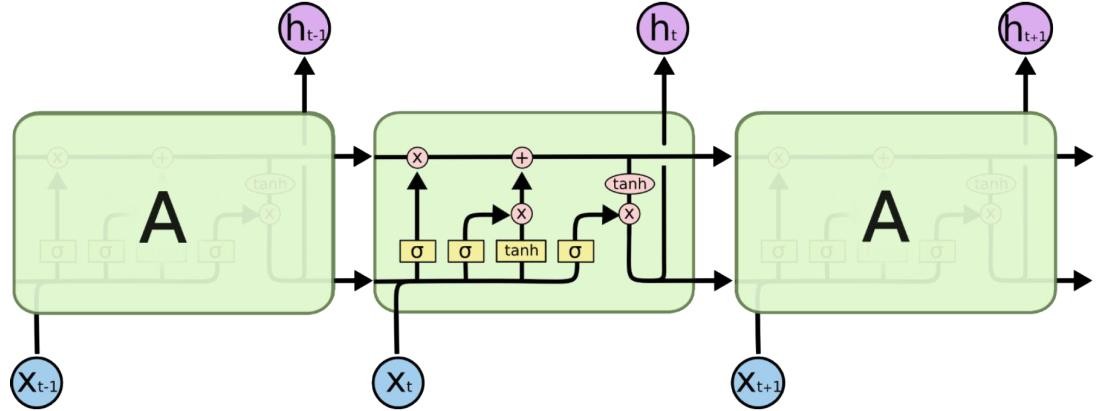
**Q23.** What type of neural network uses the equations below?

ℎ𝑡=𝜎(𝑊ℎℎ𝑡−1+𝑊𝑥𝑥𝑡)

𝑦𝑡=𝑠𝑜𝑓𝑡𝑚𝑎𝑥(𝑊𝑦ℎ𝑡)

1. recurrent neural network
2. recursive neural tensor network
3. convolutional network
4. gated recurrent unit
5. long short term memory network

**Q24.** What type of neural network is represented here (image from Chris Olah)?



1. recurrent neural network
2. recursive neural tensor network
3. convolutional network
4. gated recurrent unit
5. long short term memory network

**Q25.** In the sentence “She is eating a red apple”, which of the following pairs of words would exhibit the \*highest\* attention scores?

1. she, red
2. is, a
3. eating, red
4. she, apple
5. eating, apple

# Question 21

The sentence "My aunt collects **old** clocks and silverware" illustrates what type of ambiguity?

1. Prepositional phrase (PP) ambiguity
2. The sentence is not ambiguous.
3. Dialogue ambiguity d.
4. Coordinating conjunction (CC) ambiguity

M2. Which of the following statements about ambiguity in (English) part of speech tagging is the most accurate? The lecture gave an example from the Brown corpus.

a. The percentage of ambiguous tokens is about the same as the percentage of ambiguous types.

b. The percentage of ambiguous tokens is slightly smaller than the percentage of ambiguous types.

c. The percentage of ambiguous tokens is significantly larger than the percentage of ambiguous types.

d. The percentage of ambiguous tokens is slightly larger than the percentage of ambiguous types.

e. The percentage of ambiguous tokens is significantly smaller than the percentage of ambiguous types.

M4. Which of the following features are not typically used for sentence boundary recognition?

a. punctuation

b. capitalization

c. word length

d. case (upper case, lower case)

# Question 54

Consider the noun-noun phrases such as "cat food", "baby goat", "attorney general". What is/are some reasonable CCG parses for these:

(a) N/N N ->NP

(b) N N\N ->NP

1. N/N N/N -> NP
2. both a and b

N/N N -> NP: This parse treats the first noun as a function that takes a noun to its right and returns a noun phrase.

N N\N -> NP: This parse treats the second noun as a function that takes a noun to its left and returns a noun phrase. This means the second noun is modifying the first. It's a less common interpretation in English but could be used in cases where the second noun somehow restricts or modifies the first.

# Question 57

Adverbs are used to specify all of the following EXCEPT:

1. place
2. time
3. manner
4. degree
5. agent

# Question 58

The hypernym and hyponym relations in WordNet hold between which of the following notions (pick one):

1. words
2. lemmas
3. stems
4. synsets

"vehicle" is a hypernym of "car", and "car" is a hyponym of "vehicle".

# Question 59

The dependency representation of the sentence "Jane has a cat" is the following (with the arrows pointing from parent to child node):

1. has -> cat, has -> Jane, cat -> a
2. cat -> has, Jane -> cat, a -> cat
3. has -> Jane, Jane -> cat, cat -> has
4. Jane -> has, has -> a, a -> cat
5. has -> Jane, Jane -> cat, a -> cat

# Question 61

Which of the following statements about syntactic constituents is false?

1. Constituents are non-crossing.
2. Each word is a constituent.
3. If two constituents share one word, then one of them must completely contain the other.
4. Constituents are continuous.
5. Constituents cannot be nested.

# Question 62

Assuming that exactly two constituents get combined at each iteration, a sequence of three nouns can be parenthesized in two different ways:

(a(bc)) and ((ab)c).

A sequence of four nouns can be parenthesized in five different ways: ((ab)c)d (a(bc))d (ab)(cd) a((bc)d) a(b(cd)).

In how many ways can a sequence of five nouns (or, alternatively, an adjective followed by four nouns) be parsed? 14

This sequence is known as the Catalan numbers, which counts certain types of lattice paths, permutations, binary trees, and many other combinatorial objects as well. For n=5, the Catalan number is 14.

1. ((((ab)c)d)e)
2. (((ab)(cd))e)
3. ((ab)((cd)e))
4. (((ab)c)(de))
5. ((ab)(c(de)))
6. ((a(bc))d)e
7. (a((bc)d))e
8. (a(bc))(de)
9. ((a(bc))de)
10. (a((bc)de))
11. (a(b(cde)))
12. ((a(b(cd)))e)
13. (a((b(cd))e))
14. (a(b((cd)e)))

# Question 63

Which of the following statements is true?

The operator ">" here means "strictly more expressive (powerful) than"

CCG = Combinatory Categorial Grammar, CSG = Context Sensitive Grammar, CFG = Context Free Grammar, TAG = Tree Adjoining Grammar, TSG = Tree Substitution Grammar.

mildly context sensitive grammar: CCG, TAG, TSG

1. CSG > CFG > TAG
2. CSG >TSG >CFG
3. CSG >CCG >CFG
4. CCG > CSG > CFG
5. CSG > CFG > CCG

**Q1.** What property (properties) of Tree Adjoining Grammars makes them (strictly) more powerful than Context Free Grammars?

TAG can model cross serial dependency by an adjunction operator.

# Question 64

Consider the corpus:

cat cat cat dog dog rat rat bat bat bat bat bat bat fox

Using "Add One" Laplacian smoothing, what is the estimate for P(rat)?

(2+1)/(14+5)=3/19

# Question 65

A regular die has 6 sides, numbered from 1 to 6. If you throw two regular dice together, what is the probability that their sum is an even number?

(a) 1/4 (b) 1/18

(c) 1/6

(d) 1/3

(e) 1/2

# Question 66

Which of the following statements about evaluation of relation extraction is \*not\* correct?

1. Precision = #correctly extracted relations / #all extracted relations
2. F1 = harmonic mean of Precision and Recall
3. Recall = #correctly extracted relations / #all existing relations
4. F1 = arithmetic mean (average) of Precision and Recall
5. F1 = 2 x Precision x Recall/(Precision+Recall)

# Question 67

Which of the following part of speech categories is open class?

(a) adjectives

1. prepositions
2. interjections
3. articles
4. conjunctions

open class: new words can be added over time, including nouns, verbs, adjectives, adverbs

closed class: new words are rarely added,

including pronouns, prepositions, conjunctions, interjections

# Question 68

What does this logical expression mean in English?

∃e: Arriving(e) 𝖠 Arriver(e, Speaker) 𝖠 Destination(e, NewYork) 𝖠 IntervalOf(e,i) 𝖠

EndPoint(i,p) 𝖠 Precedes(p, Now)

1. I am arriving in New York
2. He is arriving in New York
3. I will arrive in New York

(d) I arrived in New York

(e) She will arrive in New York

"There exists an event 'e' such that 'e' is an arriving event. The speaker is the one who arrived (Arriver) in this event. The destination of this arrival event is New York. The event 'e' occurs within an interval 'i', and 'i' is the endpoint of a period 'p'. This period 'p' precedes the current time (Now)."

# Question 69

The CKY (Cocke-Kasami-Younger) parsing algorithm only works when...

1. the grammar only includes a single production for any non-terminal.
2. the grammar has been converted to Chomsky Normal form.
3. the input sentence has exactly one verb.
4. the input sentence is in English.
5. the input sentence is not ambiguous.

# Question 70

In the absence of any other relevant information, how should an out of vocabulary (OOV) word be tagged?

1. determiner
2. adverb
3. noun
4. verb
5. adjective

# Question 71

The sentence "Stolen painting found by tree" exhibits what phenomenon:

1. prepositional phrase attachment ambiguity
2. coordinating conjunction attachment ambiguity
3. syntactic ambiguity
4. all of the above
5. none of the above

The stolen painting was found next to a tree (where "by" indicates location). The tree found the stolen painting (where "by" indicates agency).

# Question 72

What part of speech is \*least likely\* after an article? 冠词: a, an, the

1. noun
2. adjective
3. verb
4. numeral

# Question 73

In the Bayes formula: P(H|E) = P(E|H)P(H)/P(E), what do "H" and "E" stand for?

(a) H=hypothesis, E=evidence

1. H=hyperparameter, E=evidence
2. H=hyperparameter, E=estimate
3. H=hypothesis, E=estimate
4. none of the above

# Question 74 Zipf law

In a large corpus, the frequencies of the three most frequent words are approximately 10%, X%, and 3%. What is the value of X, assuming a Zipfian distribution?

1. 9
2. 7
3. 6
4. 5
5. 4

Given that the most frequent word occurs 10% of the time, the second most frequent word (n=2) should occur approximately half as often. So, in this case, X would be approximately 5%.

**Q6.** A corpus contains 1,000,000 tokens, including 6,000 instances of the word "the". The next four most frequent words are "of", "and", "to", and "a". What are the expected counts for each of these four words in the corpus?

"of": 6000\*1/2=3000

"and": 6000\*1/3=2000

"to": 6000\*1/4=1500

"a": 6000\*1/5=1200

# Question 75

In language modeling, the "add-1" method is an example of:

1. linear interpolation
2. backoff (c)smoothig
3. caching
4. hypothesis testing

# Question 76

An experiment was done to measure the perplexity of unigram, bigram, and trigram models on a news corpus corpus.

Which of the following sets of numbers makes the most sense?

1. unigram 1000, bigram 200, trigram 100
2. unigram 100, bigram 100, trigram 100
3. unigram 100, bigram 200, trigram 1000
4. unigram 100, bigram 0, trigram 0

# Question 77

For a sentence with n words, the maximum number of boxes in the CKY table that can be non- empty is:

1. n\*n\*n
2. n
3. n log n

(d) n\*(n-1)/2

(e) n\*(n+1)/2

# Question 79

Which of the following sentences is non-projective:

1. The non-callable issue, which can be put back to the company in 1999, was priced at 99 basis points above the Treasury’s 10-year note.
2. John saw a dog yesterday which was a Yorkshire Terrier.
3. Price details weren't immediately available.
4. The collateral is being sold by a thrift institution.
5. Ms. Haag plays Elianti.

Non-projectivity in a sentence refers to the condition when there are crossing dependency edges in the dependency parse of the sentence.

This usually happens when a word in a sentence is more closely related (in terms of the sentence's meaning or syntax) to a word that's not right next to it, compared to the words that are adjacent.

The relative clause "which was a Yorkshire Terrier" is associated with "a dog," creating a non-projective parse as the dependency crosses over other dependencies in the sentence.

# Question 80

Which of the following relation(s) is/are symmetric:

1. brother(X,Y)
2. sister(X,Y)
3. mother(X,Y)
4. both a. and b. above
5. none of the above

# Question 81

Which of the following propositional logic statements is always true? The symbol "==" here is used to express equivalence.

(a) NOT (A AND B) == (NOT A) OR (NOT B) (b) NOT (A OR B) == (NOT A) AND (NOT B)

1. NOT A == NOT B
2. A == NOT B
3. more than one of the above

# Question 82

Which of the following is true:

1. English is an SOV language, Japanese is an SVO language
2. English is an SOV language, Japanese is an SOV language
3. English is an SVO language, Japanese is an SVO language
4. English is an SVO language, Japanese is an SOV language
5. English is an SOV language, Japanese is an VSO language

# Question 83

The BLEU evaluation metric is essentially:

1. n-gram recall with a penalty for brevity
2. n-gram recall with a penalty for excess length
3. n-gram precision with a penalty for brevity
4. n-gram precision with a penalty for excess length
5. none of the above

Part 2: short answer

# Question 44

Explain each of these terms and give an example (or formula, if appropriate) of each:

**LCS (lowest common subsumer)**: the most specific concept which is an ancestor of two or more concepts.WordNet can be used to find the LCS of two or more synsets (concepts).

Animal

/ \

Mammal Fish

/ \

Dog Cat

LCS of "Dog" and "Cat" is "Mammal",

The LCS of "Dog" and "Fish" is "Animal"

**Q2.** What WordNet node (synset) is likely to be the Lowest Common Subsumer (LCS) for

*eagle* and *goose* (in their most frequent senses as nouns).

Bird

**the principle of semantic compositionality** – meaning of a whole is a function of the meanings of simpler parts and the way those parts are put together.

**horizontal markovization within lexicalization**, label with the left siblings in your immediate tree. This takes into context where you are in your local tree structure.

**labeled dependency accuracy** takes into account heads and labels when computing accuracy

**HMM trellis** represents the possible state sequences

**negative sampling (for word embeddings)** a way of approximating the softmax by using a small number of randomly selected contexts for the denominator for efficiency in training word embeddings.

**confusion matrix for binary classification** is a table which specifies predicted values vs actual values for binary classification and is useful for calculating statistics such as recall and precision; shows true/false positives and negatives

**backpropagation over time for RNN** – a method used in gradient descent for recurrent networks. It begins by unfolding the network over time and then applying backprogagation to find the gradient of the cost with respect to all parameters

**softmax function** takes in an input vector of real numbers and normalizes it into a probability distribution

**Linguistics**

**ambiguity**

# Question 26 Winograd schema

Give an example of a sentence that would be a valid Winograd schema.

A Winograd schema is a type of linguistic challenge involving sentence pairs that are identical except for one or two words, and which require commonsense knowledge to disambiguate. Here's an example:

"The city councilmen refused the demonstrators a permit because they feared violence."

"The city councilmen refused the demonstrators a permit because they advocated violence."

In both sentences, the pronoun "they" could logically refer to either the city councilmen or the demonstrators, but our understanding of the context and common sense tells us that in the first sentence "they" refers to the city councilmen (who fear the potential for violence), and in the second sentence "they" refers to the demonstrators (who are advocating violence).

morphology

# Question 12

Give an example of each of the following instances of morphological derivation:

verb → noun: run, runner

adjective → noun; good, goodness

adjective → adverb; great, greatly

adjective → verb. white, whiten

# Question 1

Given the stem “snow”, give one example each of words that would result from the following kinds of morphological processes involving snow:

# Compounding: ~~snow white~~ snowball

1. **Inflection: snowing snowed**

# Derivation: snowman snowy

# Question 11

For each of the following linguistic and computational phenomena give an example sentence and explain in a paragraph or two why it presents a challenge to natural language processing systems:

1. garden path sentence
2. inflectional morphology
3. coordination ambiguity

**(a)** “The horse raced past the barn fell.” – This can be challenging because most NLP systems these days are statistical rather than rule-based. A rule-based system could likely rule out the incorrect parse here, but a statistical system could get confused here since “the horse raced past the barn” (without the last verb) is a likely sentence, and even with “fell” mistagged could be considered a more likely parse than the correct parse.

