Homework 2

Problem 1

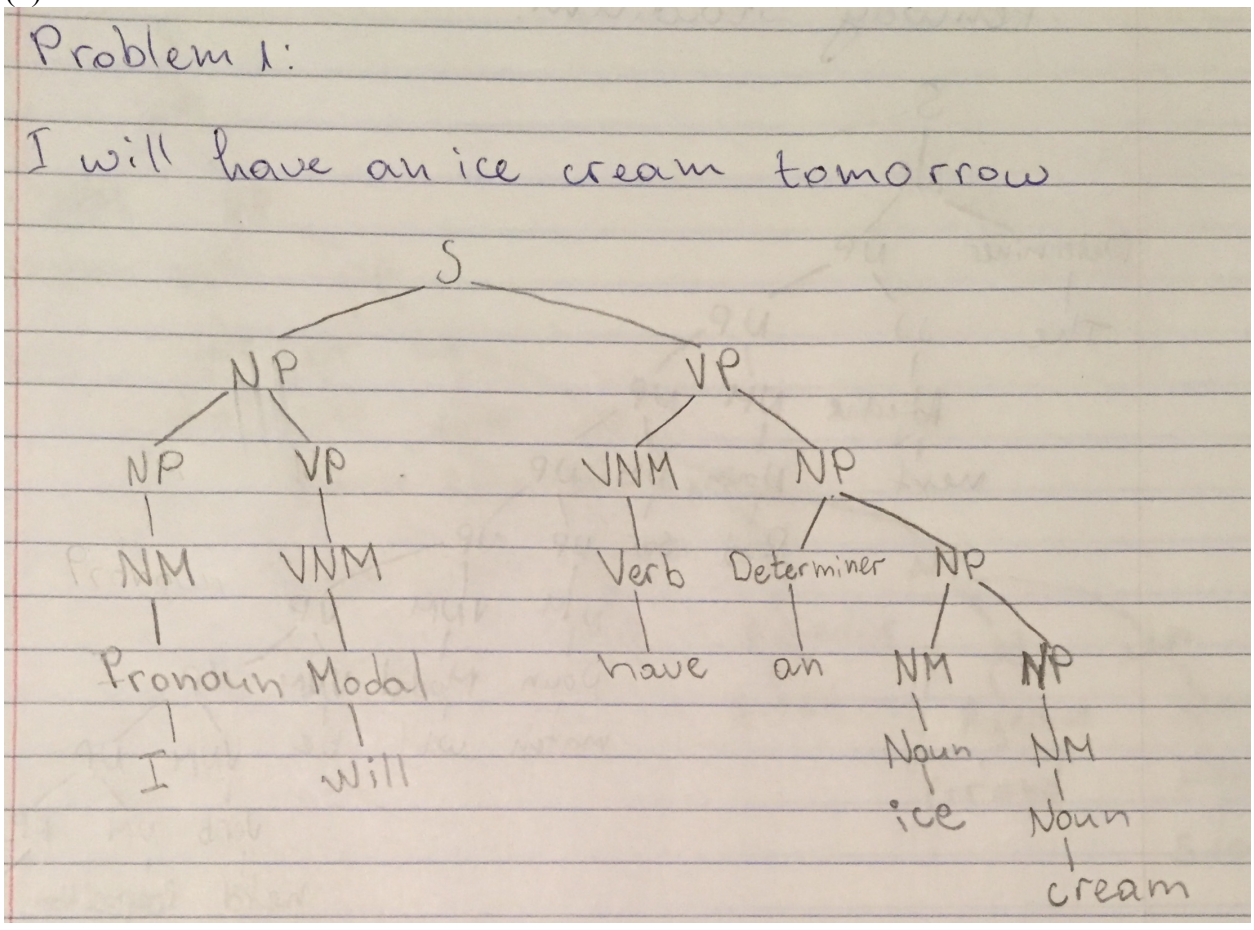
Grammar:

