Gourab Bhattacharyya – 170048888 – CSE 628 Assignment#2 – Report

1. System configuration:

OS: MAC OS 10.13.3
Python Version: Python 2.7.10

Need to install **num2words** package. This is being used in feat_gen.py file.

2. Commands to execute

For LR:
python data.py --model lr --test

./conlleval.pl -r -d \\t < ./predictions/twitter_dev.lr.pred

For CRF:
python data.py --model crf --test

./conlleval.pl -r -d \\t < ./predictions/twitter_dev.crf.pred

3. Implementation of viterbi.py

- Initialized a zero matrix of dimension of emission_scores. Name this as scores
- Initialized a zero matrix but with type = int3 2 of dimension of emission_scores. Name this as back_pointers
- Add start_scores to emission_scores
- Add start_scores to trans_scores
- Set 0^{th} record of scores as 0^{th} records of emission_scores added with start_scores
- For each index, I in range (1, N(# of tokens)):
    - Set score_with_transition as expanded dimension of scores[i-1] row to single and adding trns_scores with that.
    - ith value of scores is set as emission_scores[i] and max value of score_with_transition
    - set back_pointers[i] as row wise max value of score_with_transition
- Generate path as viterbi which is row wise max of scores[-1] added with end_scores
- For each element, bp in reversed back_pointers:
    - append value at viterbi[-1] from bp to viterbi list
- Reverse the generated viterbi list
- Generate viterbi_score as sum of max of scores[-1] added with end_scores
- Return viterbi and viterbi_score.

4. Updated code in tagger.py for CRF

I have updated **decay_exponent** in the below line for CRF in tagger.py:

```
self.cls = struct_perceptron.StructuredPerceptron(self, max_iter=25,decay_exponent=0.1,
average=True, verbose=True)
```

5. Description of the added features

Preprocessing step:

a. Strip each word in the sentence to remove extra line spaces.
b. Replace “tab” in between words with single space (“ ”).

Features Addition step:

a. Is Emoji – This will check if emoticons are present in the tweets or not

```
# is Emoji
emoji = {
    "&lt;3": "positive", ":D": "positive", ":d": "positive", ":dd": "positive", ":P": "positive",
    ":p": "positive",
    "8)": "positive",
    "8-)": "positive", ":-)": "positive", ":)": "positive", ";)": "positive", "(-:": "positive",
    "(:": "positive",
    ":')": "positive", "xD": "positive", "XD": "positive", "yay!": "positive", "yay": "positive",
    "yaay": "positive",
    "yaaay": "positive", "yaaaay": "positive", "yaaaaay": "positive", "Yay!": "positive",
    "Yay": "positive",
    "Yaay": "positive",
    "Yaaay": "positive", "Yaaaay": "positive", "Yaaaaay": "positive", ":/": "negative",
    "&gt;": "negative",
    ":'(": "negative",
    ":-(": "negative", ":(": "negative", ":s": "negative", ":-s": "negative", "-_-": "negative",
    "-.-": "negative"}

isPresent = "IS_NOT_EMOJI"
if word in emoji:
    isPresent = "IS_EMOJI"
ftrs.append(isPresent)
```

b. Is Hashtag - This will check if hashtag(#string) are present (starts with #) in the tweets or not

```
# is HashTag
if word.startswith("#"):
    ftrs.append("IS_HASHTAG" if len(word[1:]) != 0 else "IS_NOT_HASHTAG")
```

c. **is URL – This will check if “https://” are present in the tweets or not and treat that as URL**

```python
# is URL
if word.startswith("http://"):
   ftrs.append("IS_URL" if len(word[8:]) != 0 else "IS_NOT_URL")
```

d. **is Header – This will check if tweet starts with “@” then it considers it as Header**

```python
# is Header
if word.startswith("@"):
   ftrs.append("IS_HEADER" if len(word[1:]) != 0 else "IS_NOT_HEADER")
```

e. **has Exclamaton – This will check if exclamation “!” are present in the tweets or not**

```python
# has Exclamaton
if "!" in word:
   ftrs.append("HAS_EXCLAIMATION")
```

f. **has Question – This will check if “?” are present in the tweets or not**

```python
# has Question
if "?" in word:
   ftrs.append("HAS_QUESTION")
```

