Homework 2 COS/LIN 280

Part of Speech Tagging

Due: October 10, 2008 Answer Key

1 Words and Tokens (8 points)

Taking a corpus, tokenizer, and stemmer of your choice, find the stem of every word in the corpus. Answer the following questions:

1. What corpus, tokenizer, and stemmer did you use? Did you do anything else to normalize the corpus (remove HTML, convert to lowercase, remove stop words, etc.)?

   For this assignment, we'll just use Sense and Sensibility (which is tokenized by the Gutenberg reader).

2. How many unique tokens are there? How many tokens in total are there?

```
from nltk.probability import FreqDist
from nltk.corpus import gutenberg
fd = FreqDist()
for word in gutenberg.words(austen-persuasion.txt):
  fd.inc(word)
print "Unique tokens", fd.B()
print "Total tokens", fd.N()
```

3. How many unique stems are there?

```
fd_stem = FreqDist()
from nltk.stem.porter import PorterStemmer
porter = PorterStemmer()
for word in gutenberg.words("austen-persuasion.txt"):
  fd_stem.inc(porter.stem(word))
print "Unique stems", fd_stem.B()
```

4. What are the ten most common tokens longer than two characters?

```
ss = fd.sorted()
count = 0
for word in ss:
  if len(word) > 2:
    print word, fd[word]
      count += 1
  if count >= 10:
     break
```

5. What are the ten most common stems longer than two characters? (It's fine to filter either the tokens by length or filter the stems as long as you filter out punctuation.)

   Same as above, but replace "fd" with "fd_stem."

6. What stem has the most unique tokens that got mapped to it?

```
from collections import defaultdict
d = defaultdict(dict)
# Create a mapping from each stem to all the unique tokens that got mapped to it
for word in gutenberg.words("austen-persuasion.txt"):
  d[porter.stem(word.lower())][word] = 1

# Find out the most tokens that got mapped to a single value
max_tokens = max(map(len, d.values()))

# Print out stems that had that many tokens
for stem in d:
  if len(d[stem]) == max_tokens:
    print stem, d[stem].keys()
```

2 Tagging and NLTK (18 points)

2.1 When taggers go bad (5 points)

Consider the following sentences:

1. British Left Waffles on Falkland Islands
2. Teacher Strikes Idle Kids
3. Clinton Wins Budget; More Lies Ahead
4. Juvenile Court to Try Shooting Defendant

(You're also more than welcome to create or find another sentence that is similarly confusing.) Choose one of these sentences and tag it in two different (but plausible) ways. What sort of taggers, if any, would get these sentences right? Which, if any, would get it wrong? Back up your answers with either sound reasoning or data.

Note that "British Left Waffles on Falkland Islands" can be tagged as follows:

1. British(Adjective) Left(Noun) Waffles(Verb) on(Preposition) Falkland Islands (Two-word Noun Phrase)

2. British(Noun) Left(Verb) Waffles(Noun) on(Preposition) Falkland Islands (Two-word Noun Phrase)

The first tagging is clearly the correct one, as that meaning corresponds to a historical event, while the meaning of the second tagging is just absurd.
Taggers trained on corpora composed of newspapers would probably tag the sentence correctly, because newspapers often use sentences similar in parts of speech to the correct version to convey similar ideas about the political scene, and rarely ever use sentences tagged similarly to the wrong version. Indeed, this sentence seems like it was a newspaper headline.
Taggers trained on corpora composed of political texts would probably tag the sentence correctly, because in the field of politics, 'left' almosts always denotes a noun and 'waffles' almost always denotes a verb.
Taggers trained on any other kind of text would probably tag it wrong, because in other fields, 'left' denotes a verb more often than a noun, and 'waffles' denotes a noun much more often than a verb.
Unigram taggers would probably tag it wrong, because 'waffles' almost always denotes a noun.
n-gram taggers would probably tag it correctly, because 'waffles' is almost always a verb in the context of 'Left'.

2.2 Exploring the tag set (5 points)

There are 265 distinct words in the Brown Corpus having exactly four possible tags (assuming nothing is done to normalize the word forms).

1. Create a table with the integers 1...10 in one column, and the number of distinct words in the corpus having $\{1, \ldots, 10\}$ distinct tags.