**(b)** “Omnēs enim quī accēperint gladium, gladiō perībunt.” (student-provided answer)– I chose a non-English sentence here because this is a much bigger problem in  
languages with more inflection than English (which has very little inflection left anymore). The sentence means roughly “For all who take up the sword will die by the sword”. The word for “take up” here is “accēperint”. The morphology of this verb is actually ambiguous. It could be either a perfect subjunctive, meaning “might have taken up the sword” or a future perfect “will have taken up the sword”. It's actually not even 100% clear to a human reader which of the two it is, much less so for a computer.

If we're considering this in the context of a machine translation system, before the system could even decide whether to translate “might have taken up” or “will have taken up”, it needs to know what verb it's looking at. “accēperint” is an inflected form of the verb “accipio”. But as it turns out, “accēperint” is a fairly uncommon inflection. (“might have” and “will have” are uncommon things to say) It's quite possible that the system will never have seen the word “accēperint” before (even if it's seen other perfect subjunctive or future perfects before), and if it didn't have a way to process morphology in a general way, it may not know that this is a form of “accipio”.

**(c)** “John and Jane like apples and oranges” – This sentence is ambiguous, even for a human. If you are trying to discover the semantics of this sentence, for example, it's not clear whether the correct answer should include “like(John, oranges)”

**compositional semantics**

# Question 27

Which (zero, one, or more) of the following phrases are non-compositional?

1. white cloud
2. red herring 障眼法
3. green tree
4. black market 黑市，非法交易的市场
5. blue box

# Question 6

English has the wonderful feature that it lets you stick two nouns together into a compound noun, whose meaning derives in some idiosyncratic way from the meanings of its parts:

water fountain: a fountain that supplies water water ballet: a ballet that takes place in water

water meter: a device (called meter) that measures water

water barometer: a barometer that uses water instead of mercury (to measure air pressure) water biscuit: a biscuit that is made with water

water glass: a glass that is meant to hold water

Even more fun is that one of the two nouns in the compound noun could itself be a compound noun, as in the case of ice cream soda. But what's the recipe for that beverage? It depends. You make [[ice cream] soda] by dropping ice cream into soda, but you make [ice [cream soda]] by dropping ice into cream soda.

1. The paragraph above used [square brackets] to distinguish two possible meanings of ice cream soda, one of them being the conventional meaning. Add brackets to each compound below to indicate whether the most likely meaning corresponds to [[X Y] Z] or [X [Y Z]].
2. [[ice cream] soda]
3. [[science fiction] writer]
4. [[customer service] representative]
5. [state [chess tournament]]
6. [[Mars Rover] landing]
7. [plastic [water cooler]]
8. [[typeface design] report]
9. Choose the most likely bracketing for the 4-word compound noun country song platinum album.
   1. [country [song [platinum album]]]
   2. [country [[song platinum] album]]
   3. [[country song] [platinum album]]
   4. [[country [song platinum]] album]
   5. [[[country song] platinum] album]

Give a plausible definition of [[space mission] [[control freak] show]]. (If you must use compound nouns in your definition, define them too.)

**A reality television show about people in space who must control the entire mission. (a reality television show is a drama on television that is unscripted)**

1. Show the most likely bracketing for the 8-noun sequence below. As in the examples above, your bracketing must have the form [X Y], where each of X and Y is either a single-word noun or a compound noun (which must also be written as a bracketing [X Y] and so on.)

family board game togetherness effect government study

[[[family [board game] [togetherness effect]] [government study]] author]

1. A computer program knows less about the world than you do, so it may have more trouble interpreting these sequences of nouns. How many bracketings must it choose among? Complete the following table by inserting the correct number for f(5). Bonus if you give f(6) as well.

f(1) = 1

f(2) = 1

f(3) = 2

f(4) = 5 (see part B. above)

f(5) = ???

f(6) = ???

1. This can be done recursively. 5 words can be broken down 4 ways:

5 → 1, 4 = 1 bracketing \* 5 bracketings = 5 bracketings

5 → 2, 3 = 1 bracketing \* 2 bracketings = 2 bracketings

5 → 3, 2 = 2 bracketings \* 1 bracketing = 2 bracketings

5 → 4, 1 = 5 bracketings \* 1 bracketing = 5 bracketings.

This is 14 total. To f(5) = 14.

f(6) can be computed similarly:

6 → 1, 5 = 1 \* 14 = 14

6 → 2, 4 = 1 \* 5 = 5

6 → 3, 3 = 2 \* 2 = 4

6 → 4, 2 = 5 \* 1 = 5

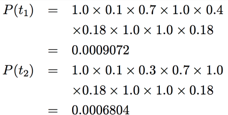
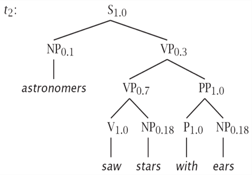
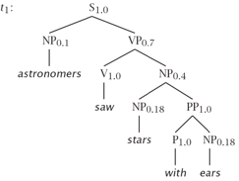
6 → 5, 1 = 14 \* 1 = 14

So f(6) = 42

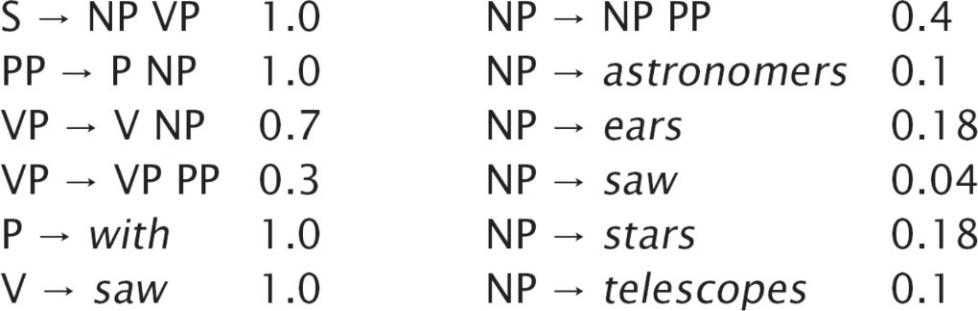
# Question 50 syntactic parsing

1. The following sentence contains a syntactic ambiguity. Draw two possible parse trees for the sentence.

*astronomers saw stars with ears*



1. Assume the following probabilistic rules.



Calculate the probability of each tree that you drew. According to your results, which parse is more probable?

# Grammar

# Question 13

Write a simple context-free grammar that can produce the following sentences:

1. Joe ate dinner.
2. Did Maureen eat dinner?
3. When did Joe eat dinner?
4. Eat Dinner!

|  |  |
| --- | --- |
| S → NP VP S → VP SQ → VBD NP VP  S → WHADVP SQ  WHADVP → WRB  NP → NNP NP → NN VP → VB NP VP → VBD NP | NNP → Joe  NNP → Maureen  NN → dinner  VBD → did VBD → ate  WRB → when  VB → eat |

**Probabilistic Context Free Grammars (PCFG)**

# Question 38

Probabilistic Context Free Grammars (PCFG), by design, make certain independence assumptions that can hurt the performance of a natural language parser. Techniques such as parent annotation, constituent splitting, and vertical markovization are often used to mitigate such shortcomings. In this example, we will split the NP constituent into NP-SUBJ and NP-OBJ.

Let’s consider the following probabilities:

p = P (NP → PRP)

n = 1-p

ps = P (NP-SUBJ → PRP)

ns = 1-ps

po = P (NP-OBJ → PRP)

no = 1-po

Notes: PRP is a personal pronoun (such as “I”, “she”, and “they”). NP-SUBJ is a noun phrase that serves as the subject of the sentence. NP-OBJ is a noun phrases that serves as the object of the sentence.

Assume a PCFG trained on the Wall Street Journal portion of the Penn Treebank. Which of the following inequalities is/are likely to be accurate? Pick 0, 1, or more answers.

a. <

b. >

c. >

d. >

**Q3.** Write down a PCFG such that:

1. Any sentence consisting of the word “the” *n* times in a row, where *n ≥* 1, has probability 0.4 × 0.6n-1
2. Any other sentence has probability 0.

One possible PCFG is:

S -> A  
A -> the p=.4  
A -> A the p=.6  
Any sequence of n the’s requires 1x the first rule, n-1x the last rule, and 1x the middle rule.

# Question 3

Consider the five sentences shown in Figure 1. Think of them as your training data to build a probabilistic context-free grammar (PCFG).

Delta flight 411 leaves Toronto for Atlanta at 6 PM This flight serves a light meal

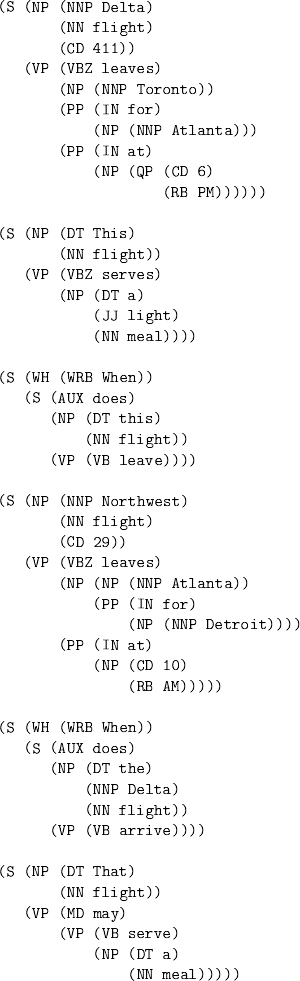
When does this flight leave

# Northwest flight 29 leaves Atlanta for Detroit at 10 AM When does the Delta flight arrive

That flight may serve a meal

Figure 1: Training sentences.

The parse trees for these sentences are shown in Figure 2



Build a PCFG using the training data. For each non-lexical rule (e.g., S → NP VP), indicate its probability.

|  |  |  |  |
| --- | --- | --- | --- |
| S -> NP VP | 4/8 | NP -> DT JJ NN | 1/15 |
| S -> WH S | 2/8 | VP -> VBZ NP PP | 1/7 |
| S -> AUX NP VP 2/8 |  | VP -> VBZ NP PP PP | 1/7 |
| NP -> NNP NN CD | 2/15 | VP -> VBZ NP | 1/7 |
| NP -> NP PP | 1/15 | VP -> VB | 2/7 |
| NP -> NNP | 4/15 | VP -> MD VP | 1/7 |
| NP -> CD RB | 1/15 | VP -> VB NP | 1/7 |
| NP -> QP 1/15 |  | PP -> IN NP | 1 |
| NP -> DT NNP NN | 1/15 | QP -> CD RB | 1 |
| NP -> DT NN | 4/15 | WH -> WRB | 1 |

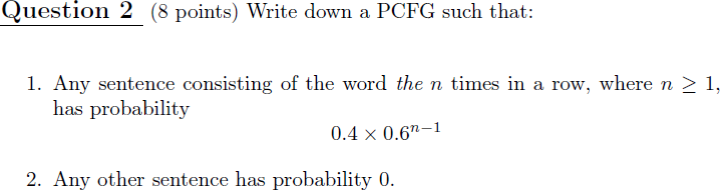
1. Build a probabilistic lexicon for the PCFG. Give the probability of each lexical rule (e.g., NN

→ flight).

|  |  |
| --- | --- |
| NNP -> Delta | 2/7 |
| NNP -> Toronto | 1/7 |
| NNP -> Atlanta | 2/7 |
| NNP -> Detroit | 1/7 |
| NNP -> Northwest | 1/7 |
| NN -> flight | 6/8 |
| NN -> meal | 2/8 |
| CD -> 411 | ¼ |
| CD -> 6 | ¼ |
| CD -> 29 | 1/4 |
| CD -> 10 | 1/4 |
| VBZ -> leaves | 2/3 |
| VBZ -> serves | 1/3 |
| IN -> for | 1/3 |
| IN -> at | 2/3 |
| RB -> PM | ½ |
| RB -> AM | 1/2 |
| DT -> that | 1/6 |
| DT -> a | 2/6 |
| DT -> the | 1/6 |
| DT -> this | 2/6 |
| VB -> serve | 1/3 |
| VB -> arrive | 1/3 |
| VB -> leave | 1/3 |
| WRB -> when | 1 |
| AUX -> does | 1 |
| JJ -> light | 1 |
| MD -> may | 1 |

1. Smoothing (reserving probability mass for unobserved rules is very important when building PCFGs. Redo parts (a) and (b) above using 10% of the probability mass for each non-terminal or lexical category to cover unknown words. Example: if A → B C has a probability of .6 and A → B D has a probability of .4, you need to create a new rule A → α with a probability of .1 and readjust downward the probabilities of the other two rules that have A on the left-hand side.
2. For each of the following two sentences "When does Northwest flight 77 leave for Milwaukee" and "Does this flight leave for Milwaukee", draw **one** parse tree according to the (smoothed) grammar in part (c). If you are getting any zero probabilities, return to part (c) and fix you grammar appropriately. What are the final probabilities for each of these two sentences? For this question, you don't need to find all possible parses of a given sentence. One parse per sentence will be enough.

**Question 8**

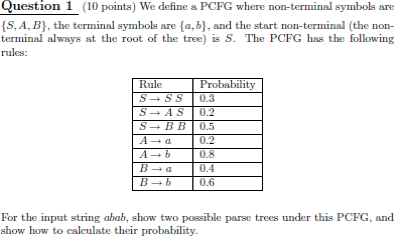


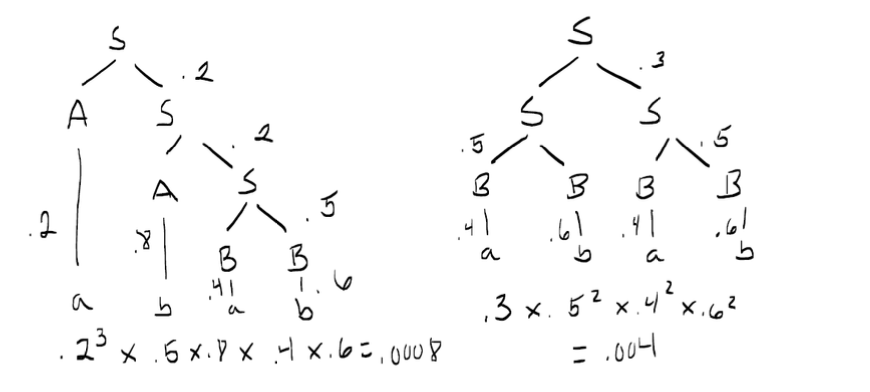
One possible PCFG is:

S -> A p=1  
A -> the p=.4  
A -> A the p=.6

Any sequence of n the’s requires 1x the first rule, n-1x the last rule, and 1x the middle rule.

**Question 2**





# Question 5 mildly context-sensitive grammars

Why are “mildly context-sensitive grammars” like Tree Adjoining Grammars (TAG) and Combinatory Categorial Grammars (CCG) used in NLP? Give two reasons.

mildly context-sensitive grammars are able to model grammatical phenomena that context-free grammars are not able to. *For example, any of these grammars listed are able to model the language anbncn, which is not context-free.*

Can capture cross-serial dependencies

# Question 12 Combinatory Categorial Grammars (CCG)

Consider the sentence

Marina often gives Bill cold water

1. Write a linguistically motivated **CCG** lexicon entries for the six **words** in this sentence. Make sure to capture the correct subcategorization frame for “gives”. Use standard symbols, such as S, S\NP, etc.

|  |  |  |
| --- | --- | --- |
| Marina  often | NP  (S\NP)/(S\NP) | (or N) |
| gives | ((S\NP)/NP)/NP |  |
| John | NP | (or N) |
| cold | NP/NP | (or N/N) |
| water | NP | (or N) |

(b) Show the full CCG parse for the sentence.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Marina N | often (S\NP)/(S\NP) | gives ((S\NP)/NP)/NP | John N | cold N/N | water N |
| NP |  |  | NP | N | |
|  |  | (S\NP)/NP | | NP | |
|  | (S\NP)/NP | | | NP | |
|  | S\NP | | | | |
| S | | | | | |

# Question 48

Write a CCG lexical entry for a transitive verb such as “watch” in the following sentence: “Akira watched the movie”.