g. **ends with ed – This will check if tweets end with “ed”**

```python
# ends with ed
if word.split(" ")[0].endswith("ed"):
   ftrs.append("ED_ENDED")
```

h. **ends with ing – This will check if tweets end with “ing”**

```python
# ends with ing
if word.split(" ")[0].endswith("ing"):
   ftrs.append("ING_ENDED")
```

i. **POS count – This will count the POS tag for each word over the entire sentence**

```python
# POS count
if posCount(word, sent) > 0:
   ftrs.append("HAS " + num2words(posCount(word, sent)).upper() + " COUNT")
```

j. **avg length – This will calculate the average length of words using the sentence**

```
# avg length
ftrs.append("HAS " + getLen(word, sent).upper() + " LENGTH")
```

k. **word length – This will calculate length of each word**

```
# word length
ftrs.append("HAS LENGTH " + str(len(word) - 1) + " WORD")
```

l. **POS index – This will find the maximum index of POS tag from the sentence for a word**

```
# POS index
if posIndex(word, sent) > 0:
    ftrs.append("HAS " + num2words(posIndex(word, sent)).upper() + " INDEX")
```

m. **byte length – This will calculate the byte length of each word**

```
# byte length
ftrs.append("HAS WORD " + num2words(sys.getsizeof(word)).upper() + " BYTELENGTH")
```

n. **HASH length – This will calculate the HASH value of each word and considers this as features**

```
# HASH length
ftrs.append("HAS HASH_" + str(hash(word.split(" ")[0])) + "_LENGTH")
```

6. Feature Comparison with example sentence:

Feature type	Basic Features (default provided)

Feature Values	`# bias`
	`ftrs.append("BIAS")`
	`# position features`
	`if i == 0:`
	    `ftrs.append("SENT_BEGIN")`
	`if i == len(sent)-1:`
	    `ftrs.append("SENT_END")`

	# the word itself
	word = unicode(sent[i])
	ftrs.append("WORD=" + word)
	ftrs.append("LCASE=" + word.lower())
	# some features of the word
	if word.isalnum():
	    ftrs.append("IS_ALNUM")
	if word.isnumeric():
	    ftrs.append("IS_NUMERIC")
	if word.isdigit():
	    ftrs.append("IS_DIGIT")
	if word.isupper():
	    ftrs.append("IS_UPPER")
	if word.islower():
	    ftrs.append("IS_LOWER")
	
	# previous/next word feats
	if add_neighs:
	    if i > 0:
	        for pf in token2features(sent, i-1, add_neighs = False):
	            ftrs.append("PREV_" + pf)
	    if i < len(sent)-1:
	        for pf in token2features(sent, i+1, add_neighs = False):
	            ftrs.append("NEXT_" + pf)

Example sentence	sents = [
	    ["What NOUN",
	"a DET",
	"productive ADJ",
	"daying NOUN"]
	    ]

Output	What NOUN : ['BIAS', 'SENT_BEGIN', u'WORD=What NOUN', u'LCASE=what noun', 'NEXT_BIAS', u'NEXT_WORD=a DET', u'NEXT_LCASE=a det'] a DET : ['BIAS', u'WORD=a DET', u'LCASE=a det', 'PREV_BIAS', 'PREV_SENT_BEGIN', u'PREV_WORD=What NOUN', u'PREV_LCASE=what noun', 'NEXT_BIAS', u'NEXT_WORD=productive ADJ', u'NEXT_LCASE=productive adj'] productive ADJ : ['BIAS', u'WORD=productive ADJ', u'LCASE=productive adj', 'PREV_BIAS', u'PREV_WORD=a DET', u'PREV_LCASE=a det', 'NEXT_BIAS', 'NEXT_SENT_END', u'NEXT_WORD=daying NOUN', u'NEXT_LCASE=daying noun'] daying NOUN : ['BIAS', 'SENT_END', u'WORD=daying NOUN', u'LCASE=daying noun', 'PREV_BIAS', u'PREV_WORD=productive ADJ', u'PREV_LCASE=productive adj']