S -> NP VP
S -> NP
S -> VP
NP -> NM NP
NP -> NP VP
NP -> NM Punctuation
NP -> JJ NP
NP -> OR NP
NP -> Determiner NP
NP -> NM
NM -> Noun
NM -> Pronoun
NM -> Preposition
VP -> VNM VP
VP -> VNM NP
VP -> VNM
VNM -> Modal
VNM -> Verb
VNM -> Question
JJ -> Adjective NP
JJ -> Adjective
OR -> Adverb
OR -> Preposition

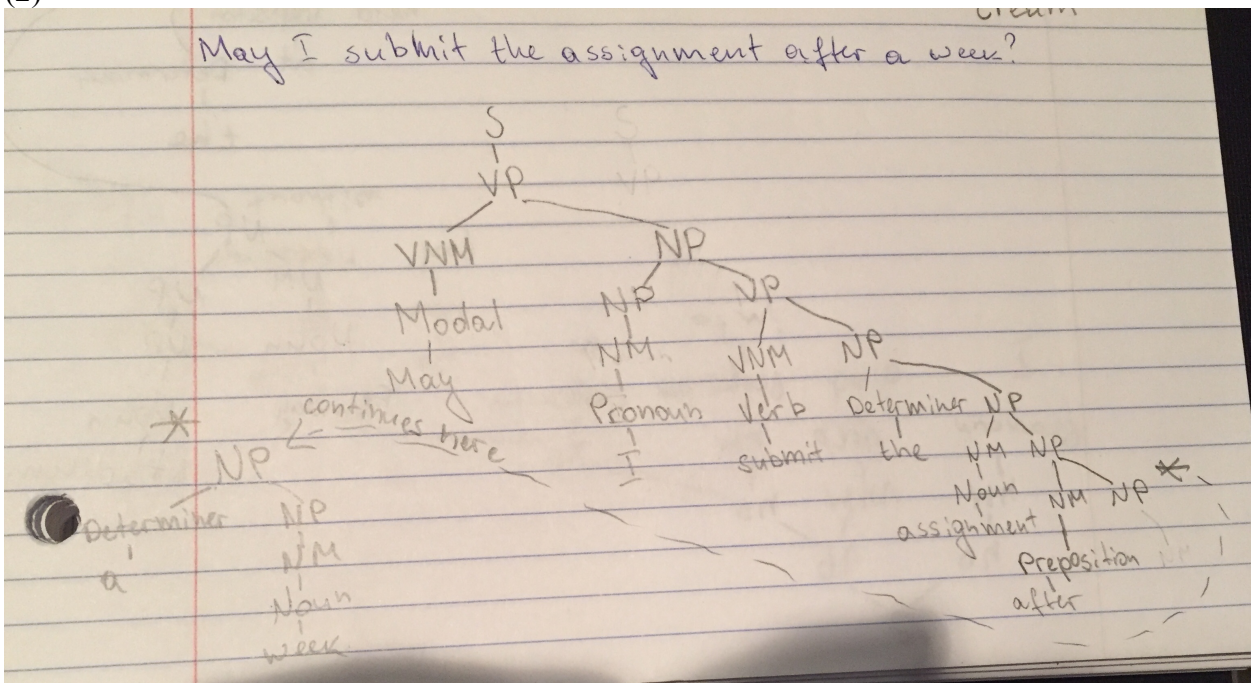
Pronoun -> I
Modal -> will
Verb -> have
Determiner -> an
Noun -> ice
Noun -> cream
Noun -> tomorrow
Modal -> may
Verb -> submit
Determiner -> the
Noun -> assignment
Preposition -> after
Determiner -> a
Noun -> week
Adjective -> next
Noun -> Red
Noun -> Sox

Noun -> match
Modal -> will
Verb -> be
Verb -> held
Preposition -> at
Noun -> Fenway
Noun -> Stadium
Question -> when
Verb -> did
Noun -> Federer
Verb -> win
Pronoun -> his
Adjective -> first
Adjective -> Grand
Noun -> Slam
Verb -> dived
Preposition-> into
Noun -> pool
Preposition -> without
Noun - lifejackets