(S\NP)/NP

# Question 53

Complete the following CCG derivation.

1. replace the four instances of ???? with words that form an appropriate sentence.
2. Fill in the rest of the derivation in the empty space.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| ???? | ???? | ???? | ???? |
| NP | ((S\NP)/NP) | (NP/N) | N |

S

# CKY parsing

# Question 47 CKY parsing

Why does the CKY parsing algorithm for CFG require the grammar to be in CNF?

at each step, the CKY chart can only combine two items into one

# Question 18

What is the worst-case time complexity of the CKY-CFG parsing algorithm used to find all parses for a sentence.

O(*n*3x |*G*|) where n is the length of the parsed string and |*G*| is the size of the grammar (note: G in CNF)

# Question 51

Fill in the Probabilistic CKY chart below for sentence: *Time flies like an arrow*

Assume the following rules and weights: 1 S → NP VP

6 S → Vst NP

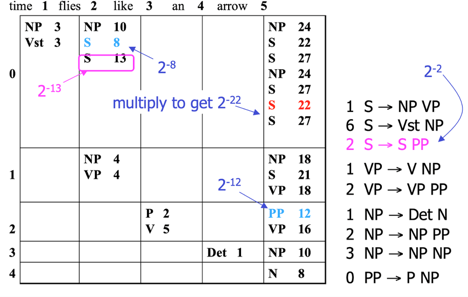
2 S → S PP

1. VP → V NP
2. VP → VP PP
3. NP → Det N
4. NP → NP PP
5. NP → NP NP

0 PP → P NP

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Time | flies | like | an | arrow |
|  | 1 | 2 | 3 | 4 |  |
| 0 | NP 3  Vst 3 |  |  |  |  |
| 1 |  | NP 4  VP 4 |  |  |  |
| 2 |  |  | P 2  V 5 |  |  |
| 3 |  |  |  | Det 1 |  |
| 4 |  |  |  |  | N 8 |

Extra credit: recover the best parse by tracing back-pointers.



# Question 17

Rewrite the grammar below in Chomsky Normal Form:

A  B C D

A  B h A  a | F B  b

C  c

D  d | e F  f|g

Where A, B, C, D, F are non-terminals, A is the start symbol, and a, b, c, d, e, f, g, h are terminals.

**A**B X | a | f | g

Bb

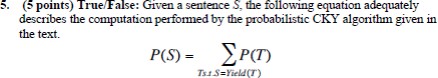
Cc

Dd | e

XC D

Xh

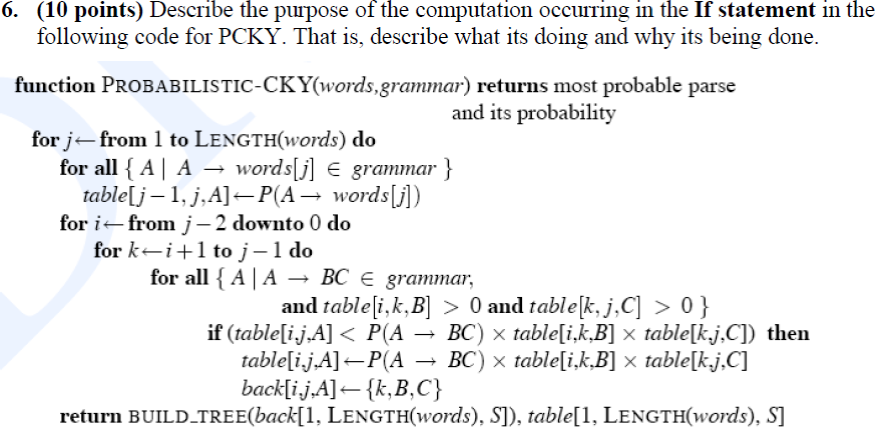
**Question 4**



True, the probability of a sentence is equal to the sum of probabilities of all possible parse trees of this sentence.

A possible parse tree is any tree that yields the sentence (sequence of tokens).

# Question 5



The if statement is the key step in building the table. Essentially, we are trying to determine the most probable parse tree for word sequence i...j. Therefore, we iterate over all possible rules in the grammar which have a nonzero probability for this sequence and split point. The if statement checks each split point and rule to see if they produce the highest probability tree for the sequence i...j. If so, we store that value.

This algorithm works from the bottom up and starts with smaller phrases, building off those until the most probably parse is determined for the whole tree. Until the most probable parse is determined for the whole tree.

# Question 45 shift-reduce parsing

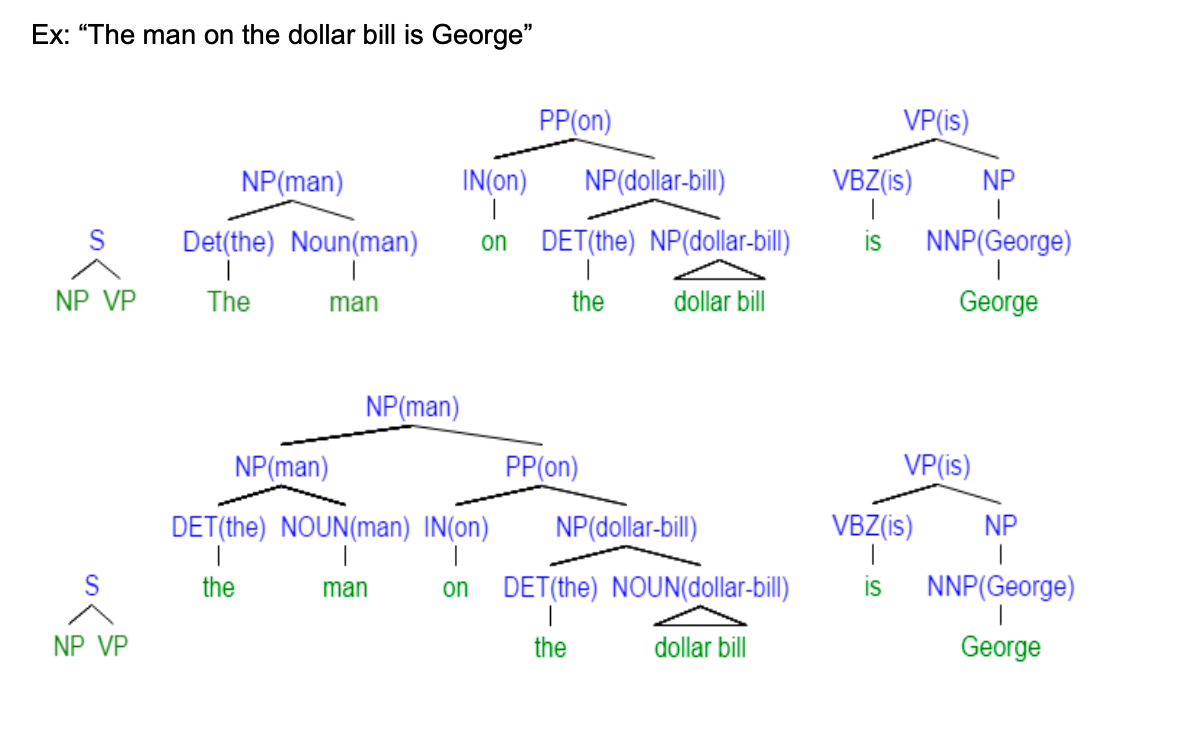
How does shift-reduce constituent parsing work?

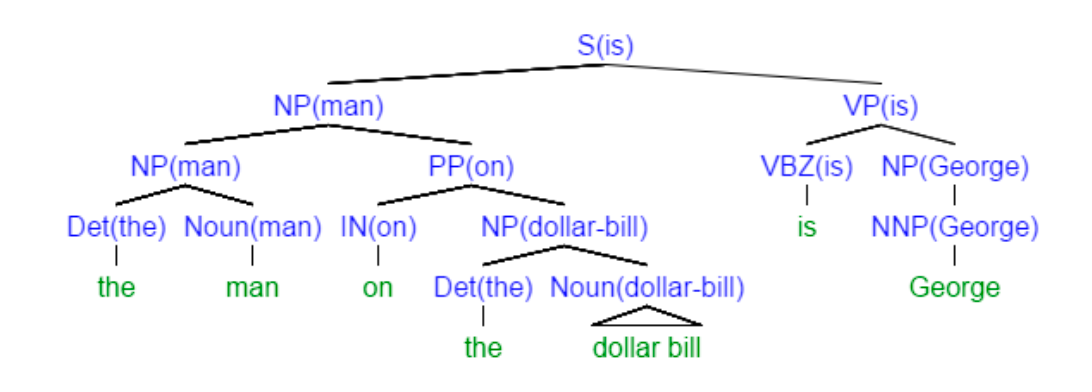
A shift-reduce parser scans and parses input text in a single forward pass, building up a parse tree incrementally from left to right, accumulating a list of subtrees of the text that have already been parsed. Shift adds a single-node parse tree while reduce applies a completed grammar rule to recently parsed trees and joins them.

# Question 16

How does one deal with long-distance dependencies in statistical parsing? Give an example.

begin with smaller trees of various words in the sentence (that are the most probable according to statistical parsing), then put the smaller trees together to form the full tree. The heads of each non-terminal are also shown.





# Question 34

b. Perform the following feature structure unifications. If the feature structures do not unify, write “FAIL” (2 points).

b1.

CAT NP CAT NP

AGR NUM 3rd 𝖴 AGR GEND F PERS SG

b2.

CAT NP CAT VP

AGR NUM 1st 𝖴 AGR NUM 1st PERS PL ARI 2

c. Consider the following feature structures:

CAT VP CAT VP

NUM 3rd PERS SG

PERS SG TNS PLUP

Draw each feature structure as a directed acyclic graph. If the feature structures unify, draw the unification of the feature structures. If not, explain why they do not unify (4 points).

**Q3.** Consider the sentence *Fruit flies like a banana*.

1. This sentence is ambiguous. Paraphrase the two possible interpretations
2. Consider the following CCG parses of the sentence, which have some of their labels removed (indicated with [ ]). For each parse, fill in the missing labels: [1.], [2.], … [6.]
3. Note: a “fruit fly” is a tiny insect.

## Parse 1:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Fruit N |  | flies S\NP | like  **[ 1. ]** | a NP/N | banana N |
| **[ 2. ]** |  |  |  | NP | |
|  |  |  |  | **[3. ]** |  |
|  |  |  | S\NP | |  |
|  |  |  | S |  |  |

**Parse 2:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fruit N | flies N\N | like  **[ 4. ]** | a NP/N | banana N |
| N |  |  | NP | |
| **[ 5. ]** |  |  | **[ 6. ]** |  |
|  |  | S |  |  |

c. For each parse, indicate which of the two meanings the given parse corresponds to.

**Solution**

* 1. This sentence is ambiguous. Paraphrase the two possible interpretations.
     1. Fruit flies (insect) enjoy/have a preference for a banana.
     2. Fruit (food) flies (verb) in the way a banana flies.

# Parse 1:

1. ((S\NP)\(S\NP))/NP

2. NP

3. (S\NP)\(S\NP)

# Parse 2:

1. (S\NP)/NP
2. NP
3. S\NP

Parse 1 = meaning 2 in a

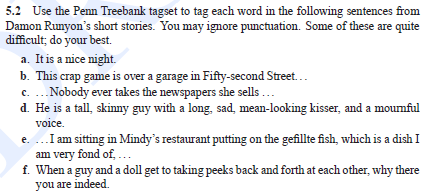
Parse 2 = meaning 1 in a

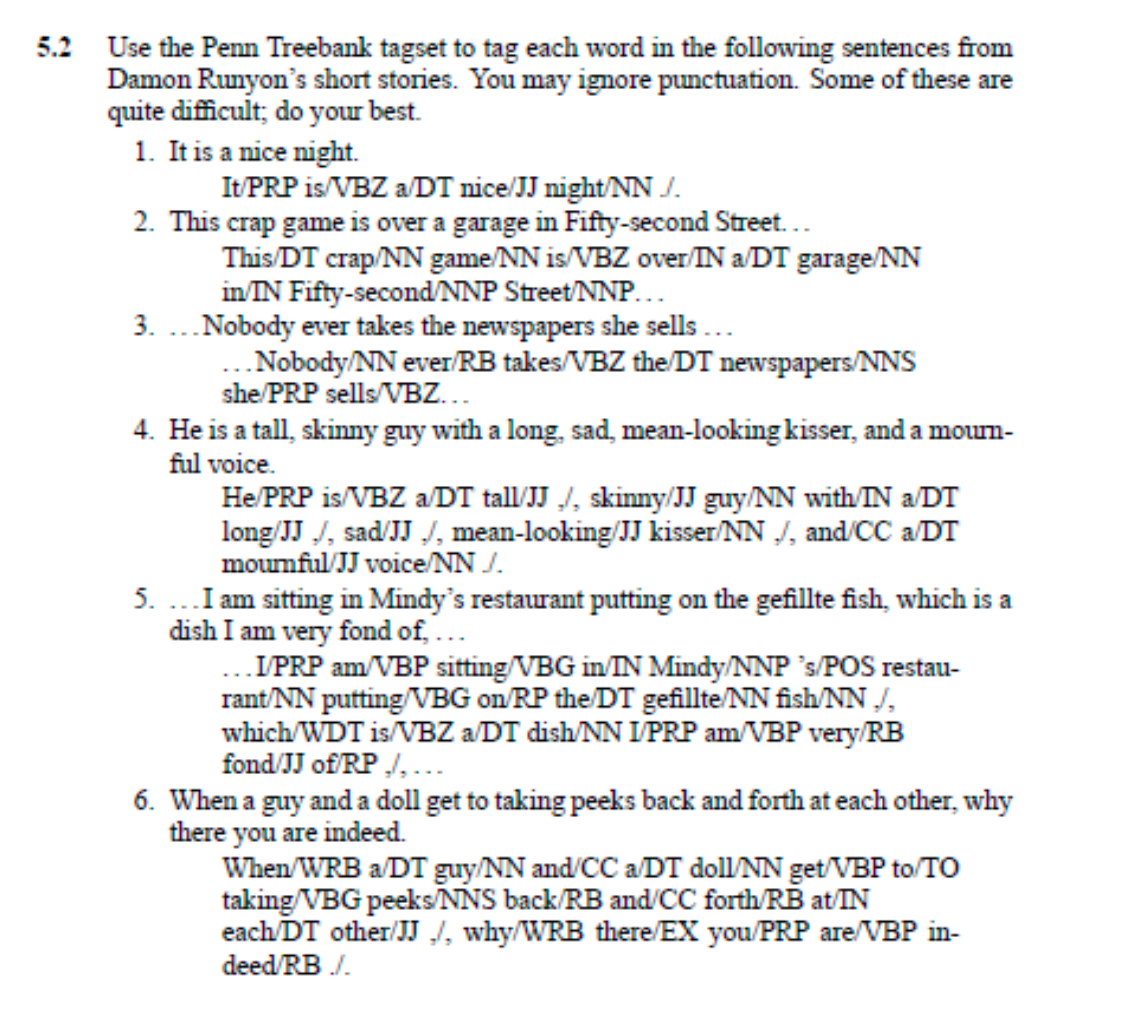
# Question 15

What is transformation-based part-of-speech tagging? What makes it different from the other major techniques in POS tagging?

A transformation-based POS tagger is a mixture of a statistical and a rule-based tagger. It assigns initial POS to each token based on the most frequent POS for that word (this part is statistical), and then applies rule-based transformations to these POS trying to correct possible tagging errors.