Feature type	Added Customized New Features

Feature Values	# more added features
	
	# is Emoji
	emoji = {
	    "<3": "positive", ":D": "positive", ":d": "positive", ":dd": "positive", ":P": "positive",
	    ":p": "positive",

```
    "8)": "positive",
    "8-)": "positive", ":-)": "positive", ":)": "positive", ";)": "positive", "(-:": "positive",
    "(:": "positive",
    ":')": "positive", "xD": "positive", "XD": "positive", "yay!": "positive", "yay":
"positive",
    "yaay": "positive",
    "yaaay": "positive", "yaaaay": "positive", "yaaaaay": "positive", "Yay!": "positive",
    "Yay": "positive",
    "Yaay": "positive",
    "Yaaay": "positive", "Yaaaay": "positive", "Yaaaaay": "positive", ":/": "negative",
    "&gt;": "negative",
    ":'(": "negative",
    ":-(": "negative", ":(": "negative", ":s": "negative", ":-s": "negative", "-_-":
"negative",
    "-.-": "negative"}

isPresent = "IS_NOT_EMOJI"
if word in emoji:
    isPresent = "IS_EMOJI"
ftrs.append(isPresent)

# is HashTag
if word.startswith("#"):
    ftrs.append("IS_HASHTAG" if len(word[1:]) != 0 else "IS_NOT_HASHTAG")

# is URL
if word.startswith("http://"):
    ftrs.append("IS_URL" if len(word[8:]) != 0 else "IS_NOT_URL")


# is Header
if word.startswith("@"):
    ftrs.append("IS_HEADER" if len(word[1:]) != 0 else "IS_NOT_HEADER")




# has Exclamaton
if "!" in word:
    ftrs.append("HAS_EXCLAIMATION")


# has Question
if "?" in word:
    ftrs.append("HAS_QUESTION")



# ends with ed
if word.split(" ")[0].endswith("ed"):
    ftrs.append("ED_ENDED")

# ends with ing
```

	<pre>if word.split(" ")[0].endswith("ing"):
	    ftrs.append("ING_ENDED")
	
	# POS count
	if posCount(word, sent) > 0:
	    ftrs.append("HAS " + num2words(posCount(word, sent)).upper() + " COUNT")
	
	# avg length
	ftrs.append("HAS " + getLen(word, sent).upper() + " LENGTH")
	
	# word length
	ftrs.append("HAS LENGTH " + str(len(word) - 1) + " WORD")
	
	# POS index
	# print(posIndex(word, sent).upper())
	if posIndex(word, sent) > 0:
	    ftrs.append("HAS " + num2words(posIndex(word, sent)).upper() + " INDEX")
	
	# byte length
	ftrs.append("HAS WORD " + num2words(sys.getsizeof(word)).upper() + "
	BYTELENGTH")
	
	# HASH length
	ftrs.append("HAS HASH_" + str(hash(word.split(" ")[0])) + "_LENGTH")
	
	#done adding features</pre>

Example sentence	<pre>sents = [
	    ["@LogUpdate",
	    "What NOUN",
	"a DET",
	"productive ADJ",
	"day:D NOUN",
	":D ADJ",
	"Enjoyed VERB",
	"Walking VERB",
	"around DET",
	"Beach!!! ADJ",
	"http://beachPhotos.com X",
	"#FunUnlimited ADJ"]
	    ]</pre>

Output	@LogUpdate : ['BIAS', 'SENT_BEGIN', u'WORD=@LogUpdate', u'LCASE=@logupdate', 'IS_NOT_EMOJI', **'IS_HEADER', u'HAS ONE COUNT', u'HAS ONE LENGTH', 'HAS LENGTH 9 WORD', u'HAS WORD SEVENTY BYTELENGTH', 'HAS HASH_7948059180188312563_LENGTH**', 'NEXT_BIAS', u'NEXT_WORD=What NOUN', u'NEXT_LCASE=what noun', 'NEXT_IS_NOT_EMOJI', u'NEXT_HAS TWO COUNT',