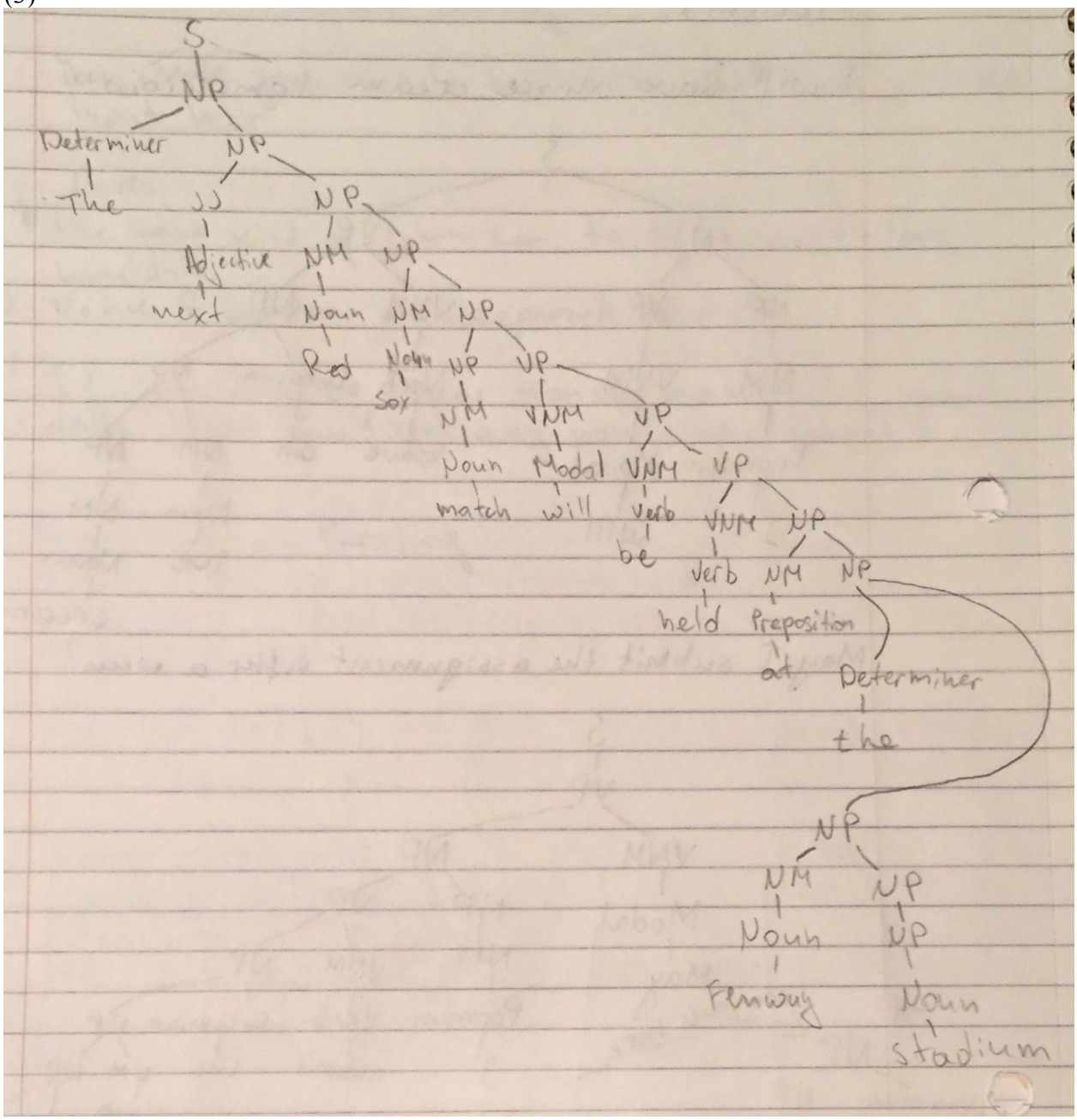
(1)