**Question 10**



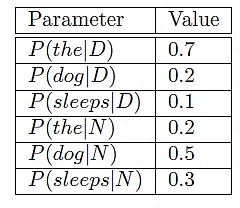


**Question 2:**

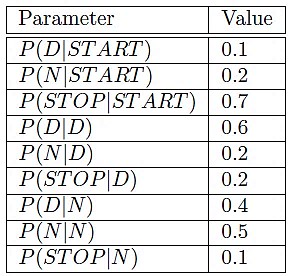
Consider the following HMM model for tagging. We will assume that the vocabulary consists of three words, *the, dog, sleeps*. There are two possible part-of-speech tags, *D, N*.

One set of parameters in the HMM are probabilities of the form P(*word*|*tag*), for example P(*the*|

*D*), P(*the*| *N*), etc. In our HMM these probabilities take the following values:



Another set of parameters in the HMM are of the form P(*tag*|*previous-tag*), for example P(*N*|*D*), P(*D*|*N*), etc. Note that we have a bigram HMM tagger, in that each tag depends only on the previous tag. We take START to be a special tag which always appears at the start of a sentence, and STOP to be a tag that appears at the end of the sentence. In our model these parameters take the following values:



We would like you to define a PCFG that is "equivalent" to the above HMM. By "equivalent" we mean the following:

* For any symbol sequence x and state sequence y which has probability PHMM(x, y) > 0 under the HMM: 1) there should be a parse tree y' such that the pair x, y’ gets probability PPCFG(x, y’) under the PCFG; 2) this probability should satisfy PHMM (x, y) = PPCFG(x, y’)
* There should be a one-to-one function f(y) = y' that maps a state sequence in the HMM to a parse tree generated by the PCFG, and that satisfies PHMM (x, y) = PPCFG(x, f(y))

In your solution you should: (a) write down your PCFG which is equivalent to the above HMM;

(b) define the function f(y) between state sequences and parse trees.

Note: you may find it useful to make use of productions in your PCFG where 𝜖 (the empty string) is on the right-hand-side of your rule. For example,

𝑋 → 𝜖 𝑤𝑖𝑡ℎ𝑃(𝑋 → 𝜖|𝑋) = 0.1

states that the non-terminal X can rewrite to the empty string with probability 0.1. To elaborate further, the context-free grammar

𝑆 → 𝑎𝑋

𝑋 → 𝜖

Generates one string, i.e., the string *a*.

PCFG:

|  |  |
| --- | --- |
| S  START D’ | 0.1 |
| S  START N’ | 0.2 |
| S  START STOP | 0.7 |
| D’  D D’ | 0.6 |
| D’  D N’ | 0.2 |
| D’  D STOP | 0.2 |
| N’  N D’ | 0.4 |
| N’  N N’ | 0.5 |
| N’  N STOP | 0.1 |
| D  the | 0.7 |
| D  dog | 0.2 |
| D  sleeps | 0.1 |
| N  the | 0.2 |
| N  dog | 0.5 |
| N  sleeps | 0.3 |
| START  \* | 1.0 |
| STOP  \* | 1.0 |

f(y):

Let s1, s2, …, sn be the state sequence where s\_i ∈{D, N} and w1, w2,…., wn be the corresponding symbol sequence. Then return the following tree:



**Question 3:**

Consider the following context-free grammar that recognizes simple sentences such as "the students teach the students":

Grammar:

S --> NP VP NP --> DET N NP --> N

VP --> V

VP --> V NP

Lexicon:

DET --> the N --> student N --> students V --> look

V --> looks V --> teach V --> teaches

**Part A:** While this grammar parses all grammatical sentences for the given lexicon, it also parses many ungrammatical sentences. For example, sentences with subject/verb disagreement, such as "the student teach", are parsed. In addition to sentences with subject/verb disagreement, identify two types of ungrammatical sentences that can be parsed with the given grammar and provide an example for each.

The following types of ungrammatical sentences parse:

* Type: sentences with incorrect definiteness / determiner usage
  + Example: “student teaches”
* Type: sentences with incorrect verb transitiveness
  + Example: “students look the student”

**Part B:** We would like to augment the basic grammar with unification constraints such that all grammatical sentences unify, producing valid parses, while ungrammatical sentences fail unification. In a paragraph, describe how the grammar could be augmented with unification constraints to address subject/verb disagreement and the additional types of ungrammatical sentences you identified above. Clearly indicate what features should be added to the lexical entries and what constraints should be added to the grammar rules.

In the lexicon, nouns can be augmented with number (num = sg/pl), and verbs can be augmented with usage (use = sg/pl) and transitiveness (trans = +/-).

Rules building a NP from a single N should disallow singular nouns (N.num = pl).

Rules combining a V and NP should require V to be transitive (V.trans = +).

Rules combining a NP and VP should require NP.N’s number to agree with VP.V’s usage.

# **Question 7**

# Consider the following (inelegant) grammar rules from the Penn Treebank:

# VP → VBD PP

# VP → VBD PP PP

# VP → VBD PP PP PP

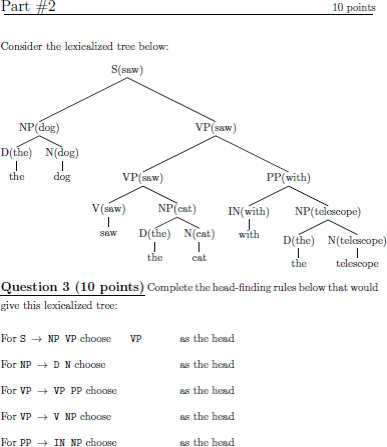
# VP → VBD PP PP PP PP

# Give two rules that can be used to replace these four (and all the ensuing... rules).

VP->VBD PP

VP -> VP PP

# Question 9



Complete the head-finding rules below that would give this lexicalized tree:

For S→ NP VP choose VP as the head

For NPD N choose N as the head

For VP→ VP PP choose VP as the head

For VP→ V NP choose V as the head

For PP→ IN NP choose IN the head

# Semantic parsing

# Question 55

Consider the sentence "Bill drives a Honda from Albany to New York." Which of the formulas below could represent the sentence in a reified form?

1. ∃w,x,y: Driving(Bill, w,x,y)
2. ∃z: Driving(Bill, Honda, Albany, NewYork)
3. ∃w,x,y,z: Driving(Bill, Honda, Albany, NewYork)
4. ∃w,x,y,z: Driving(w,x,y,z)
5. None of the above.

# Question 56

Represent the following sentence "No person is immortal." in First-Order Predicate Calculus (FOPC):

1. ¬∃x: Person(x) ˄ Mortal(x)
2. ¬∃x: ¬Mortal(x)
3. ∃x: Person(x) ˄ Mortal(x)
4. ¬∃x: Person(x) ˄ ¬Mortal(x)
5. ¬∃x: Mortal(x)

# Question 1 first-order logic

Represent the following sentences in first-order predicate calculus (FOPC). There may be multiple ways to represent each of them. Give only one representation for each sentence.

1. Only one person understood the play.

∃ x : Understood(x, thePlay) 𝖠 (∀ y: Understood(y, thePlay) → x=y)

1. Exactly two people understood the play.

∃ x,y : Understood(x, thePlay) 𝖠 Understood(y, thePlay) 𝖠 x ≠ y 𝖠 (∀ z: Understood(z, thePlay) → (x=z ∨ y=z) )

**Question 7**

**Give first-order logic translations for the following sentences:**

Vegetarians do not eat meat. Not all vegetarians eat eggs

∀ x: Vegetarian (X) => ¬Eats(x,Meat) ( there may be other solutions)

∃ x: Vegetarian (X) ^ ¬Eats(x,Eggs) (there may be other solutions)

S10. Translate into plain English the following FOL sentence:

For all x, if x is y’s mother, then x is y’s parent

# Question 2 verb subcategorization

Give an example for each of the following verb subcategorization phrases.

1. NP – run a race
2. NP NP – give my friend an apple
3. Ø – sleep
4. Pto – want to eat.
5. S – said he was fired.

M8. What is the semantic type of the sentence fragment "*walks(Janet)*"?

a. e

b. t → t

c. t

d. e → t

# Question 30

1. **explain the difference between syntactic and semantic parsing. Consider questions such as: How is semantic parsing done? In what situation might semantic parsing be more useful?**

Semantic parsing conveys different information than semantic parsing; often, this information is *in addition* to the syntactic parse, such as in compositional semantics. In compositional semantics, we first parse the sentence syntactically, then associate some semantic information/representation with each word, and finally follow a parsing algorithm, combining the semantics of words and non-terminals recursively until reaching the root of the sentence. Semantic parsing is most helpful in situations where the syntactic information is not enough. One such example is the case of selectional restrictions; the verb “eats” must take an edible object as an argument; this is not imposed by the syntactic structure (in which the direct object simply must be a noun phrase), but it is imposed by the semantic selectional restrictions.

1. **Please explain how can the semantic meaning of a sentence be represented formally. Consider an example sentence such as “All children like toys.” You may use words instead of symbols.**

Semantics is often expressed with first order logic - *for all* children: like(children, toys). This is an objective and effective way to structurally represent the semantics of a sentence.

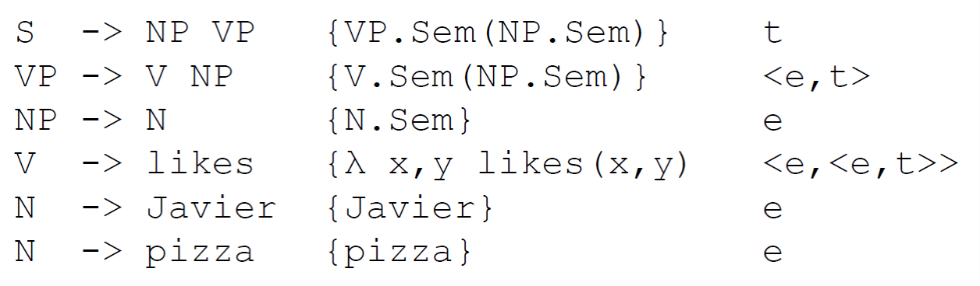
1. **Please describe the approach to semantic parsing using Abstract Meaning Representation (AMR).**

AMR uses a tree/graph structure that includes information about predicate-argument structure, named entity recognition, and coreference resolution. It is similar to (syntactic) dependency parsing, and makes use of the additional operations *swap*, *re-attach*, *replace head*, and *merge*.

1. **Please describe the approach to semantic parsing employing the SQL database language. Which do *you* think is more effective (AMR vs. SQL), and why?**

Sentences are expressed as SQL queries, and a seq2seq network is trained on such examples. It involves less information about syntactic structure.

**Q5.** Fill in the missing semantic types. Hint: one of the types is <e, t>



…e…

<e, t>…

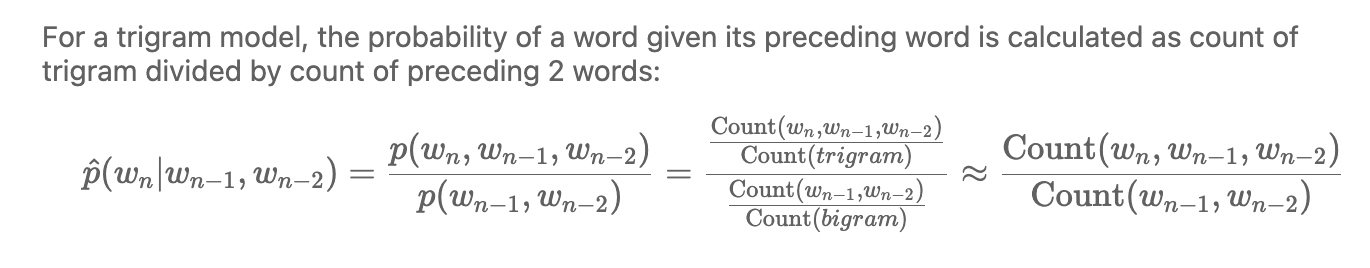
e…

<e. <e. t>>…

**n-gram Model**

# Question 4

What maximum likelihood formula is used to estimate trigram probabilities P(wn|wn-1,w n-2) using a corpus. Don’t worry about smoothing or backoff.



# Question 10

Consider the following Dr. Seuss rhyme:

*One fish two fish red fish blue fish black fish blue fish*

Show a table with the bigram counts for this corpus.

Given this table, give **P(fish|two)** and **P(black|fish).**

|  |  |
| --- | --- |
| bigram | count |
| one fish | 1 |
| fish two | 1 |
| two fish | 1 |
| fish red | 1 |
| red fish | 1 |
| fish blue | 2 |
| blue fish | 2 |
| fish black | 1 |
| black fish | 1 |
| total | 11 |

unigram corpus: one 1, two 1, red 1, blue 2, black 1, fish 6

P(fish|two)=Count(fish two)/Count(two)=1/1=1

P(black|fish)=Count(black fish)/Count(fish)=1/6

**Question 4:**

We covered language models in class. Please recall the n-gram models that were used in Homework 1.

1. We toss a fair coin a million times. estimate the unigram, bigram, and trigram probabilities for each of H and T (heads and tails), e.g., H, HT, TTH, etc.

Unigram probabilities: P(H) = P(T) = 0.5

Bigram probabilities: P(HH) = P(HT) = P(TH) = P(TT) = 0.25

Trigram probabilities:P(HHH) = P(HHT) = P(HTH) = P(THH) = P(HTT) = P(THT) = P(TTH) = P(TTT) = 0.125

1. Based on your estimation in (a), which n-gram model do you think works better in estimating the next outcome after the following outcomes: **T H H T H T H T T**?
2. unigram is better
3. bigram is better
4. trigram is better
5. None of the above

Is the answer the same or different compared to what you found in homework 1?

the n-grams are equally useless because the coin tosses are independent events.

The probability of each outcome (Heads or Tails) remains the same, regardless of the previous tosses.

This is different from homework 1. Because this is totally random, while language is not.

1. In Homework 1, how did you evaluate how good the model is (e.g., by what method or algorithm)?
2. MLE
3. Entropy
4. Word Error Rate
5. Perplexity
6. Precision and Recall
7. True or False: If in Homework 1 you were instructed to work with 50-grams, the result would be significantly better than 1-, 2-, 3-grams.

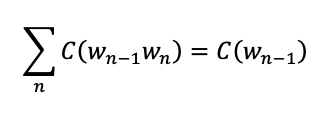
Because data is sparse in that case, and Markov assumption

1. To work with unseen data, we use smoothing. Using add-one (Laplace) smoothing, please calculate the smoothed unigram of “Pikachu” probability in the following corpus (It’s ok to leave it in fractions rather than in decimals):

Snorlax Charmander Bulbasaur Pikachu Pikachu Squirtle Snorlax Snorlax Charmander Pikachu

P(Pikachu)=3+1/10+5=4/15

1. Please choose the technique that can be used for dealing with sparse data.
2. Backoff
3. Interpolation
4. Regularization
5. Kneser-Ney
6. A and B
7. True or False



Where C is the count of certain words in corpus, and wn is the word appearing after wn-1.

**Hidden Markov Model**

# Question 8

Consider the HMM below, as defined in the following transition and emission tables:

# Transition Probabilities

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **to 0** | **to 1** | **to 2** | **to 3** | **Comment** |
| **from 0** |  | 0.4 | 0.6 |  | Start state |
| **from 1** |  | 0.4 | 0.5 | 0.1 |  |
| **from 2** |  | 0.6 | 0.2 | 0.2 |  |
| **from 3** |  |  |  |  | Final state |

For example, the probability to get from state 0 to state 2 is 0.6

# Emission Probabilities

|  |  |  |
| --- | --- | --- |
| **state** | **symbol** | **prob** |
| **1** | a | 0.7 |
| **1** | b | 0.3 |
| **2** | a | 0.2 |
| **2** | b | 0.8 |

For example, the probability of emitting "a" from state 1 is 0.7

**What is the probability of generating "baa"?**

Answer = 0.0128

The probability of a given path that generates “baa” is the product of its transition and emission probabilities. To find the probability of generating “baa”, we need to sum over all the possible paths that can generate “baa”.