	u'NEXT_HAS ONE LENGTH', 'NEXT_HAS LENGTH 8 WORD', u'NEXT_HAS WORD SIXTY-EIGHT BYTELENGTH', 'NEXT_HAS HASH_-7487826120235232766_LENGTH'] What NOUN : ['BIAS', u'WORD=What NOUN', u'LCASE=what noun', 'IS_NOT_EMOJI', u'HAS TWO COUNT', u'HAS ONE LENGTH', 'HAS LENGTH 8 WORD', u'HAS WORD SIXTY-EIGHT BYTELENGTH', 'HAS HASH_-7487826120235232766_LENGTH', 'PREV_BIAS', 'PREV_SENT_BEGIN', u'PREV_WORD=@LogUpdate', u'PREV_LCASE=@logupdate', 'PREV_IS_NOT_EMOJI', 'PREV_IS_HEADER', u'PREV_HAS ONE COUNT', u'PREV_HAS ONE LENGTH', 'PREV_HAS LENGTH 9 WORD', u'PREV_HAS WORD SEVENTY BYTELENGTH', 'PREV_HAS HASH_7948059180188312563_LENGTH', 'NEXT_BIAS', u'NEXT_WORD=a DET', u'NEXT_LCASE=a det', 'NEXT_IS_NOT_EMOJI', u'NEXT_HAS TWO COUNT', u'NEXT_HAS THREE LENGTH', 'NEXT_HAS LENGTH 4 WORD', u'NEXT_HAS WORD SIXTY BYTELENGTH', 'NEXT_HAS HASH_12416037344_LENGTH'] a DET : ['BIAS', u'WORD=a DET', u'LCASE=a det', 'IS_NOT_EMOJI', u'HAS TWO COUNT', u'HAS THREE LENGTH', 'HAS LENGTH 4 WORD', u'HAS WORD SIXTY BYTELENGTH', 'HAS HASH_12416037344_LENGTH', 'PREV_BIAS', u'PREV_WORD=What NOUN', u'PREV_LCASE=what noun', 'PREV_IS_NOT_EMOJI', u'PREV_HAS TWO COUNT', u'PREV_HAS ONE LENGTH', 'PREV_HAS LENGTH 8 WORD', u'PREV_HAS WORD SIXTY-EIGHT BYTELENGTH', 'PREV_HAS HASH_-7487826120235232766_LENGTH', 'NEXT_BIAS', u'NEXT_WORD=productive ADJ', u'NEXT_LCASE=productive adj', 'NEXT_IS_NOT_EMOJI', u'NEXT_HAS FOUR COUNT', u'NEXT_HAS ONE LENGTH', 'NEXT_HAS LENGTH 13 WORD', u'NEXT_HAS WORD SEVENTY-EIGHT BYTELENGTH', 'NEXT_HAS HASH_3331480792833632675_LENGTH'] productive ADJ : ['BIAS', u'WORD=productive ADJ', u'LCASE=productive adj', 'IS_NOT_EMOJI', u'HAS FOUR COUNT', u'HAS ONE LENGTH', 'HAS LENGTH 13 WORD', u'HAS WORD SEVENTY-EIGHT BYTELENGTH', 'HAS HASH_3331480792833632675_LENGTH', 'PREV_BIAS', u'PREV_WORD=a DET', u'PREV_LCASE=a det', 'PREV_IS_NOT_EMOJI', u'PREV_HAS TWO COUNT', u'PREV_HAS THREE LENGTH', 'PREV_HAS LENGTH 4 WORD', u'PREV_HAS WORD SIXTY BYTELENGTH', 'PREV_HAS HASH_12416037344_LENGTH', 'NEXT_BIAS', u'NEXT_WORD=day:D NOUN', u'NEXT_LCASE=day:d noun', 'NEXT_IS_NOT_EMOJI', u'NEXT_HAS TWO COUNT', u'NEXT_HAS ONE LENGTH', 'NEXT_HAS LENGTH 9 WORD', u'NEXT_HAS WORD SEVENTY BYTELENGTH', 'NEXT_HAS HASH_-6734976527753343071_LENGTH'] day:D NOUN : ['BIAS', u'WORD=day:D NOUN', u'LCASE=day:d noun', 'IS_NOT_EMOJI', u'HAS TWO COUNT', u'HAS ONE LENGTH', 'HAS LENGTH 9 WORD', u'HAS WORD SEVENTY BYTELENGTH', 'HAS HASH_-6734976527753343071_LENGTH', 'PREV_BIAS', u'PREV_WORD=productive ADJ', u'PREV_LCASE=productive adj', 'PREV_IS_NOT_EMOJI', u'PREV_HAS FOUR COUNT', u'PREV_HAS ONE LENGTH', 'PREV_HAS LENGTH 13 WORD', u'PREV_HAS WORD SEVENTY-EIGHT BYTELENGTH', 'PREV_HAS HASH_3331480792833632675_LENGTH', 'NEXT_BIAS', u'NEXT_WORD=:D ADJ', u'NEXT_LCASE=:d adj', 'NEXT_IS_UPPER', 'NEXT_IS_EMOJI', u'NEXT_HAS FOUR COUNT', u'NEXT_HAS TWO LENGTH', 'NEXT_HAS LENGTH 5 WORD', u'NEXT_HAS WORD SIXTY-TWO BYTELENGTH', 'NEXT_HAS HASH_7424044602067048_LENGTH'] :D ADJ : ['BIAS', u'WORD=:D ADJ', u'LCASE=:d adj', 'IS_UPPER', **'IS_EMOJI',** u'HAS FOUR COUNT', u'HAS TWO LENGTH', 'HAS LENGTH 5 WORD', u'HAS WORD SIXTY-TWO BYTELENGTH', 'HAS HASH_7424044602067048_LENGTH', 'PREV_BIAS', u'PREV_WORD=day:D NOUN', u'PREV_LCASE=day:d noun', 'PREV_IS_NOT_EMOJI', u'PREV_HAS TWO COUNT', u'PREV_HAS ONE LENGTH', 'PREV_HAS LENGTH 9 WORD', u'PREV_HAS WORD SEVENTY BYTELENGTH', 'PREV_HAS HASH_-6734976527753343071_LENGTH', 'NEXT_BIAS', u'NEXT_WORD=Enjoyed VERB',