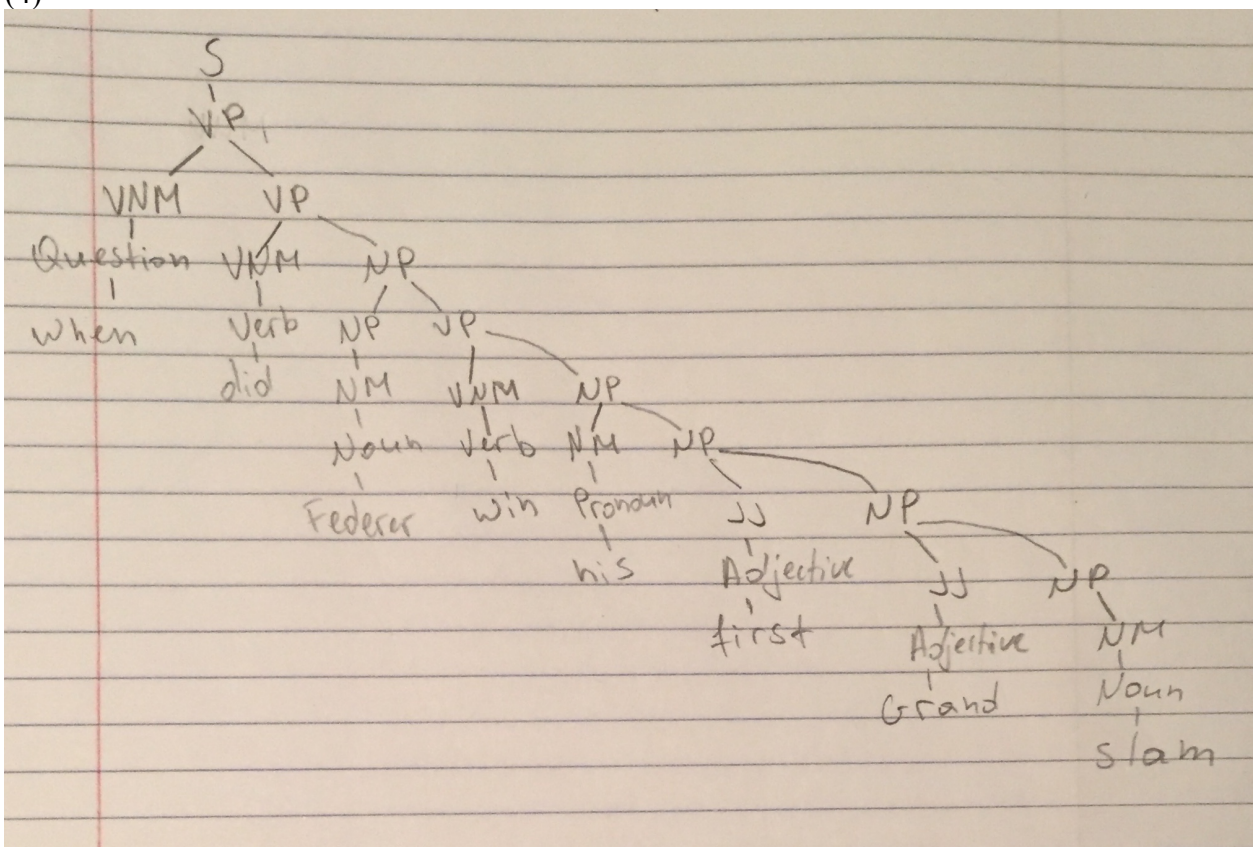
(2)


May I submit the assignment after a week?
S
VP
VNM
Modal
May
NP
NP
NM
Pronoun
I
VP
VNM
Verb
submit
NP
Determiner
the
NP
NM
Noun
assignment
NM
Preposition
after
NP
continues here
NP
Determiner
a
NP
NM
Noun
week