We could use the Forward algorithm to do this, but since there are only 8 possible paths, it is maybe more straightforward to enumerate them and calculate their probabilities and sum the result:

|  |  |  |  |
| --- | --- | --- | --- |
| State sequence | Transition probs | Emission probs | p(“baa”) |
| 1 1 1: | .4 \* .4 \* .4 \* .1 | \* .3 \* .7 \* .7 | 0.0009408 |
| 1 1 2: | .4 \* .4 \* .5 \* .2 | \* .3 \* .7 \* .2 | 0.000672 |
| 1 2 1: | .4 \* .5 \* .6 \* .1 | \* .3 \* .2 \* .7 | 0.000504 |
| 2 1 1: | .6 \* .6 \* .4 \* .1 | \* .8 \* .7 \* .7 | 0.0056448 |
| 1 2 2: | .4 \* .5 \* .2 \* .2 | \* .3 \* .2 \* .2 | 0.000096 |
| 2 1 2: | .6 \* .6 \* .5 \* .2 | \* .8 \* .7 \* .2 | 0.004032 |
| 2 2 1: | .6 \* .2 \* .6 \* .1 | \* .8 \* .2 \* .7 | 0.0008064 |
| 2 2 2: | .6 \* .2 \* .2 \* .2 | \* .8 \* .2 \* .2 | 0.0001536 |
|  |  |  | **Total 0.0128** |

**What is the most likely state sequence associated with the output string "aba"?**

Answer = 0 1 2 1 3 (or just 1 2 1)

Similar to the process for part (a), we can enumerate the possible paths that generate “aba”, find their probabilties, and see which path has though enumerating 8 possible paths the highest probability. We could also use the Viterbi algorithm, is possibly faster and less complicated:

|  |  |  |  |
| --- | --- | --- | --- |
| State sequence | Transition probs | Emission probs | p(“aba”) |
| 1 1 1: | .4 \* .4 \* .4 \* .1 | \* .7 \* .3 \* .7 | 0.0009408 |
| 1 1 2: | .4 \* .4 \* .5 \* .2 | \* .7 \* .3 \* .2 | 0.000672 |
| **1 2 1:** | **.4 \* .5 \* .6 \* .1** | **\* .7 \* .8 \* .7** | **0.004704** |
| 2 1 1: | .6 \* .6 \* .4 \* .1 | \* .2 \* .3 \* .7 | 0.0006048 |
| 1 2 2: | .4 \* .5 \* .2 \* .2 | \* .7 \* .8 \* .2 | 0.000896 |
| 2 1 2: | .6 \* .6 \* .5 \* .2 | \* .2 \* .3 \* .2 | 0.000432 |
| 2 2 1: | .6 \* .2 \* .6 \* .1 | \* .2 \* .8 \* .7 | 0.0008064 |
| 2 2 2: | .6 \* .2 \* .2 \* .2 | \* .2 \* .8 \* .2 | 0.0001536 |

# Question 11

You are in a noisy bar diligently studying for your midterm, and your friend is trying to get your attention, using only a two words vocabulary. She has said a sentence but you couldn’t hear one of the words:

(w1 = hi; w2 = yo; w3 =???; w4 = yo)

Assume that your friend was generating words from this first-order Markov model:

p(hi|hi) = 0.7 p(yo|hi) = 0.3

p(hi|yo) = 0.5 p(yo|yo) = 0.5

**Given these parameters, what is the posterior probability of whether the missing word is “hi” or “yo”?**

This question is asking for p(w3|w1,w2,w4).

The **Markov assumption** states that the probability of a word at position t depends only on the word at position t-1, we can ignore w1 completely, thus just p(w3|w2,w4).

background-conditional **Bayes Rule**

p(a|bc)= p(b|ac)p(a|c)p(bc), which is like normal Bayes Rule except there's a "background" variable c always hanging on the right side.

p(w3|w2, w4) = p(w4|w3, w2)p(w3|w2)/p(w4|w2) = P(w4|w3)\*P(w3|w2)

because we use first-order Markov model, p(w4|w3, w2) simplifies to P(w4|w3) and we can ignore p(w4|w2)

where the prior is p(w3|w2 = yo) and the likelihood is p(w4 = yo|w3)

If the missing word w3 is "hi"

P(w3 = hi | w2 = yo, w4 = yo) = P(w3 = hi | w2 = yo) \* P(w4 = yo | w3 = hi) = 0.5 \* 0.3 = 0.15

If the missing word w3 is "yo",

P(w3 = yo | w2 = yo, w4 = yo) = P(w3 = yo | w2 = yo) \* P(w4 = yo | w3 = yo) = 0.5 \* 0.5 = 0.25

Normalize these probabilities so they add up to 1

P(w3 = hi | w2 = yo, w4 = yo) = 0.15 / (0.15 + 0.25) = 0.375

P(w3 = yo | w2 = yo, w4 = yo) = 0.25 / (0.15 + 0.25) = 0.625

0.625 > 0.375 missing word was more likely "yo".

# Question 15

Assume in the contexts of HMMs that *S* refer to a sequence of states, *O* refers to a sequence of observations, and *M* is a particular HMM model.

What is the algorithm that computes the value: argmax *P* (*S* | *O*, *M*)? **Viterbi**

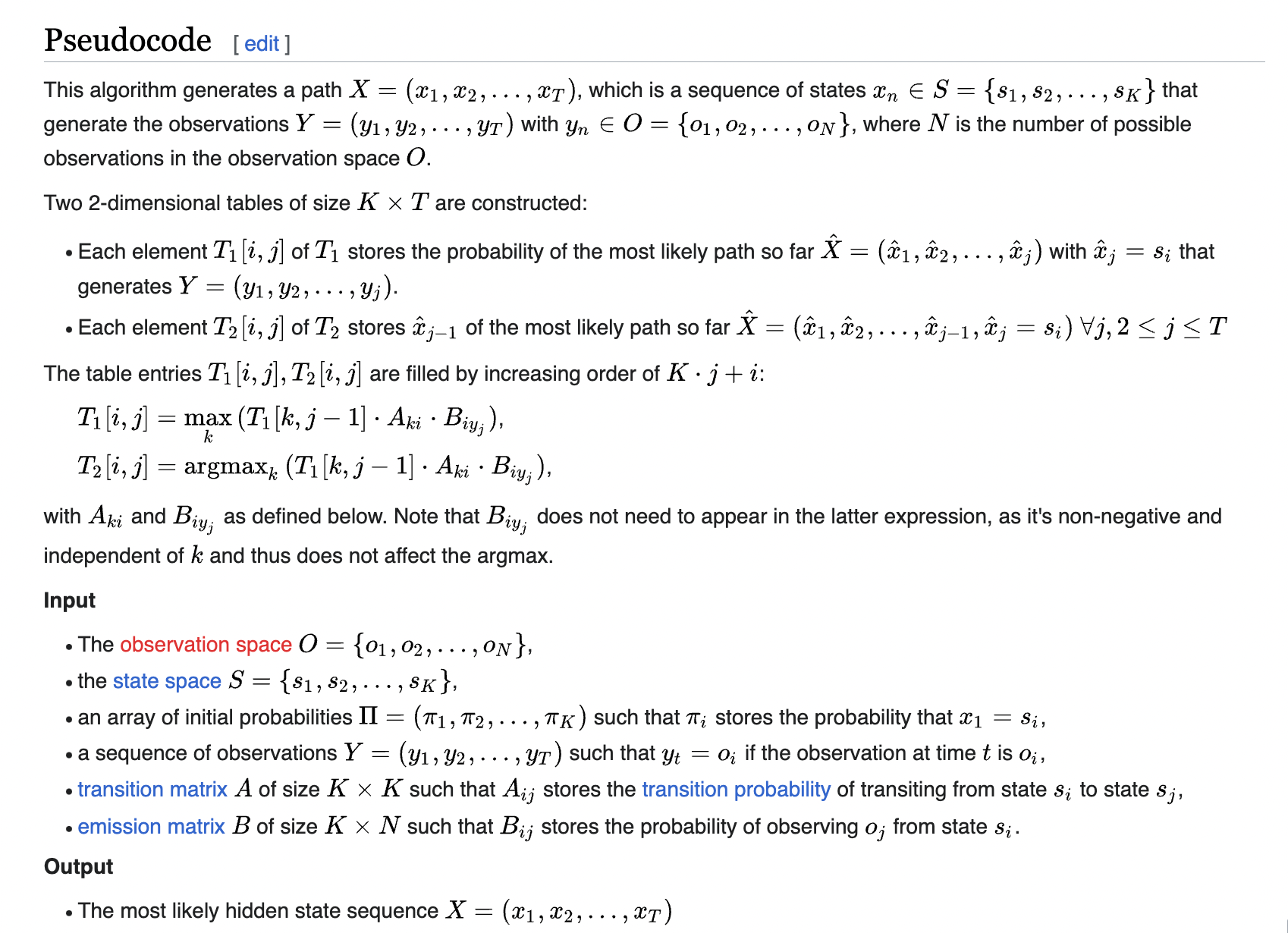
# Question 40

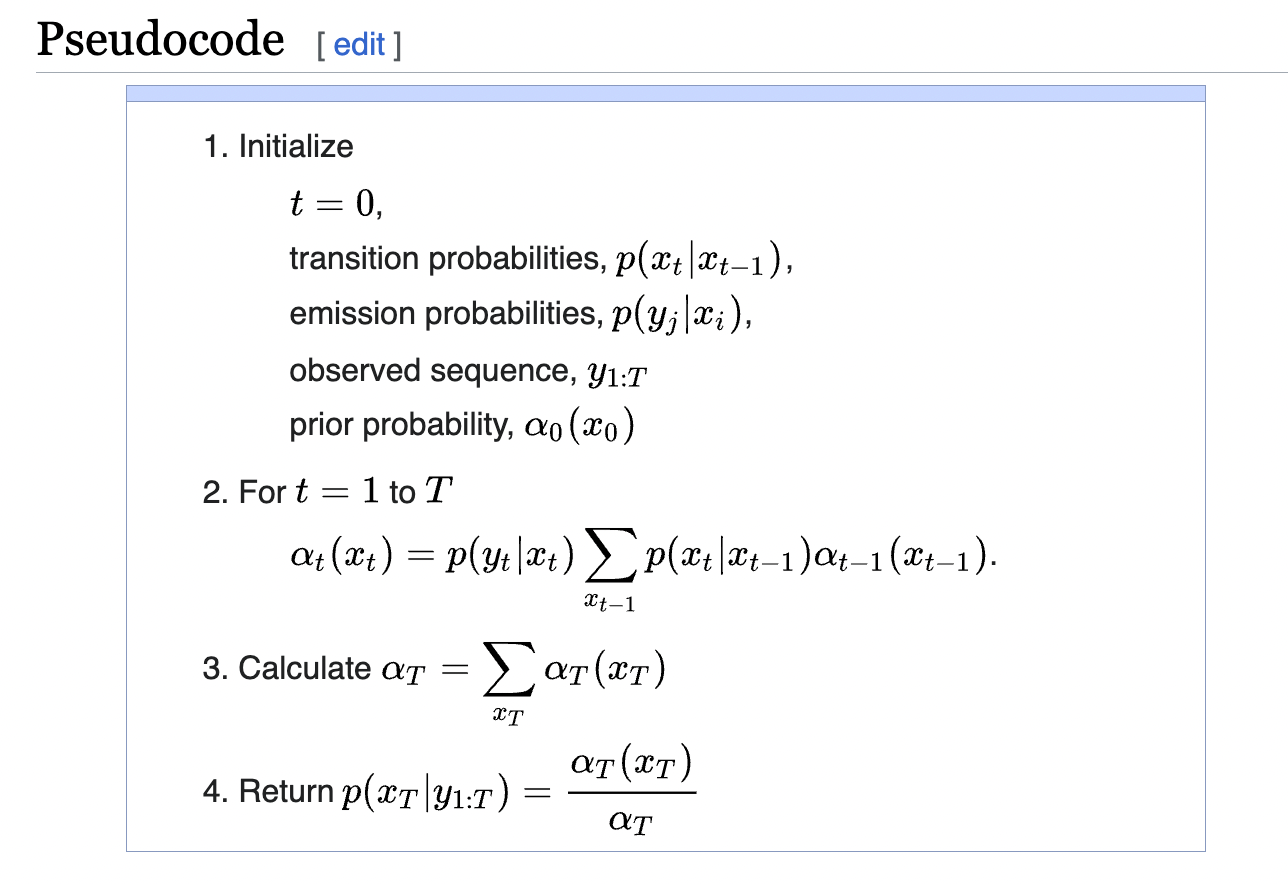
**What is the difference between the Viterbi algorithm and the Forward algorithm for POS tagging?**

The Viterbi algorithm is used when you are given a sequence of symbols and a model and want to find the most likely sequence of states that produced the sequence.

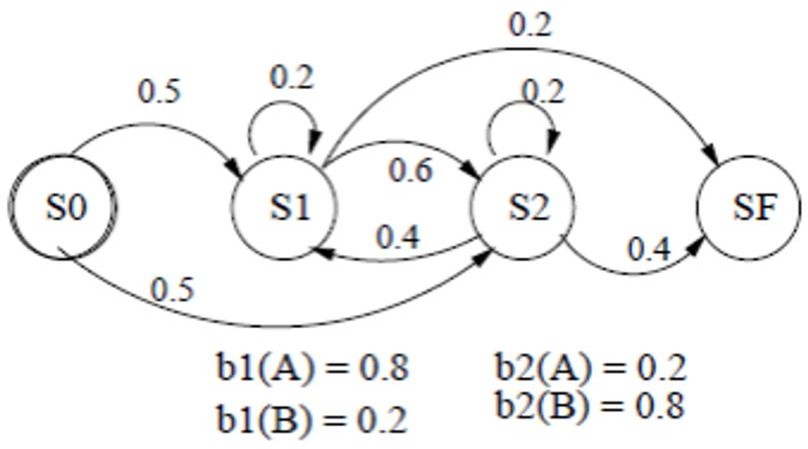
The forward algorithm is used when you are given a model structure and a set of sequences and want to find the model that best fits the data.

**Give pseudo-code for both**

[https://en.wikipedia.org/wiki/Viterbi\_algorithm#Pseudocode](https://en.wikipedia.org/wiki/Viterbi_algorithm%23Pseudocode) [https://en.wikipedia.org/wiki/Forward\_algorithm#Pseudocode](https://en.wikipedia.org/wiki/Forward_algorithm%23Pseudocode)



**Q2.** Consider the HMM below where the transition probabilities and the observation probabilities (where V = {A,B}) of each state are shown in the graph.



Use the forward algorithm (or some other method) to compute the probability of generating the short output string “A B”, starting at node S0.

α1(S1)=A01\*b1(A)=0.5\*0.8=0.4

α1(S2)=A02\*b2(A)=0.5\*0.2=0.1

α2(S1)=b1(B)(A11\*α1(S1)+A21\*α1(S2))=0.2\*(0.2\*0.4+0.4\*0.1)=0.024

α2(S2)=b2(B)(A12\*α1(S1)+A22\*α1(S2))=0.8\*(0.6\*0.4+0.2\*0.1)=0.208

α3(SF)=A1F\*α2(S1)+A2F\*α2(S2)=0.2\*0.024+0.4\*0.208=0.088

P=0.088

# noisy channel model

# Question 43 noisy channel model

n the following table we consider four candidates for when , which is the most likely correction? Why? dear

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **original word** | **proposed word** | **edit operation**  **|c** | **|c)** |  | P(w|c) \* P(c) |
| deea | deer | a|r | 0.0001 | 0.003 | 3e-7 |
| deea | idea | de|id | 0.0007 | 0.0009 | 6.3e-7 |
| deea | dear | ea|ar | 0.0003 | 0.007 | 21e-7 |
| deea | deea |  | 0.85 | 0.00000006 | 5.1e-7 |

In the Noisy Channel Model used for spelling correction, we are interested in finding the correction "c" that maximizes the probability of "c" given the observed word "w". P(c|w)

This can be expressed with Bayes' theorem as follows:

P(c|w) = P(w|c) \* P(c) / P(w)

P(c|w) is the posterior probability of the correction given the observed word.