	u'NEXT_LCASE=enjoyed verb', 'NEXT_IS_NOT_EMOJI', 'NEXT_ED_ENDED', u'NEXT_HAS TWO COUNT', u'NEXT_HAS ONE LENGTH', 'NEXT_HAS LENGTH 11 WORD', u'NEXT_HAS WORD SEVENTY-FOUR BYTELENGTH', 'NEXT_HAS HASH_-4314997815532340483_LENGTH'] Enjoyed VERB : ['BIAS', u'WORD=Enjoyed VERB', u'LCASE=enjoyed verb', 'IS_NOT_EMOJI', 'ED_ENDED', u'HAS TWO COUNT', u'HAS ONE LENGTH', 'HAS LENGTH 11 WORD', u'HAS WORD SEVENTY-FOUR BYTELENGTH', 'HAS HASH_-4314997815532340483_LENGTH', 'PREV_BIAS', u'PREV_WORD=:D ADJ', u'PREV_LCASE=:d adj', 'PREV_IS_UPPER', 'PREV_IS_EMOJI', u'PREV_HAS FOUR COUNT', u'PREV_HAS TWO LENGTH', 'PREV_HAS LENGTH 5 WORD', u'PREV_HAS WORD SIXTY-TWO BYTELENGTH', 'PREV_HAS HASH_7424044602067048_LENGTH', 'NEXT_BIAS', u'NEXT_WORD=Walking VERB', u'NEXT_LCASE=walking verb', 'NEXT_IS_NOT_EMOJI', 'NEXT_ING_ENDED', u'NEXT_HAS TWO COUNT', u'NEXT_HAS ONE LENGTH', 'NEXT_HAS LENGTH 11 WORD', u'NEXT_HAS WORD SEVENTY-FOUR BYTELENGTH', 'NEXT_HAS HASH_-2629076805332514078_LENGTH'] Walking VERB : ['BIAS', u'WORD=Walking VERB', u'LCASE=walking verb', 'IS_NOT_EMOJI', **'ING_ENDED'**, u'HAS TWO COUNT', u'HAS ONE LENGTH', 'HAS LENGTH 11 WORD', u'HAS WORD SEVENTY-FOUR BYTELENGTH', 'HAS HASH_-2629076805332514078_LENGTH', 'PREV_BIAS', u'PREV_WORD=Enjoyed VERB', u'PREV_LCASE=enjoyed verb', 'PREV_IS_NOT_EMOJI', 'PREV_ED_ENDED', u'PREV_HAS TWO COUNT', u'PREV_HAS ONE LENGTH', 'PREV_HAS LENGTH 11 WORD', u'PREV_HAS WORD SEVENTY-FOUR BYTELENGTH', 'PREV_HAS HASH_-4314997815532340483_LENGTH', 'NEXT_BIAS', u'NEXT_WORD=around DET', u'NEXT_LCASE=around det', 'NEXT_IS_NOT_EMOJI', u'NEXT_HAS TWO COUNT', u'NEXT_HAS ONE LENGTH', 'NEXT_HAS LENGTH 9 WORD', u'NEXT_HAS WORD SEVENTY BYTELENGTH', 'NEXT_HAS HASH_6524420213203603013_LENGTH'] around DET : ['BIAS', u'WORD=around DET', u'LCASE=around det', 'IS_NOT_EMOJI', u'HAS TWO COUNT', u'HAS ONE LENGTH', 'HAS LENGTH 9 WORD', u'HAS WORD SEVENTY BYTELENGTH', 'HAS HASH_6524420213203603013_LENGTH', 'PREV_BIAS', u'PREV_WORD=Walking VERB', u'PREV_LCASE=walking verb', 'PREV_IS_NOT_EMOJI', 'PREV_ING_ENDED', u'PREV_HAS TWO COUNT', u'PREV_HAS ONE LENGTH', 'PREV_HAS LENGTH 11 WORD', u'PREV_HAS WORD SEVENTY-FOUR BYTELENGTH', 'PREV_HAS HASH_-2629076805332514078_LENGTH', 'NEXT_BIAS', u'NEXT_WORD=Beach!!! ADJ', u'NEXT_LCASE=beach!!! adj', 'NEXT_IS_NOT_EMOJI', 'NEXT_HAS_EXCLAIMATION', u'NEXT_HAS FOUR COUNT', u'NEXT_HAS ONE LENGTH', 'NEXT_HAS LENGTH 11 WORD', u'NEXT_HAS WORD SEVENTY-FOUR BYTELENGTH', 'NEXT_HAS HASH_-5772302544317265036_LENGTH'] Beach!!! ADJ : ['BIAS', u'WORD=Beach!!! ADJ', u'LCASE=beach!!! adj', 'IS_NOT_EMOJI', 'HAS_EXCLAIMATION', u'HAS FOUR COUNT', u'HAS ONE LENGTH', 'HAS LENGTH 11 WORD', u'HAS WORD SEVENTY-FOUR BYTELENGTH', 'HAS HASH_-5772302544317265036_LENGTH', 'PREV_BIAS', u'PREV_WORD=around DET', u'PREV_LCASE=around det', 'PREV_IS_NOT_EMOJI', u'PREV_HAS TWO COUNT', u'PREV_HAS ONE LENGTH', 'PREV_HAS LENGTH 9 WORD', u'PREV_HAS WORD SEVENTY BYTELENGTH', 'PREV_HAS HASH_6524420213203603013_LENGTH', 'NEXT_BIAS', u'NEXT_WORD=http://beachPhotos.com X', u'NEXT_LCASE=http://beachphotos.com x', 'NEXT_IS_NOT_EMOJI', 'NEXT_IS_URL', u'NEXT_HAS ONE COUNT', u'NEXT_HAS ZERO LENGTH', 'NEXT_HAS LENGTH 23 WORD', u'NEXT_HAS WORD NINETY-EIGHT BYTELENGTH', 'NEXT_HAS HASH_4472518068751912937_LENGTH'] http://beachPhotos.com X : ['BIAS', u'WORD=http://beachPhotos.com X', u'LCASE=http://beachphotos.com x', 'IS_NOT_EMOJI', **'IS_URL'**, u'HAS ONE COUNT',