Number of Tags	Number of Distinct Words
1	47328
2	7186
3	1146
4	265
5	87
6	27
7	12
8	1
9	1
10	2

2. For the word with the greatest number of distinct tags, print out sentences from the corpus containing the word, one for each possible tag.

   The word with the greatest number of distinct tags is 'that' with 12 distinct tags.
   The following are the 12 distinct tags with an example sentence with that tag, one for each possible tag of 'that':

   (a) CS-NC: ('When', 'WRB'), ('I', 'PPSS-NC'), ('have', 'HV-NC'), ('instructions', 'NNS-NC'), ('to', 'TO-NC'), ('leave', 'VB-NC'), ('is', 'BEZ'), ('equivalent', 'JJ'), ('in', 'IN'), ('meaning', 'NN'), ('to', 'IN'), ('I', 'PPSS-NC'), ('have', 'HV-NC'), ('instructions', 'NNS-NC'), ('that', 'CS-NC'), ('I', 'PPSS-NC'), ('am', 'BEM-NC'), ('to', 'TO-NC'), ('leave', 'VB-NC'), ('this', 'DT-NC'), ('place', 'NN-NC'), (',', ','), ('dominant', 'JJ'), ('stress', 'NN'), ('is', 'BEZ'), ('ordinarily', 'RB'), ('on', 'IN'), ('leave', 'VB-NC'), ('.', '.')

   (b) DT-NC: ('Thus', 'RB'), ('to', 'IN-NC'), ('has', 'HVZ'), ('light', 'JJ'), ('stress', 'NN'), ('both', 'ABX'), ('in', 'IN'), ('that', 'DT-NC'), ('was', 'BEDZ-NC'), ('the', 'AT-NC'), ('conclusion', 'NN-NC'), ('that', 'WPO-NC'), ('I', 'PPSS-NC'), ('came', 'VBD-NC'), ('to', 'IN-NC'), ('and', 'CC'), ('in', 'IN'), ('that', 'DT-NC'), ('was', 'BEDZ-NC'), ('the', 'AT-NC'), ('conclusion', 'NN-NC'), ('I', 'PPSS-NC'), ('came', 'VBD-NC'), ('to', 'IN-NC'), ('.', '.')

   (c) NIL: ('Thus', 'NIL'), (',', ','), ('as', 'NIL'), ('a', 'NIL'), ('development', 'NIL'), ('program', 'NIL'), ('is', 'NIL'), ('being', 'NIL'), ('launched', 'NIL'), (',', ','), ('commitments', 'NIL'), ('and', 'NIL'), ('obligations', 'NIL'), ('must', 'NIL'), ('be', 'NIL'), ('entered', 'NIL'), ('into', 'NIL'), ('in', 'NIL'), ('a', 'NIL'), ('given', 'NIL'), ('year', 'NIL'), ('which', 'NIL'), ('may', 'NIL'), ('exceed', 'NIL'), ('by', 'NIL'), ('twofold', 'NIL'), ('or', 'NIL'), ('threefold', 'NIL'), ('the', 'NIL'), ('expenditures', 'NIL'), ('to', 'NIL'), ('be', 'NIL'), ('made', 'NIL'), ('in', 'NIL'), ('that', 'NIL'), ('year', 'NIL'), ('.', '.')

   (d) WPS-NC: ('But', 'CC'), ('when', 'WRB'), ('to', 'TO-NC'), ('represents', 'VBZ'), ('to', 'IN-NC'), ('consciousness', 'NN-NC'), ('in', 'IN'), ('that', 'WPS-NC'), ('was', 'BEDZ-NC'), ('the', 'AT-NC'), ('moment', 'NN-NC'), ('that', 'CS-NC'), ('I', 'PPSS-NC'), ('came', 'VBD-NC'), ('to', 'IN-NC'), (',', ','), ('and', 'CC'), ('similarly', 'RB'), ('in', 'IN'),

(’that’, ’WPS-NC’), (’was’, ’BEDZ-NC’), (’the’, ’AT-NC’), (’moment’, ’NN-NC’), (’I’, ’PPSS-NC’), (’came’, ’VBD-NC’), (’to’, ’IN-NC’), (’,’, ’,’), (’there’, ’EX’), (’is’, ’BEZ’), (’much’, ’QL’), (’stronger’, ’JJR’), (’stress’, ’NN’), (’on’, ’IN’), (’to’, ’IN-NC’), (’.’, ’.’)