P(w|c) is the likelihood of observing the word given the correction.

P(c) is the prior probability of the correction.

P(w) is the evidence, the probability of the observed word.

because P(w) is the same for all candidates, you can ignore it and just focus on maximizing P(w|c) \* P(c).

P(dear|deea)=21e-7 has max posterior probability

**Naive Bayes classifier**

## Q4.

|  |  |  |  |
| --- | --- | --- | --- |
| P(A|B) | | **A = “dog”** | |
| Yes | No |
| **B = NN** | Yes | 0.1 | 0.9 |
| No | 0.01 | 0.99 |

Suppose *p(NN)* = 0.4. Use the Bayes rule to write the expression for *p (NN| “dog”)*. Write it first as a function of *p(dog)* and then, also give the actual probability value.

p (NN| “dog”)=p(dog|NN)p(NN)/p(dog)=0.1\*0.4/p(dog)

# Question 18

1. **Give two reasons why a Naive Bayes classifier may be preferred over a neural network for some tasks.**
2. very fast, unlikely to overfit, and, if the features are independent, they can perform very well.
3. don't require any hyperparameter tuning.
4. 3) conceptually simple and we can prove things about their performance,
5. **What important statistical assumption does a Naive Bayes classifier make that isn't likely to be true of text?**

features are independent.

1. **Let's use Naive Bayes to predict whether or not a sandwich will be tasty given its ingredients. Assume we have two possible classes of tastiness, "good" and "bad". Given:**

P ("pickles" | "good") = 0.8

P ("mayo" | "bad") = 0.7

P ("ham" | "bad") = 0.2

P ("anchovies" | "good") = 0.1

P ("cheddar" | "good")= 0.8

P ("bad") = 0.4

**What, according to a Naive Bayes classifier, is the probability a sandwich containing ham, cheddar, and mayo is good?**

P("good" | "ham", "cheddar", "mayo")

= [P("ham"| "good") \* P("cheddar"| "good") \* P("mayo"| "good") \* P("good")] / P("ham", "cheddar", "mayo")

= (1-0.2)\*0.8\*(1-0.7)\*(1-0.4) / P(sandwich)

= .1152 / P(sandwich)

**What is the probability the sandwich is bad?**

P("bad" | "ham", "cheddar", "mayo")

= [P("ham"| "bad") \* P("cheddar"| "bad") \* P("mayo"| "bad") \* P("bad")] / P("ham", "cheddar", "mayo")

= 0.2\*(1-0.8)\*0.7\*0.4 / P(sandwich)

= .0112 / P(sandwich)

**In a Naive Bayes classification, the denominator is the same for all classes and can be ignored for the purpose of comparison.**

Since we only have two classes, we normalize by the sum of the probabilities (.1152 +.0112 )

P("good" | "ham", "cheddar", "mayo") =.1152/(.1152+.0112)=.9113

P("bad" | "ham", "cheddar", "mayo") =.0112/(.1152+.0112)=.0886

**What assumption might the Naive Bayes classifier be making that isn't true of combining ingredients in a sandwich to determine if the sandwich will taste good?**

The assumption made here is that the influence of each ingredient on the tastiness of the sandwich is independent of what other ingredients are present, which is not true.

# Question 35

**Applying Naive Bayes to word sense disambuguition**

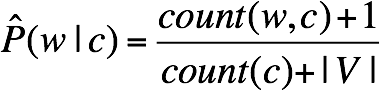
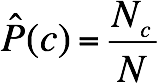
P(c) is the prior probability of a given sense of a word (counted in a labeled training set).

P(w|c) is the conditional probability of a word given a particular sense, such that P(w|c) = count(w, c) / count(c).

We can also generalize this paradigm to look at features besides words (i.e. P(f|c), the conditional probability of a feature given a sense).

# Consider the following data and vocabulary:

|  |  |  |  |
| --- | --- | --- | --- |
|  | DOC | WORDS w | CLASS c |
| TRAINING | 1 | elephant smoked elephant | e |
|  | 2 | elephant line | e |
|  | 3 | elephant smoked | e |
|  | 4 | flute jazz line | f |
| TEST | 5 | line flute jazz jazz | ? |

V = {elephant, smoked, line, flute, jazz}

Recall the following formulas: maximum likelihood estimation with add 1 smoothing

And assume the following prior probabilities P(C):

P(e) = 2/3

P(f) = 1/3

Part A: Calculate the following conditional probabilities:

P(line|e) = (1+1) / (7+5) = 2/12 =1/6

P(flute|e) = (0+1) / (7+5) = 1/12

P(jazz|e) = (0+1) / (7+5) = 1/12

P(line|f) = (1+1) / (3+5) = 2/8 = 1/4

P(flute|f) = (1+1) / (3+5) = 2/8 = 1/4

P(jazz|f) = (1+1) / (3+5) = 2/8 = 1/4

Part B: Comparing p(e|d5) and p(f|d5), choose the appropriate class.

P(C|D) = P(D|C) \* P(C) / P(D) -> P(C|D) = P(D|C) \* P(C)

* P(C|D) is the posterior probability of class C given document D
* P(D|C) is the likelihood of document D given class C
* P(C) is the prior probability of class C (e or f)
* P(D) is the evidence, the probability of the document (d1, ..., d5). often ignore the denominator P(D) when comparing classes, because it's the same for all classes.

P(D|C) = P(w1|C) \* P(w2|C) \* ... \* P(wn|C)

P(D|C) is lcalculated as the product of the conditional probabilities of each word given the class, assuming the words are independent (the "naive" assumption)

P(e|d5) =P(line|e) \* P(flute|e) \* P(jazz|e) \* P(jazz|e) \* P(e)

=2/3 \* 1/6 \* (1/12)2 \* 1/14 ≈ 0.00006

P(f|d5) =P(line|f) \* P(flute|f) \* P(jazz|f) \* P(jazz|f) \* P(f)

=1/3 \* 1/4 \* (1/4)2 \* 1/4 ≈ 0.001

# Question 37

A Naive Bayes classifier has to decide whether document number 5 ‘London Paris’ is news about the United Kingdom (class U) or news about Spain (class S). You can think of documents 1 through 4 as independent Bernoulli trials.

|  |  |  |
| --- | --- | --- |
| DOC | WORDS | CLASS c |
| 1 | London Paris | U |
| 2 | Madrid London | S |
| 3 | London Madrid | U |
| 4 | Madrid Paris | S |
| 5 | London Paris | ? |

1. Estimate the probabilities that are relevant for this decision from the following four documents. Answer with fractions.

P("London"|U) = count("London", U) / count(U) = 2/2 = 1

P("Paris"|U) = count("Paris", U) / count(U) = 1/2 = 0.5

P("London"|S) = count("London", S) / count(S) = 1/2 = 0.5

P("Paris"|S) = count("Paris", S) / count(S) = 1/2 = 0.5

1. Based on the estimated probabilities, which class does the classifier predict? Assume a uniform prior distribution over the classes U and S.

The classifier chooses the class that maximizes the posterior probability.

P(C|D) = P(D|C) \* P(C) ≈ P(D|C)

since we're assuming a uniform prior distribution over classes, P(C) is the same for both classes (0.5), and so it can be ignored when comparing classes. Therefore, we only need to calculate P(D|C), which is the product of the conditional probabilities of each word in the document given the class (assuming the words are independent):

P(D5|U) = P("London"|U) \* P("Paris"|U) = 1 \* 0.5 = 0.5

P(D5|S) = P("London"|S) \* P("Paris"|S) = 0.5 \* 0.5 = 0.25

Given that P(D|U) > P(D|S), the classifier predicts that document 5 is news about the United Kingdom (class U).

1. Practical implementations of a Naïve Bayes classifier often use log probabilities. Explain why.

**Avoiding underflow:** When dealing with very small probabilities, multiplying them together can lead to underflow (i.e., the result is too small to be represented accurately by the computer's floating-point arithmetic). Taking the log transforms the product of probabilities into a sum of log-probabilities, avoiding this issue.

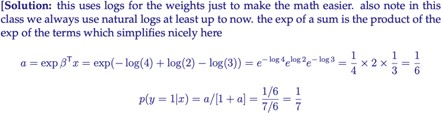
**Addition Efficiency**: Addition (used when summing log probabilities) is computationally cheaper than multiplication (used when multiplying probabilities). This can lead to substantial efficiency gains when dealing with large amounts of data.

**Logistic regression**

# Question 16

Consider a logistic regression model used to classify documents into “sports” vs. “not sports”. The weights β = (−ln(4), ln(2), −ln(3)). A given document D is represented as the feature vector x = (1, 1, 1).

What is the probability that document D is about sports? 1/7



**Neural Network**

# Question 39 universal function approximation

Which one of the following can be considered as a "universal function approximator"?

1. a two-layer neural network
2. a hidden markov model
3. a push-down automaton
4. a finite-state automaton

Universal Approximation Theorem [Cybenko, 1989]:

A neural network with **a single hidden layer containing a sufficient but finite number of neurons** and proper non-linear activation function can approximate any continuous function to a reasonable accuracy on compact subsets of its domain, given an appropriate choice of weights and biases.

# Question 21 activation

**What is the purpose of an activation function in a neural network?**

non-linearity: The activation function introduces non-linearity, so that the network is more powerful in that it is capable of representing any function. That is, neural networks are Universal Function Approximators.

**What would happen if we didn’t have one?**

Without an activation function, a neural network would be like a linear regression model, which has limited capabilities.

# Question 13 sigmoid

What is the formula for the sigmoid function f(z)? f(z) = 1/(1+e^(-z))

Compute the derivative of the logistic function: f’(z)=f(z)(1-f(z))

Why is it commonly used in Neural Networks?

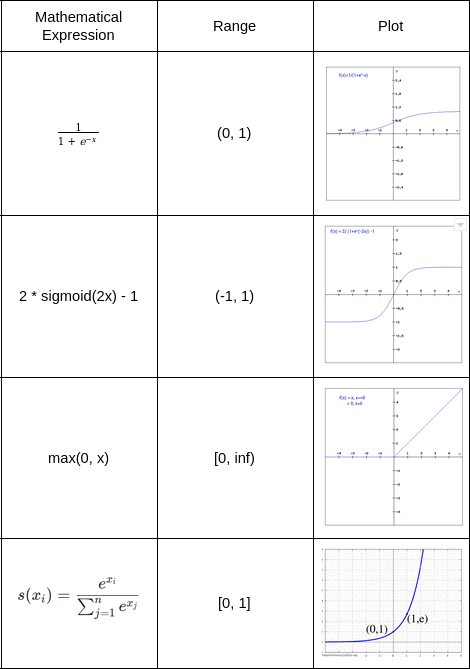
**It is used as a non-linear activation function. It is differentiable, monotonic, and has values between 0 and 1, which makes it useful for computing probabilities.**

Which of the following is true of sigmoid?

1. Sigmoid overcomes the vanishing gradient problem.
2. Optimization is easier in sigmoid than in tanh.
3. The resultant outputs are in the range [0,1].
4. Sigmoid is preferred to ReLU.

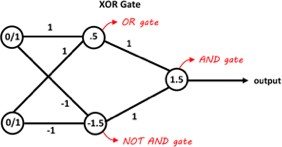
# Question 23

What are the names of the four functions below? sigmoid, tanh, ReLU, softmax



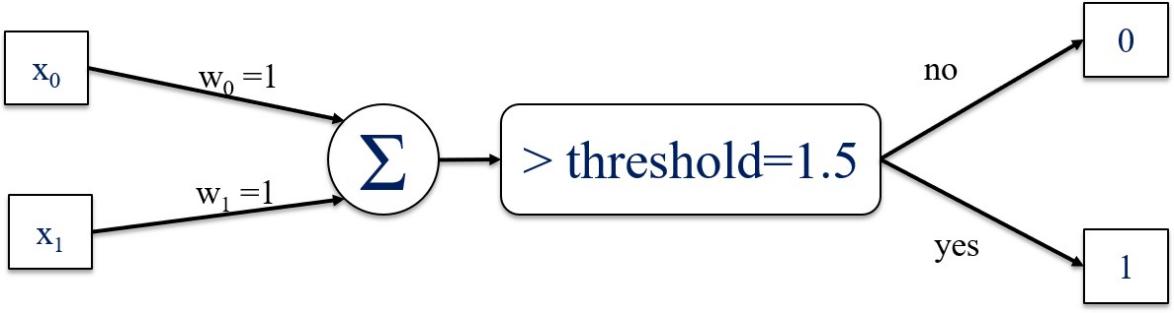
# Question 24 XOR

Draw a neural network with a hidden layer that can compute X XOR Y for the Boolean variables X and Y. Show all biases and other weights and explain why your network works. Note that there are many possible answers.

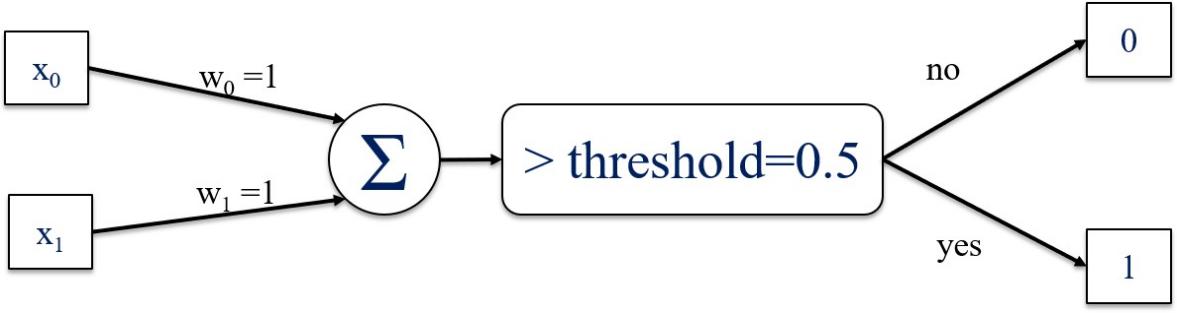


# Question 20 perceptron

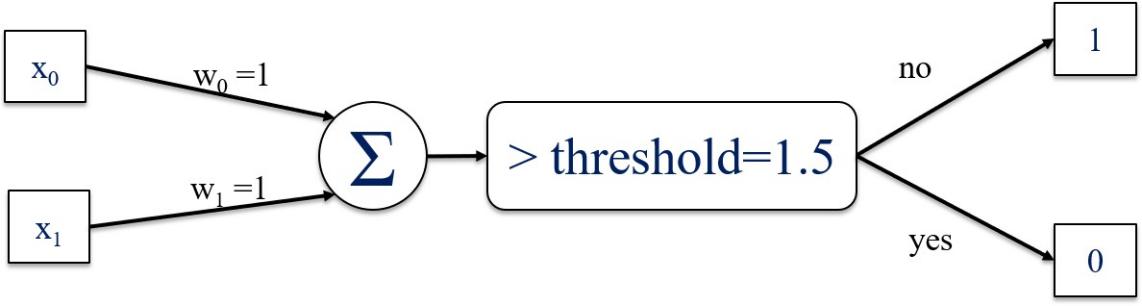
(1) Consider the following perceptrons: (a)