	u'HAS ZERO LENGTH', 'HAS LENGTH 23 WORD', u'HAS WORD NINETY-EIGHT BYTELENGTH', 'HAS HASH_4472518068751912937_LENGTH', 'PREV_BIAS', u'PREV_WORD=Beach!!! ADJ', u'PREV_LCASE=beach!!! adj', 'PREV_IS_NOT_EMOJI', 'PREV_HAS_EXCLAIMATION', u'PREV_HAS FOUR COUNT', u'PREV_HAS ONE LENGTH', 'PREV_HAS LENGTH 11 WORD', u'PREV_HAS WORD SEVENTY-FOUR BYTELENGTH', 'PREV_HAS HASH_-5772302544317265036_LENGTH', 'NEXT_BIAS', 'NEXT_SENT_END', u'NEXT_WORD=#FunUnlimited ADJ', u'NEXT_LCASE=#fununlimited adj', 'NEXT_IS_NOT_EMOJI', 'NEXT_IS_HASHTAG', 'NEXT_ED_ENDED', u'NEXT_HAS FOUR COUNT', u'NEXT_HAS ONE LENGTH', 'NEXT_HAS LENGTH 16 WORD', u'NEXT_HAS WORD EIGHTY-FOUR BYTELENGTH', 'NEXT_HAS HASH_3010452094198567990_LENGTH'] #FunUnlimited ADJ : ['BIAS', 'SENT_END', u'WORD=#FunUnlimited ADJ', u'LCASE=#fununlimited adj', 'IS_NOT_EMOJI', '**IS_HASHTAG', 'ED_ENDED'**, u'HAS FOUR COUNT', u'HAS ONE LENGTH', 'HAS LENGTH 16 WORD', u'HAS WORD EIGHTY-FOUR BYTELENGTH', 'HAS HASH_3010452094198567990_LENGTH', 'PREV_BIAS', u'PREV_WORD=http://beachPhotos.com X', u'PREV_LCASE=http://beachphotos.com x', 'PREV_IS_NOT_EMOJI', 'PREV_IS_URL', u'PREV_HAS ONE COUNT', u'PREV_HAS ZERO LENGTH', 'PREV_HAS LENGTH 23 WORD', u'PREV_HAS WORD NINETY-EIGHT BYTELENGTH', 'PREV_HAS HASH_4472518068751912937_LENGTH']