(e) CS-HL: (’According’, ’IN’), (’to’, ’IN’), (’the’, ’AT’), (’official’, ’JJ’), (’interpretation’, ’NN’), (’of’, ’IN’), (’the’, ’AT’), (’Charter’, ’NN-TL’), (’,’, ’,’), (’a’, ’AT’), (’member’, ’NN’), (’cannot’, ’MD*’), (’be’, ’BE’), (’penalized’, ’VBN’), (’by’, ’IN’), (’not’, ’*’), (’having’, ’HVG’), (’the’, ’AT’), (’right’, ’NN’), (’to’, ’TO’), (’vote’, ’VB’), (’in’, ’IN’), (’the’, ’AT’), (’General’, ’JJ-TL’), (’Assembly’, ’NN-TL’), (’for’, ’IN’), (’nonpayment’, ’NN’), (’of’, ’IN’), (’financial’, ’JJ’), (’obligations’, ’NNS’), (’to’, ’IN’), (’the’, ’AT’), (’“’, ’“’), (’special’, ’JJ’), (”””, ”””), (’United’, ’VBN-TL’), (”Nations’”’, ’NNS$-TL’), (’budgets’, ’NNS’), (’,’, ’,’), (’and’, ’CC’), (’of’, ’IN’), (’course’, ’NN’), (’cannot’, ’MD*’), (’be’, ’BE’), (’expelled’, ’VBN’), (’from’, ’IN’), (’the’, ’AT’), (’Organization’, ’NN-TL’), (’(’, ’(’), (’which’, ’WDT’), (’you’, ’PPSS’), (’suggested’, ’VBD’), (’in’, ’IN’), (’your’, ’PP$’), (’editorial’, ’NN’), (’)’, ’)’), (’,’, ’,’), (’due’, ’RB’), (’to’, ’IN-HL’), (’the’, ’AT-HL’), (’fact’, ’NN-HL’), (’that’, ’CS-HL’), (’there’, ’EX’), (’is’, ’BEZ’), (’no’, ’AT’), (’provision’, ’NN’), (’in’, ’IN’), (’the’, ’AT’), (’Charter’, ’NN-TL’), (’for’, ’IN’), (’expulsion’, ’NN’), (’.’, ’.’)

(f) WPO-NC: (’Thus’, ’RB’), (’to’, ’IN-NC’), (’has’, ’HVZ’), (’light’, ’JJ’), (’stress’, ’NN’), (’both’, ’ABX’), (’in’, ’IN’), (’that’, ’DT-NC’), (’was’, ’BEDZ-NC’), (’the’, ’AT-NC’), (’conclusion’, ’NN-NC’), (’that’, ’WPO-NC’), (’I’, ’PPSS-NC’), (’came’, ’VBD-NC’), (’to’, ’IN-NC’), (’and’, ’CC’), (’in’, ’IN’), (’that’, ’DT-NC’), (’was’, ’BEDZ-NC’), (’the’, ’AT-NC’), (’conclusion’, ’NN-NC’), (’I’, ’PPSS-NC’), (’came’, ’VBD-NC’), (’to’, ’IN-NC’), (’.’, ’.’)

(g) WPS: (’She’, ’PPS’), (’was’, ’BEDZ’), (’a’, ’AT’), (’living’, ’VBG’), (’doll’, ’NN’), (’and’, ’CC’), (’no’, ’AT’), (’mistake’, ’NN’), (’–’, ’–’), (’the’, ’AT’), (’blue-black’, ’JJ’), (’bang’, ’NN’), (’,’, ’,’), (’the’, ’AT’), (’wide’, ’JJ’), (’cheekbones’, ’NNS’), (’,’, ’,’), (’olive-flushed’, ’JJ’), (’,’, ’,’), (’that’, ’WPS’), (’betrayed’, ’VBD’), (’the’, ’AT’), (’Cherokee’, ’NP’), (’strain’, ’NN’), (’in’, ’IN’), (’her’, ’PP$’), (’Midwestern’, ’JJ-TL’), (’lineage’, ’NN’), (’,’, ’,’), (’and’, ’CC’), (’the’, ’AT’), (’mouth’, ’NN’), (’whose’, ’WP$’), (’only’, ’AP’), (’fault’, ’NN’), (’,’, ’,’), (’in’, ’IN’), (’the’, ’AT’), (”novelist’s”, ’NN$’), (’carping’, ’VBG’), (’phrase’, ’NN’), (’,’, ’,’), (’was’, ’BEDZ’), (’that’, ’CS’), (’the’, ’AT’), (’lower’, ’JJR’), (’lip’, ’NN’), (’was’, ’BEDZ’), (’a’, ’AT’), (’trifle’, ’NN’), (’too’, ’QL’), (’voluptuous’, ’JJ’), (’.’, ’.’)