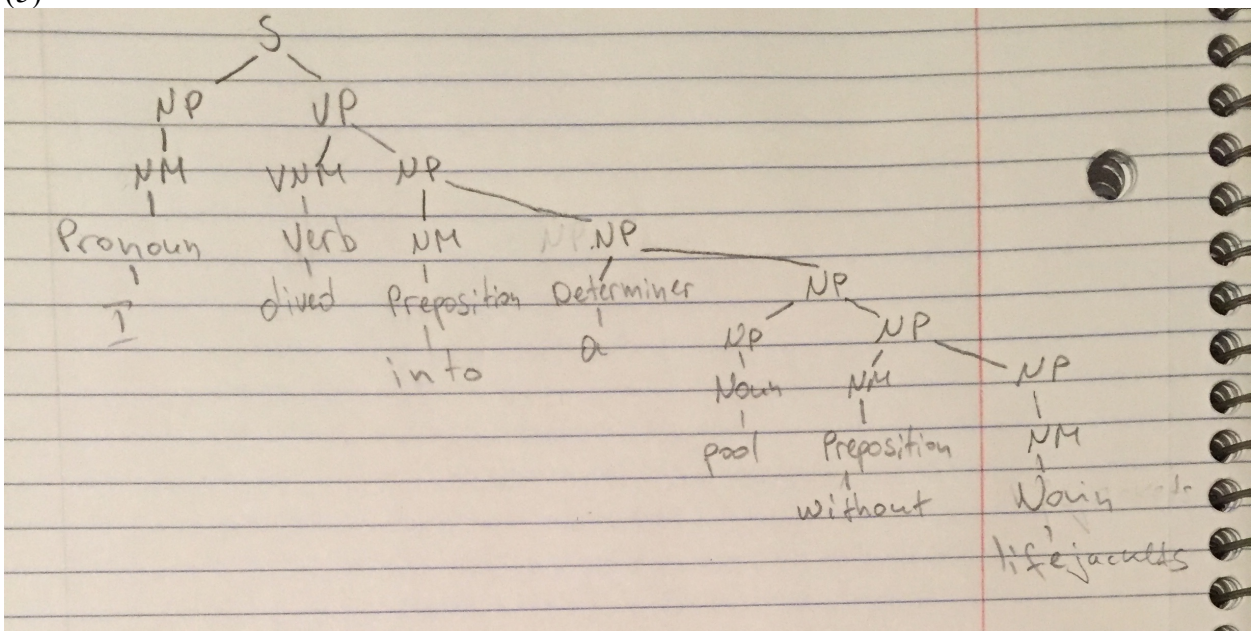
(3)


S
NP
Determiner
NP
The
JJ
NP
Adjective
NM
NP
next
Noun
NM
NP
Red
Noun
NP
VP
Sox
NM
VNM
VP
Noun
Modal
VNM
VP
match
will
Verb
VNM
NP
be
Verb
NM
NP
held
Preposition
Determiner
at
the
NP
NM
NP
Noun
NP
Fenway
Noun
stadium

(4)


S
VP
VNM
VP
Question
VNM
NP
When
Verb
NP
VP
did
NM
VNM
NP
Noun
Verb
NM
NP
Federer
win
Pronoun
JJ
NP
his
Adjective
JJ
NP
first
Adjective
NM
Grand
Noun
slam

(5)


S
NP
VP
NM
VNM
NP
Pronoun
Verb
NM
NP
I
dived
Preposition
Determiner
NP
into
a
NP
NP
Noun
NP
pool
NM
Preposition
without
NM
Noun
lifejackets

Problem 2

The parsing of a sentence could be augmented to follow the steps below.

1. Generate a Probabilistic Context-Free Grammar (PCFG).
2. Start generating the parse tree from bottom up
3. Read the first/next word in a sentence
4. If the read word does not fall in the lexicon, look at the word or words before it. Find in the PCFG what type of terminal is the unknown/misspelled word most likely to be given the word/s before it. The PCFG rule can tell us for example that after a N, it is most likely that we see a V. So if we had "Tom wooorks", we could estimate that "wooorks" is most likely to be a V. If we do not have any data about what might follow the word on the left, we will always assume that the misspelled word is an N.
5. If the first word in a sentence is unknown.
    a. Go right in the sentence until you find a word that is correctly spelled
    b. From there go to the left word by word basing your predictions for the misspelled words on the words to right of it. If we do not have any data about what might follow the word on the right, we will always assume that the misspelled word is an N.
    c. Once you get back to the first word, go to step 2
6. Assign the respective terminal to the word
7. If there are more words in the sentence, go back to step 2

The problem with misspelled words is that it becomes impossible for the word to be parsed, whatever grammar we have. However, my algorithm described above takes an educated guess what part of speech (and terminal) a word could be based on the word before it. The algorithm would work because whatever terminal we have seen before the misspelled work X, there should be some probability of what might follow it and we will always take the most likely terminal. If for some reason there is no data about what might follow the known word, we will assume that the misspelled word is a noun.