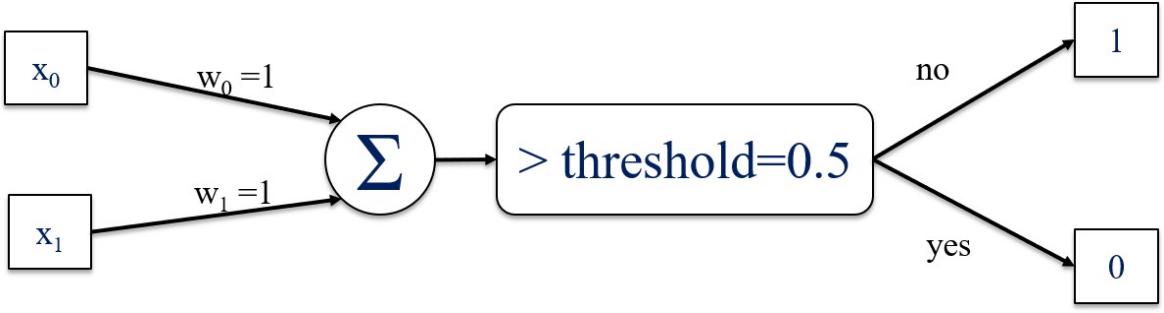


(b)



(c)





* 1. 5 points. Please match the perceptrons above with the logical gates below. One or more of the gates may not match any of the perceptrons, and you can use (e) in this case.

a AND gate

/ XOR gate

b OR gate

c NAND gate (not and)

d NOR gate (not or)

* 1. If we take the threshold node out, the perceptrons will be equivalent to which one of the following:

A. Linear regression

B. Logistic regression

* 1. Which of the following methods may help with preventing overfitting

A. Use a larger dataset

B. Increase the number of hidden layers

C. Penalize the weights which is to use the regularized loss function

* + 1. Normalize input data
    2. Train the neural net with batches of examples instead of one at a time
  1. Which of the following methods may help with training the neural net faster

A. Penalize the weights which is to use the regularized loss function

B. Normalize input data

C. Increase the learning rate as large as possible

# Question 46 gradient descent

What is the update formula for a perceptron used to compute the value of w at time i+1 from the value of w at time i.?

w(i+1) = w(i) … …

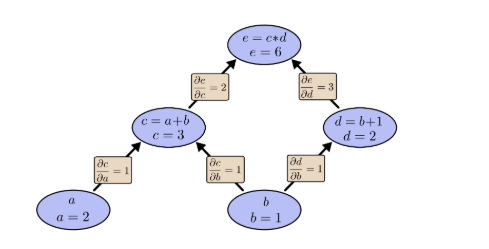
w(i + 1) = w(i) + y(i)x(i)

**y(i) is the true class label at time i**

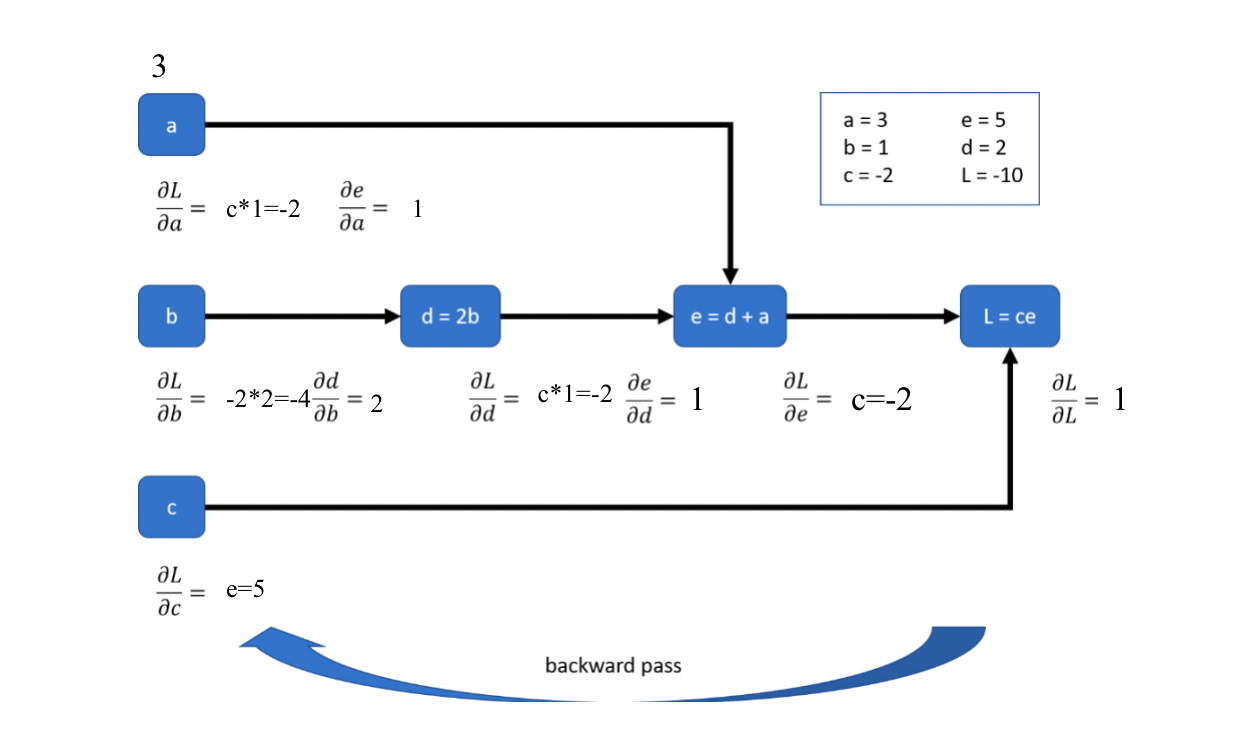
**x(i) is the input vector at time i**

**没懂**

For the given computational graph below, calculate the following partial derivatives:



**Q5.** Compute all the missing partial derivatives.using chain rule



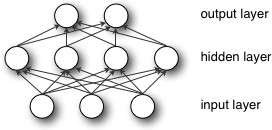
# Question 23

Consider an autoencoder with three layers: input layer (X neurons), hidden layer (Y neurons), and output layer (Z neurons).

What is the relationship between the number of neurons in each layer?

1. X < Y < Z
2. X = Y = Z
3. X < Y, Y > Z
4. X = Z, Y < X
5. X > Y > Z

**Q1.** We use a 3-layer neural network for sentiment classification of words. The architecture of the neural network is shown in the picture. Words are represented by 3-dimensional embeddings as inputs, and the network outputs a probability distribution over the positive and negative classes as a 2-dimensional vector.



The hidden layer and the output layer use Rectified Linear Units (ReLU) as the activation function.



Suppose your parameters are





1. Suppose the input word has an embedding of:



Calculate the activation of the hidden layer.[0, 9, 3, 0]

1. Following (1), what is the probability distribution after we apply softmax over the output layer activations?

[1/(1+e^3), e^3/(1+e^3)]

We use cross-entropy loss as our objective function defined as:



where N is the batch size, and we label positive words with y = 1, and negative words with y

= 0.

1. Now suppose the current mini-batch contains four words: {good, bad, excellent, poor}. The network outputs a probability distribution as the following:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | good | bad | excellent | poor |
| P (y=1) | 0.7 | 0.3 | 0.6 | 0.2 |

What is the value of the objective function for the current mini-batch?

-1/4 (1\*log0.7+1\*log(1-0.3)+1\*log0.6+1\*log(1-0.2))=0.3618

**RNN, LSTM, GRU**

# Question 31

RNNs are known for capable of dealing with variable-length input and producing variable-length output, making them suitable for a variety of NLP tasks using sequential text data.

**How do RNNs remember past information?**

use of hidden states. At each time step, the RNN takes in the current input as well as the hidden state from the previous time step (which contains information from the past inputs) to produce the output and update hidden state.

**Vanilla RNNs, also known as Simple Recurrent Networks (SRNs), are known to have "short term memory", "vanishing gradient problem", and "exploding gradient problem".**

1. **Explain what these three terms mean and why vanilla RNN network lead to these issues.**

**Short term memory**: RNNs cannot remember long-term dependencies. This is because the hidden states near the end of the sequence disproportionately encode more of the information at the end of the sequence and forget about information near the start of the sequence.

**Vanishing gradient** **problem:** gradient becomes extremely small and close to zero, which is caused by the unrolling of the network during backpropagation through time, where each derivative is < 1, and multiplying them leads to a small number.

**For exploding gradient:** caused by a similar problem as that of vanishing gradient, except multiplying a whole lot of derivatives > 1.

1. **impact of these issues on the network's performance, including referring to at least one specific NLP task**

**short term memory:** network can't capture long-term dependencies. For example, in predicting the next word of a sentence, the network may not correctly identify the subject of a verb and thus give the right verb form, if there's a long separation between the subject and the verb, e.g. due to a long relative clause.

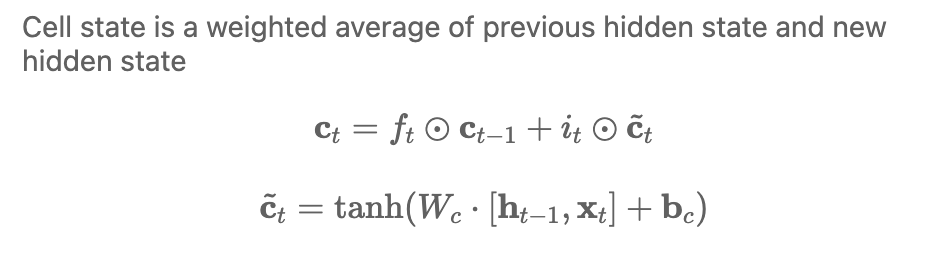
**vanishing gradient:** network would stop learning. For example, in machine translation, while training on a very long sequence, the network would stop updating its weights.

**exploding gradient:** network has very unstable performance, with large swings in its weight values.

1. **Explain how LSTMs solve these three issues. Specifically, describe what is special about the LSTM architecture and how it improves SRNs.**

LSTM solve these issues by **memory cell and gates**

**memory cell:** a unit store information at each time step

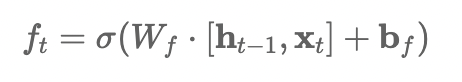


**gates:** regulate the cell state through gates, control the flow of info thru the network

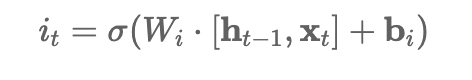
* architecture: a sigmoid neural net layer and a pointwise multiplication operation.
  + The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through.
  + A value of zero means “let nothing through,”
  + a value of one means “let everything through”.

3 kinds of gates

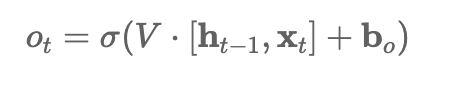
* **forget gate:** decides how much previous information from previous cell state should be added to current cell state. protects the current step from irrelevant information from previous steps

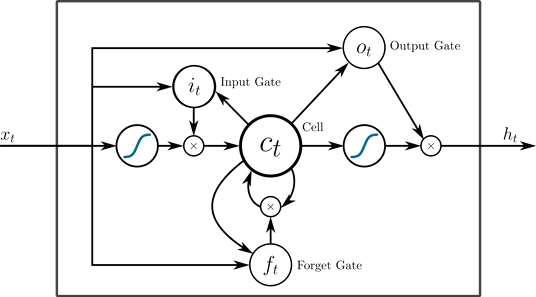


* **input gate:** decides how much new info from current input should be added to current cell state. protects the current step from irrelevant inputs



* **output gate:** determines what the next hidden state should be by controlling info from current cell state. protects the current step from passing irrelevant outputs





1. **GRU (Gated Recurrent Unit) is a variant of LSTM. Describe how it is different.**

GRU is different because it combines the forget and input gates of LSTM into a single "update gate". It also merges the cell state and hidden state.

1. **Bidirectionality is often added to RNNs such as LSTM, GRU, and SRN. Explain: 1) how it is implemented in practice**

Bidirectionality is implemented by concatenating two hidden states at each timestep, one hidden state generated by scanning data left to right, and the second by scanning right to left.

**2) what is its advantage, including reference to at least one NLP task**

encodes contextual information both to the left and to the right of the current time step, and more context is helpful for creating a better representation of a token, during tasks such as language modeling or part of speech tagging.

1. **RNNs are often used as the encoder and decoder of a sequence to sequence (seq2seq). Could bidirectionality also be used for the encoder and the decoder? If yes, given an example of a specific NLP task that takes advantage of this. If not, explain why.**

Bidirectionality can be applied to the encoder but not the decoder.

For the encoder, in tasks such as machine translation, helps encoding contextual information both to the left and to the right of an input token at each timestep, helping to create a more contextualized and meaningful **final context vecto**r for feeding into the decoder.

However, the decoder can't use bidirectionality since it can't know what is coming before it decodes the hidden state (i.e. right to left scanning is impossible).

# Question 33

**Describe the difference between recurrent neural networks and recursive neural networks. In particular, which can be thought of as a generalization of the other?**

Recursive neural networks can be thought of as generalisations of recurrent NNs.

**Suppose you start running into the “vanishing gradient problem” with a vanilla recurrent neural network. What RNN variants should you try to fix the problem?**

You’d use LSTMs or GRUs to try and fix the vanishing gradient problem.

**Suppose you want to write a neural network that predicts the sentiment of sentences on the basis of the dependency parse of the sentence. Would you want to use a recurrent neural network or a recursive neural network? Justify your choice.**

You’d want to use a recursive neural network.

**Suppose you have some time-series data of stock prices, and want to use it to predict the future movement of stock prices. Would you want to use a recurrent neural network or a recursive neural network? Justify your choice.**

You’d want to use a recurrent neural network.

**Describe briefly how backpropagation works for a recursive neural network in terms of standard backpropagation (i.e. for a multilayer perceptron).**

**Backpropagation Through Structure (BPTS):** Backpropagation in RecNN works similarly to backpropagation in standard NN, main difference is errors are propagated through tree-like structure rather than layered structure.

- \*\*forward pass\*\*: input data is passed from the bottom of the tree (the leaves) to the top (the root), with each parent node's representation being calculated based on its children nodes.

- error computation: same. difference between the network's output and the true output

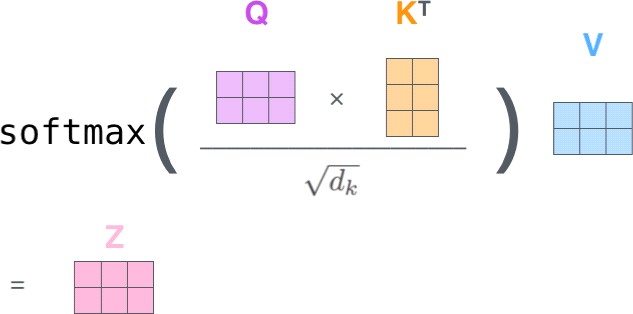
- \*\*backward pass\*\*: error is propagated from the top of the tree (the root) to the bottom (the leaves). The error at each node is used to calculate the gradients of the parameters that were used to compute that node's representation.

- Parameter Update: same

**Transformer**

# Question 29

The formula below is used in what type of neural network:



1. Bidirectional LSTM
2. Transformer
3. Stacked LSTM
4. Pointer network
5. Convolutional neural network

**BERT**

# Question 19

**What is BERT?**

BERT is Transformer encoder for language modelling.

BERT (Bidirectional Encoder Representations from Transformers) is an open-sourced NLP pre- training model developed by researchers at Google in 2018. A direct descendant to GPT (Generalized Language Models), BERT has outperformed several models in NLP and provided top results in Question Answering (SQuAD v1.1), Natural Language Inference (MNLI), and other frameworks.

**What is the key innovation?**

reads the entire sequence of words at once. Therefore bidirectional, though more accurate to say non-directional. allows the model to learn the context of a word based on all of its surroundings (left and right of the word).

This is in contrast to previous directional models which looked at a text sequence either from left to right or combined left-to-right and right-to-left training, which read the text input sequentially (left-to-right or right-to- left).