(h) WPS-HL: (’Withholding’, ’VBG-HL’), (’of’, ’IN-HL’), (’funds’, ’NNS-HL’), (’to’, ’IN-HL’), (’schools’, ’NNS-HL’), (’that’, ’WPS-HL’), (’deny’, ’VB-HL’), (’children’, ’NNS-HL’), (’on’, ’IN-HL’), (’account’, ’NN-HL’), (’of’, ’IN-HL’), (’race’, ’NN-HL’), (’.’, ’.-HL’)

(i) WPO: (’It’, ’PPS’), (’was’, ’BEDZ’), (’nothing’, ’PN’), (’that’, ’WPO’), (’he’, ’PPS’), (’said’, ’VBD’), (’or’, ’CC’), (’did’, ’DOD’), (’,’, ’,’), (’but’, ’CC’), (’it’, ’PPS’), (’seemed’, ’VBD’), (’so’, ’QL’), (’natural’, ’JJ’), (’to’, ’IN’), (’her’, ’PPO’), (’that’, ’CS’), (’she’, ’PPS’), (’should’, ’MD’), (’be’, ’BE’), (’working’, ’VBG’), (’for’, ’IN’), (’him’, ’PPO’), (’,’, ’,’), (’looking’, ’VBG’), (’forward’, ’RB’), (’to’, ’IN’), (’his’, ’PP$’), (’eventual’, ’JJ’), (’proposal’, ’NN’), (’.’, ’.’)

(j) CS: ('She', 'PPS'), ('was', 'BEDZ'), ('a', 'AT'), ('living', 'VBG'), ('doll', 'NN'), ('and', 'CC'), ('no', 'AT'), ('mistake', 'NN'), ('–', '–'), ('the', 'AT'), ('blue-black', 'JJ'), ('bang', 'NN'), (',', ','), ('the', 'AT'), ('wide', 'JJ'), ('cheekbones', 'NNS'), (',', ','), ('olive-flushed', 'JJ'), (',', ','), ('that', 'WPS'), ('betrayed', 'VBD'), ('the', 'AT'), ('Cherokee', 'NP'), ('strain', 'NN'), ('in', 'IN'), ('her', 'PP$'), ('Midwestern', 'JJ-TL'), ('lineage', 'NN'), (',', ','), ('and', 'CC'), ('the', 'AT'), ('mouth', 'NN'), ('whose', 'WP$'), ('only', 'AP'), ('fault', 'NN'), (',', ','), ('in', 'IN'), ('the', 'AT'), ("novelist's", 'NN$'), ('carping', 'VBG'), ('phrase', 'NN'), (',', ','), ('was', 'BEDZ'), ('that', 'CS'), ('the', 'AT'), ('lower', 'JJR'), ('lip', 'NN'), ('was', 'BEDZ'), ('a', 'AT'), ('trifle', 'NN'), ('too', 'QL'), ('voluptuous', 'JJ'), ('.', '.')

(k) DT: ('“', '“'), ('See', 'VB'), ('that', 'DT'), ('guy', 'NN'), (”””, ”””), ('?', '.'), ('?', '.')

(l) QL: ('Then', 'RB'), (',', ','), ('she', 'PPS'), ('was', 'BEDZ'), ('back', 'RB'), ('on', 'IN'), ('her', 'PP$'), ('feet', 'NNS'), (',', ','), ('winking', 'VBG'), ('and', 'CC'), ('smiling', 'VBG'), ('that', 'QL'), ('enormous', 'JJ'), ('smile', 'NN'), ('(', '('), ('she', 'PPS'), ('had', 'HVD'), ('lots', 'NNS'), ('of', 'IN'), ('wonderful', 'JJ'), ('big', 'JJ'), ('teeth', 'NNS'), ('that', 'CS'), ('you', 'PPSS'), ('never', 'RB'), ('would', 'MD'), ('have', 'HV'), ('suspected', 'VBN'), ('she', 'PPS'), ('had', 'HVD'), ('when', 'WRB'), ('she', 'PPS'), ('was', 'BEDZ'), ('not', '*'), ('smiling', 'VBG'), (')', ')'), ('.', '.')