If the first word in the sentence was misspelled, however, we would not be able to guess what the word might be because there is nothing before it. What we do in this case is keep looking at words after the first word until we find one word that is correctly spelled and is in the lexicon. Then, similarly to above, we guess what the word on the *left* might be. We keep guessing the left terminal based on the right terminal until we have guessed what would be the terminal for the first word. Once we have solved this problem, we start parsing the sentence again from the first word and we now parse it normally guessing the terminal for any misspelled word based on the word before it.

Problem 3

"the train has a bunker"

the	train	has	a	bunker
Det -> the .50 NP -> Det N .15 S -> NP VP .12	NP -> Det N .003 S -> NP VP .0024	S -> NP VP .0000144	S -> NP VP .000000864	S -> NP VP .00000000864
	N -> train .02 NP -> Det N .006 S -> NP VP .0048	S -> NP VP .0000288	S -> NP VP .000001728	S -> NP VP .00000001728
		V -> has .03 VP -> V NP .006 S -> NP VP .0048	VP -> V NP .00036 S -> NP VP .000288	VP -> V NP .0000036
			Det -> a .20 NP -> Det N .06 S -> NP VP .048	NP -> Det N .0006 S -> NP VP .00048
				N -> bunker .01 NP -> Det N .003 S -> NP VP .0024

Problem 4

When the PCFG is lexicalized, the first new thing that would need to be introduced is a new base case. We would have the case q(X -> x_i) which would be 0 of there is no rule in the lexicon that could allow us to generate x_i from X.

Based on this we would run a specialized recursive function in order to find out what is the most likely parsing tree. The way it would work is we would run

$\pi(i, j, X)$ for all combinations of non-terminals X and numbers i and j. We would like for the maximum combination value of the rule

$$\pi(i, j, X) = \max\ (q(X \to Y\ Z) * \pi(i, s, Y)\ \pi(s+1, n, Z)\ )$$

n is the number of words in the sentence we are parsing. S would be the point where the sentence would be split into two subsentences. So we will basically be looking for the best way to split the

original sentence (which was rooted in X) into two parts (one rooted in Y and one in Z) given its grammar and lexicon by recursive brute-force.

Disclaimer:
When learning about PCFGs, I have read multiple works online. This includes some of Michael Collins's "Lecicalized Probabilistic Context-Free Grammars"

Problem 5

5.3

The perceptron that performed best had 4 hidden layers and increased training time and learning rate.

The data for 4-layer perceptron

Data for Negative Class:
Precision: 0.8191126279863481
Recall: 0.8
F1: 0.8094435075885329

Data for Positive Class:
Precision: 0.8045602605863192
Recall: 0.8233333333333334
F1: 0.813838550247117

The data for Naive Bayes

Data for Negative Class:
Precision is: 0.7881619937694704
Recall is: 0.8433333333333334
F1 is: 0.8148148148148149

Data for Positive Class:
Precision is: 0.8315412186379928
Recall is: 0.7733333333333333
F1 is: 0.8013816925734024

As we can see Naïve Bayes seems to have higher precision for the positive class and higher recall for the negative class. But overall, the 4-layer perceptron seems to provide better results than naïve bayes. The results that the neural network generates are more balanced and consistently high whereas naïve bayes can have .83 precision vs .80 of the perceptron for the positive class but yet it has .773 recall compared to the perceptron's .823.
Also, while naïve bayes is based on just which word is used in what context, the 4-layer perceptron generates certain attributes that it associates with classes which is a more robust approach. It also becomes better and better with the increased learning time as it has more and

more time to optimize the weights between the different nodes in the network in order to create the best predictions.
Because it incorporates a higher number of linguistic features and can become more and more sophisticated with an increased amount of layers, the perceptron generally leads to better results than naïve bayes. Also, the neural networks are easy to reuse and completely independent of language as it turns words into attributes and uses them for calculations.

On the other side, naïve bayes is much easier and faster to implement. I did not have to implement the neural network this time, but because I read about it, I know that it would probably take me days or weeks to do it well. Naïve Bayes took me a few hours to implement and test. So given its simplicity Naïve Bayes could be the better choice depending on what are the necessities of the project.