**give two examples of using BERT for NLP tasks**

* 1. **Classification:** such as sentiment analysis are done similarly to Next Sentence classification, by adding a classification layer on top of the Transformer output for the [CLS] token.
  2. **Question Answering** tasks (e.g. SQuAD v1.1), the software receives a question regarding a text sequence and is required to mark the answer in the sequence. Using BERT, a Q&A model can be trained by learning two extra vectors that mark the beginning and the end of the answer.
  3. **Named Entity Recognition** (NER), the software receives a text sequence and is required to mark the various types of entities (Person, Organization, Date, etc) that appear in the text. Using BERT, a NER model can be trained by feeding the output vector of each token into a classification layer that predicts the NER label.

# Question 28

BERT is trained to predict 15% of the tokens. What does it do with these 15%? Pick all correct answers (zero, one, or more):

1. replaces the token with a mask
2. replaces the token with a random token c. keeps the token unchanged

d. duplicates the token

e. writes the token backwards

**BERT and other pre-trained language models are extremely large and expensive, how are companies applying them to low-latency production services?**

**Model Distillation**: This involves training a smaller, faster model to mimic the behavior of the larger model. The smaller model is trained to reproduce the output distribution of the larger model, which often results in comparable performance with significantly reduced complexity.

**Quantization**: This technique reduces the precision of the weights in the neural network. By using lower precision (like 16-bit floats instead of 32-bit), the model size can be reduced significantly, leading to faster inference times.

**Pruning**: This involves removing less important weights or neurons from the model. The pruned model retains most of the performance of the original model but is smaller and faster.

**Model Parallelism and Pipelining**: These techniques involve splitting the model across multiple GPUs or machines, allowing different parts of the model to run in parallel.

**On-Device Machine Learning**: Some companies use on-device machine learning to run these models directly on user devices (like mobile phones), reducing the need for network communication and thus reducing latency.

**What’s GLUE?**

The General Language Understanding Evaluation benchmark (GLUE) is a collection of 9 datasets used for comparing the performance of different models that are capable of multiple tasks.

CoLA (Corpus of Linguistic Acceptability): A binary single-sentence classification task, where the goal is to predict whether an English sentence is linguistically acceptable or not.

SST-2 (Stanford Sentiment Treebank): A binary single-sentence classification task for sentiment analysis.

MRPC (Microsoft Research Paraphrase Corpus): A corpus of sentence pairs labeled as semantic equivalents (paraphrase) or not.

STS-B (Semantic Textual Similarity Benchmark): a collection of sentence pairs, task is to predict a score of how similar two sentences are in meaning.

QQP (Quora Question Pairs): A binary classification task, goal is to determine if two questions asked on Quora are semantically equivalent.

MNLI (Multi-Genre Natural Language Inference): A three-way (entailment, contradiction, neutral) classification task where the goal is to predict the relationship of a premise sentence to a hypothesis sentence.

QNLI (Question Natural Language Inference): A binary classification version of the SQuAD dataset, where the goal is to determine if the context sentence contains the answer to the posed question.

RTE (Recognizing Textual Entailment): A binary classification task similar to MNLI, but with less training data.

WNLI (Winograd NLI): A binary natural language inference task based on reading comprehension of Winograd schema sentences.

**Text similarity**

# Question 60 edit distance

What is the Levenshtein edit distance between "apples" and "pears"? Assume the following costs: insertion=1, deletion=1, substitution=1.

edit distance=4

1. p-p-l-e-s

p-e-a-r-s

for apples

delete ‘a’ -> pples

substitute second p to e -> peles

substitute l to a -> peaes

substitute third e to r

**Q1.** Using a chart, compute the edit distance between APPLE and PIE. Assume equal costs for the three basic edit operations.

delete operation: A, P

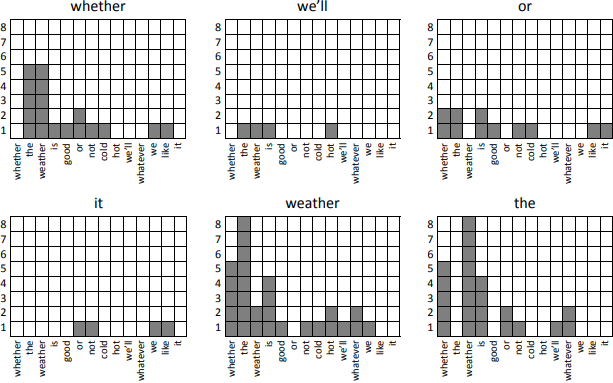
substitution operation: L->I

costs=3

**Q1.** There are many ways to represent a word’s meaning in terms of its distribution. For this question, we will build count vectors for words using the following poem:

*Whether the weather is good, or whether the weather is not, Whether the weather is cold, or whether the weather is hot, We’ll weather the weather—whatever the weather— Whether we like it or not.*

The representations of some words from this poem are shown below as obtained by counting how often each other word occurs in a certain window around the word in question.



1. Write down a vector representation for the word *is* in the same scheme as the graphs above. For example, *whether* would be encoded as <0, 5, 5, 1, 1, 2, 1, 1, 0, 0, 0, 1, 1, 0>.

[1, 4, 4, 0, 1, 2, 1, 1, 1, 1, 0, 0, 0, 0, 0 ]

Below are 33 word count vectors. These were obtained from a different sample text and show the counts of 15 words (word A through word O), but the identities of these 15 words are not given. Your task: Study these 33 word graphs and then answer the questions that follow.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Word** | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** | **J** | **K** | **L** | **M** | **N** | **O** |
| *queen* | 1 | 0 | 2 | 0 | 8 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 3 | 0 | 0 |
| *neigh* | 0 | 8 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| *man* | 9 | 0 | 1 | 0 | 2 | 0 | 9 | 0 | 3 | 0 | 0 | 0 | 7 | 3 | 0 |
| *kings* | 6 | 0 | 3 | 0 | 1 | 0 | 0 | 0 | 0 | 5 | 0 | 4 | 0 | 5 | 0 |
| *woman* | 3 | 0 | 4 | 0 | 10 | 0 | 9 | 0 | 3 | 0 | 0 | 0 | 10 | 0 | 0 |
| *horse* | 0 | 2 | 0 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 1 |
| *queens* | 0 | 0 | 5 | 0 | 8 | 0 | 0 | 0 | 0 | 6 | 0 | 3 | 3 | 2 | 0 |
| *uncle* | 9 | 0 | 0 | 1 | 0 | 0 | 0 | 2 | 0 | 0 | 4 | 0 | 1 | 4 | 0 |
| *king* | 7 | 0 | 0 | 0 | 1 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 3 | 0 |
| *ugliest* | 0 | 0 | 0 | 1 | 0 | 0 | 6 | 0 | 0 | 3 | 1 | 0 | 0 | 0 | 2 |
| X1 | 1 | 3 | 3 | 0 | 0 | 4 | 0 | 0 | 0 | 5 | 10 | 4 | 0 | 2 | 0 |
| X2 | 3 | 10 | 9 | 8 | 9 | 9 | 7 | 10 | 7 | 10 | 10 | 10 | 9 | 9 | 8 |
| X3 | 1 | 9 | 0 | 0 | 0 | 3 | 0 | 6 | 0 | 0 | 6 | 0 | 1 | 0 | 0 |
| X4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| X5 | 2 | 1 | 2 | 0 | 8 | 0 | 0 | 1 | 0 | 0 | 5 | 0 | 4 | 2 | 0 |

1. The 11 mystery words (X1-X5) have the following definitions (but not in this order):
   1. *antismartnessesquely X4*
   2. *aunt X5*
   3. *cats X1*
   4. *meow X3*
   5. *the X2*

Match each definition above to its mystery word. Hint: remember the algebraic properties of word embeddings. That is, *queens* - *queen* = *kings* - *king*. These kinds of analogies can be applied in sequence to build approximate equations for words.

queen-king=aunt-uncle

aunt=queen-king+uncle

**Q6.** Given the word embeddings (Vw) for the words w in {Paris, London, France, English, French, Euro, UK, etc.}, what is the value of (VLondon – VParis + VEuro)?

Vpound

# Question 36 term-document frequency

Use the following documents for this problem, where the frequency of word appearing (and not just the word’s presence) in a document matters.

D1 = “cat, dog, fox”

D2 = “fish, tiger, cat”

D3 = “cat, fox, dog”

D4 = “fish, cat, fish”

transform the documents into frequency vectors using the table below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | fish | cat | fox | tiger | dog |
| D1 | 0 | 1 | 1 | 0 | 1 |
| D2 | 1 | 1 | 0 | 1 | 0 |
| D3 | 0 | 1 | 1 | 0 | 1 |
| D4 | 2 | 1 | 0 | 0 | 0 |

**What is the Jaccard Similarity between D1 and D2?**

Jaccard Similarity = #common attributes/ # total attributes = 1/5

**What is the Euclidean Distance between D3 and D4?**

d=||[0, 1, 1, 0, 1]-[2, 1, 0, 0, 0]||=||[-2, 0, 1, 0, 1]||=\sqrt(4+1+1)=\sqrt 6

**Suppose we decide to use Cosine Similarity. Which Document is most similar to D2? D4**

Cosine\_Similarity(A, B) = dot\_product(A, B) / (||A|| \* ||B||)

**What are the ranges (in general, not for this particular problem) of the above similarity and distance functions: Jaccard Similarity, Euclidean Distance and Cosine Similarity? You may assume that all document vectors only consist of non-negative components.**

Jaccard: [0, 1]

Cosine: [0, 1] (for vectors with non-negative values)

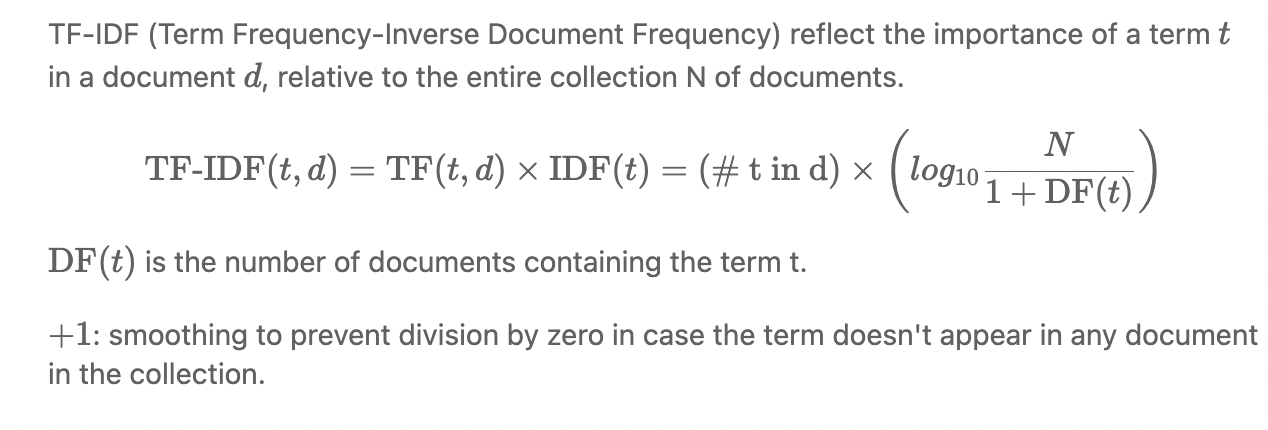
Euclidean: [0, Infinity)

# Question 52 TF-IDF

There are 10,000 documents in a collection. The number of times each of these terms occur in documents 1 to 4 as well as the number of documents in the collections are listed below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Term Frequencies | | | | |
| Terms | Documents | | | |
| Doc 1 | Doc 2 | Doc 3 | Doc 4 |
| reverse cascade | 8 | 10 | 0 | 0 |
| full shower | 3 | 1 | 2 | 2 |
| half bath | 0 | 0 | 8 | 7 |
| multiplex | 2 | 2 | 2 | 9 |

Calculate the TF\*IDF for the terms listed below for documents 1 to 4.



First calculate the IDF of the mentioned terms;

|  |  |  |
| --- | --- | --- |
| Inverse document frequency | | |
| Term t | DF(t) | IDF(t) |
| reverse cascade | 3 | log(10000/3+1)=8 |
| full shower | 50 | log(10000/50+1)=5 |
| half bath | 10 | log(10000/10+1)=7 |
| multiplex | 3 | log(10000/3+1)=8 |

then fill the following table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TF-IDF for terms in documents | | | | |
| Terms | Documents | | | |
| Doc 1 | Doc 2 | Doc 3 | Doc 4 |
| reverse cascade | 8\*8=64 | 10\*8=80 | 0 | 0 |
| full shower | 3\*5=15 | 1\*5=5 | 2\*5=10 | 2\*5=10 |
| half bath | 0 | 0 | 8\*7=56 | 7\*7=49 |
| multiplex | 2\*8=16 | 2\*8=16 | 2\*8=16 | 9\*8=72 |

# Question 14

We first need to introduce a function that compares similarity between two sentences. Recall the definition for the cosine distance between two vectors x and y:

**Part A:** How would you define a mapping from sentences s to vectors f(s) such that the cosine distance cosine (f (s1), f (s2)) is a measure of the similarity between two sentences s1 and s2?

**f(s) = a vector that contains a feature for each word w that counts the number of times w is seen in s**

**Part B:** Now, we will define a dynamic programming algorithm that computes sentence alignment of two translations. The alignment score is the sum of the similarity scores of the aligned sentences. Our goal is to find an alignment with the highest score.

We will consider alignments of the following form:

* a sentence can be aligned to an empty sentence. This happens when one of the translators omits a sentence.
* a sentence can be aligned to exactly one sentence.
* a sentence can be aligned to two sentences. This happens when one of the translators either breaks or joins sentences.

Our sentence alignment algorithm recursively computes the alignment matrix F indexed by i and j, one index for each sentence. The value stored in F(i, j) is the score of the best alignment between the first i sentences of x and the first j sentences of y. s(xi, yj) is the similarity between sentence xi and sentence yj.

* Define F(0, 0), F(i, 0) and F(0, j).
* Define F(i, j) for i > 0 and j > 0.

F(0,0) = 0

F(i,0) = 0

F(0,j) = 0

F(i,j) = max[ F(i-1, j-1) + s(i,j),

F(i-1, j),

F(i,j-1),

F(i-2,j-1) + s(i-1,j) + s(i,j),

F(i-1,j-2) + s(i,j-1) + s(i,j)]

**Part C**: Next, we'll modify the alignment score to be the same as before, but to include a fixed penalty p each time a sentence is aligned to an empty sentence. p is a parameter, which is >= 0, chosen by the user of the algorithm. Describe a modified dynamic programming method which takes this new penalty into account.

F(0,0) = 0

F(i,0) = -ip

F(0,j) = -jp

F(i,j) = max[ F(i-1,j-1)+s(i,j),

F(i-1,j)-p,

F(i,j-1)-p,

F(i-2,j-1)+s(i-1,j)+s(i,j),

F(i-1,j-2)+s(i,j-1)+s(i,j)

# Question 25

**What is the idea behind “retrofitting embeddings”, a method introduced by Manaal Faruqui?**

improve non-contextual word embeddings (word2vec or GloVe) using semantic lexicons (WordNet, FrameNet, or other domain-specific lexicons/ontologies) to better capture semantic information of words.

e.g., in WordNet, word ’bank’ have multiple senses, river or financial institution. Retrofitting ensure they have different embeddings.

Algorithm

1. start with pre-trained word vectors obtained from a method like word2vec or GloVe.
2. For each word in your lexicon, look at its neighboring words (those words that it has a semantic relationship with).
3. update the word's vector to bring it closer to the vectors of its neighbors. This is done by minimizing a cost function, which is defined over the original vector for the word and the vectors for its neighbors.
4. The process is repeated iteratively until the vectors reach a state of convergence