2.3 Creating taggers (8 points)

Create a default tagger and various unigram and n-gram taggers, incorporating backoff, and train them on part of the Brown corpus.

1. Create three different combinations of the taggers. Test the accuracy of each combined tagger on a different part of the Brown corpus. Which combination works best? The code I used to create the combinations of taggers was:

```
t0 = nltk.DefaultTagger('VBZ')
t1 = nltk.UnigramTagger(brown\_a, backoff=t0)
comboT1 = nltk.BigramTagger(brown\_a, backoff=t1)
comboT2 = nltk.TrigramTagger(brown\_a, backoff=t1)
comboT3 = nltk.TrigramTagger(brown\_a, backoff=comboT1)
```

The descriptions of each combined tagger are as follows:

(a) comboT1 is a bigram tagger that falls back on a unigram tagger that falls back on a default tagger.

(b) comboT2 is a trigram tagger that falls back on a unigram tagger that falls back on a default tagger.

(c) comboT3 is a trigram tagger that falls back on a bigram tagger that falls back on a unigram tagger that falls back on a default tagger.

All taggers were trained on category a of the Brown Corpus.

Running the combined taggers on category c of the Brown Corpus, yields the following results:

(a) comboT1 has 0.740443199686 accuracy.
(b) comboT2 has 0.736979166667 accuracy.
(c) comboT3 has 0.741671580189 accuracy.

Therefore, comboT3 works best.

2. Try varying the size of the training corpus. How does it affect your results?

   When all taggers were trained on categories a and b of the Brown Corpus, the following results were yielded:

   (a) comboT1 has 0.775280070755 accuracy.
   (b) comboT2 has 0.772602201258 accuracy.
   (c) comboT3 has 0.776631289308 accuracy.

   With a larger training corpus, the accuracies (on category c) of all three combined taggers increase.

   When all taggers were trained on categories a, b, and e of the Brown Corpus, the following results were yielded:

   (a) comboT1 has 0.801051493711 accuracy.
   (b) comboT2 has 0.799110652516 accuracy.
   (c) comboT3 has 0.802623820755 accuracy.

   With an even larger training corpus, the accuracies (on category c) of all three combined taggers still increase.

3. Create a regexp tagger that does better than the following (make sure you show how you evaluated both taggers):

```
>>> patterns = [
... (r'.*ing$', 'VBG'), # gerunds
... (r'.*ed$', 'VBD'), # simple past
... (r'.*es$', 'VBZ'), # 3rd singular present
... (r'.*ould$', 'MD'), # modals
... (r'.*\'s$', 'NN$'), # possessive nouns
... (r'.*s$', 'NNS'), # plural nouns
... (r'^-?[0-9]+(.[0-9]+)?$', 'CD'), # cardinal numbers
... (r'.*', 'NN') # nouns (default)
... ]


new_patterns=[
(r'^\.$', '.'), #Punctuation marks
(r'^\($', '('),
(r'^\)$', ')'),
(r'^--$', '--'),
(r'^,$', ','),
(r'^:$', ':'),
(r'^n[o\']t$', '*'),
(r'^to$', 'TO'),
(r'^(about|above|across|after|against|along|among|around|at|before|behind|below|beneath|
(r'^([aA]|[tT]he|[nN]o)$', 'AT'), #articles
(r'^(and|or)$', 'CC'), #conjunctions
(r'^[wW](hat|hich)(ever)?$', 'WDT'), #Wh-words
(r'^[wW]hose(ver)?$', 'WP$'),
(r'^[wW]hom$', 'WPO'),
(r'^[wW]ho(so)?(ever)?$', 'WPS'),
(r'^([wW]hen|[wW]ere|[wW]hy|[hH]ow)$', 'WRB'),
(r'^[tT]hat$', 'CS'),
(r'^[hH]ave$', 'HV'), #Have
(r'^[hH]ad$', 'HVD'),
(r'^[hH]aving$', 'HVG'),
(r'^[hH]as$', 'HVZ'),
(r'^[dD]o$', 'DO'), #Do
(r'^[dD]id$', 'DOD'),
(r'^[dD]oes$', 'DOZ'),
(r'^[bB]e$', 'BE'), #Be
(r'^[wW]ere$', 'BED'),
(r'^[wW]as$', 'BEDZ'),
(r'^[bB]eing$', 'BEG'),
(r'^[aA]m$', 'BEM'),
(r'^[bB]een$', 'BEN'),
(r'^[aA]r[et]$', 'BER'),
(r'^[iI]s$', 'BEZ'),
(r'^[tT]his$', 'DT'),
(r'^([sS]ome|[aA]ny)$', 'DTI'),
(r'^[tT]h[eo]se$', 'DTS'),
(r'^.*[eE]ither$', 'DTX'),
(r'.+est$','JJT'), # superlative comparative
(r'.{3,}ly$','RB'), #adverb
(r'^(one|two|three|four|five|six|seven|eight|nine|ten|eleven|twelve|.+teen|twenty|thirty
(r'^(first|second|third|fourth|fifth|sixth|seventh|eighth|ninth|tenth|eleventh|twelfth|t
(r'^[0-9]+(st|nd|rd|th)$','OD'),
(r'.+(thing|one)$','PN'),
(r'.{2,}ing$','VBG'), #consider length
(r'.*[^aeou]ed$','VBD'), #no vowels + ed except for i
(r'.*es$','VBZ'),
(r'.*ould$','MD'),
(r'.*[^s]\'s$','NN$'), #check for no s before the '
(r'.*\'s$','NNS$'), #catch the plurals afterwards
(r'.*[^s]s$','NNS'), #no ss
(r'-?[0-9]+(.[0-9]+)?$','CD'),
(r'^[A-Z].*[^s]\'s$','NP$'), #proper nouns
(r'^[A-Z].*\'s$','NPS$'),
(r'^[A-Z].*[^s]s$','NPS'),
(r'^[A-Z].*$','NP'),
(r'.*','NN') ] #catchall

With this I get: 0.63049618273535569 accuracy
compared to the original tagger: 0.19460649490003576

I tested using the Brown corpus, categories A and B like this:
brown_sents = nltk.corpus.brown.tagged_sents( categories=['a','b'] )
nltk.tag.accuracy(regexp_tagger, brown_sents)
```

3 Viterbi Algorithm (14 Points)

Consider the following sentences written in Klingon. For each sentence, the part of speech of each "word" has been given (for ease of translation, some prefixes/suffixes have been treated as words), along with a translation. Using these training sentences, we're going to build a hidden Markov model to predict the part of speech of an unknown sentence using the Viterbi algorithm.

N	PRO	V	N	PRO
pa'Daq	ghah	taH	tera'ngan	'e
room (inside)	he	is	human	of

The human is in the room

V	N	V	N
ja'chuqmeH	rojHom	neH	tera'ngan
in order to parley	truce	want	human

The enemy commander wants a truce in order to parley

N	V	N	CONJ	N	V	N
tera'ngan	qIp	puq	'eg	puq	qIp	tera'ngan
human	bit	child	and	child	bit	child

The child bit the human, and the human bit the child

3.1 Emission Probability (4 points)

Compute the frequencies of each part of speech in the table below for nouns and verbs. We'll use a smoothing factor of 0.1 (as discussed in class) to make sure that no event is impossible; add this number to all of your observations. Two parts of speech have already been done for you. After you've done this, compute the emission probabilities in a similar table.

	NOUN	VERB	CONJ	PRO
'e	0.1	0.1	0.1	1.1
'eg	0.1	0.1	1.1	0.1
ghaH	0.1	0.1	0.1	1.1
ja'chuqmeH	0.1	1.1	0.1	0.1
legh	0.1	0.1	0.1	0.1
neH	0.1	1.1	0.1	0.1
pa'Daq	1.1	0.1	0.1	0.1
puq	2.1	0.1	0.1	0.1
qIp	0.1	2.1	0.1	0.1
rojHom	1.1	0.1	0.1	0.1
taH	0.1	1.1	0.1	0.1
tera'ngan	4.1	0.1	0.1	0.1
yaS	0.1	0.1	0.1	0.1

3.2 Start and Transition Probability (4 points)

Now, for each part of speech, total the number of times it transitioned to each other part of speech. Again, use a smoothing factor of 0.1. After you've done this, compute the start and transition probabilities.

	NOUN	VERB	CONJ	PRO
START				
N			1.1	2.1
V			0.1	0.1
CONJ			0.1	0.1
PRO			0.1	0.1

3.3 Viterbi Decoding (6 points)

Now consider the following sentence: "tera'ngan legh yaS".

1. Suppose that we knew "legh" were a pronoun (it's not, but pronouns often act like its true part of speech in Klingon). What would be the probability of each of the four parts of speech for "yaS"?

   The probability "yaS" is a

   noun is $.1/1.4 \approx 7\%$

   verb is $1.1/1.4 \approx 79\%$

   conjunction is $.1/1.4 \approx 7\%$

   pronoun is $.1/1.4 \approx 7\%$

2. Create the decoding matrix of this sentence for nouns and verbs (ignore other parts of speech if you want; doing so won't prevent you from finding the right answer). You should have at least six numbers: $\log \delta_n(k)$ for $n = 1 \dots 3$ and k for both nouns and verbs.

   For k =N:

   - $\delta_1(k) = \pi_k \beta_{k,x_1} = (2.1/3.4)(4.1/9.3) = .272$
     $\log \delta_1(k) = \log .272 = -1.30$
   - $\delta_1(k)\theta_{k,k} = .272(.1/6.4) = .00425$
     $\delta_1(j)\theta_{j,k} = .00514(5.1/5.4) = .00485$
     $\delta_2(k) = max_j(\delta_1(j)\theta_{j,k})\beta_{k,x_2} = .00485(.1/9.3) = .0000522$
     $\log \delta_2(k) = \log .0000522 = -9.86$
   - $\delta_2(k)\theta_{k,k} = .0000522(.1/6.4) = .000000816$
     $\delta_2(j)\theta_{j,k} = .00210(5.1/5.4) = .00198$
     $\delta_3(k) = max_j(\delta_1(j)\theta_{j,k})\beta_{k,x_2} = .00198(.1/9.3) = .0000213$
     $\log \delta_3(k) = \log .0000213 = -10.8$

   For k =V:

- $\delta_1(k) = \pi_k \beta_{k,x_1} = (1.1/3.4)(.1/6.3) = .00514$
  $\log \delta_1(k) = \log .00514 = -5.27$
- $\delta_1(k)\theta_{k,k} = .00514(.1/5.4) = .0000952$
  $\delta_1(j)\theta_{j,k} = .272(3.1/6.4) = .132$
  $\delta_2(k) = max_j(\delta_1(j)\theta_{j,k})\beta_{k,x_2} = .132(.1/6.3) = .00210$
  $\log \delta_2(k) = \log .00210 = -6.17$
- $\delta_2(k)\theta_{k,k} = .00210(.1/5.4) = .0000389$
  $\delta_2(j)\theta_{j,k} = .0000522(3.1/6.4) = .0000253$
  $\delta_3(k) = max_j(\delta_1(j)\theta_{j,k})\beta_{k,x_2} = .0000389(.1/6.3) = .000000617$
  $\log \delta_3(k) = \log .000000617 = -14.3$

POS	$\log \delta_1(k)$	$\log \delta_2(k)$	b_2	$\log \delta_3(k)$	b_3
N	-1.30	-9.86	V	-10.8	V
V	-5.27	-6.17	N	-14.3	V
Word	tera'ngan	legh		yaS	

3. What is the most likely sequence of parts of speech?

   Noun, verb, noun
   because $-10.8 > -14.3$, so it most likely ends in a noun, which is most likely preceded by a verb, which is most likely preceded by a noun.

4. What is the probability of your previous answer?

   The probability of my previous answer is $\delta_3(noun) = .0000213$.

5. (For fun, not for credit) What do you think this sentence means? What word is the subject of the sentence?

   It seems that Klingon has OVS verb order. So the subject of the sentence is "yaS," which means officer. "legh" is a verb that means "to see," so that sentence means "the officer sees the human."