7. Comparison of Logistic Regression and CRFs

Logistic Regression with Basic and Added Features	
Result with basic features on DEV data	### Dev evaluation Token-wise accuracy 84.389782403 Token-wise F1 (macro) 83.3342279971 Token-wise F1 (micro) 84.389782403 Sentence-wise accuracy 8.92857142857 precision recall f1-score support . 0.94 0.98 0.96 254 ADJ 0.73 0.36 0.49 99 ADP 0.92 0.88 0.90 151 ADV 0.94 0.59 0.72 129 CONJ 1.00 0.93 0.96 42 DET 0.99 0.92 0.95 130 NOUN 0.73 0.90 0.80 479 NUM 0.85 0.68 0.75 34 PRON 0.99 0.92 0.96 194 PRT 0.89 0.88 0.88 57 VERB 0.80 0.85 0.82 362 X 0.81 0.77 0.79 183

	avg / total 0.85 0.84 0.84 2114
Result with basic and Added features on DEV data	### Dev evaluation Token-wise accuracy 85.6669820246 Token-wise F1 (macro) 84.3768537715 Token-wise F1 (micro) 85.6669820246 Sentence-wise accuracy 12.5 precision recall f1-score support . 0.96 0.99 0.97 254 ADJ 0.72 0.41 0.53 99 ADP 0.91 0.89 0.90 151 ADV 0.89 0.56 0.69 129 CONJ 1.00 0.90 0.95 42 DET 0.99 0.92 0.96 130 NOUN 0.74 0.91 0.81 479 NUM 0.85 0.68 0.75 34 PRON 0.96 0.95 0.95 194 PRT 0.91 0.91 0.91 57 VERB 0.83 0.85 0.84 362 X 0.89 0.83 0.86 183 avg / total 0.86 0.86 0.85 2114
# of features generated with basic features	Twitter pos data loaded. .. # train sents 379 .. # dev sents 112 .. # test sents 295 (7381,) -- 0 features added. -- 1000 features added. -- 2000 features added. -- 3000 features added. -- 4000 features added. -- 5000 features added. -- 6000 features added. -- 7000 features added. -- 8000 features added. -- 9000 features added.

	-- 10000 features added. -- 11000 features added. -- 12000 features added. -- 13000 features added. -- 14000 features added. Features computed (7381, 14712)
# of features generated with basic and added features	Twitter pos data loaded. .. # train sents 379 .. # dev sents 112 .. # test sents 295 (7381,) -- 0 features added. -- 1000 features added. -- 2000 features added. -- 3000 features added. -- 4000 features added. -- 5000 features added. -- 6000 features added. -- 7000 features added. -- 8000 features added. -- 9000 features added. -- 10000 features added. -- 11000 features added. -- 12000 features added. -- 13000 features added. -- 14000 features added. -- 15000 features added. -- 16000 features added. -- 17000 features added. -- 18000 features added. -- 19000 features added. -- 20000 features added. -- 21000 features added. -- 22000 features added. Features computed (7381, 22853)
./conlleval.pl -r -d \\t < ./predictions/twitter_dev.lr.pred with basic feature	processed 2114 tokens with 2114 phrases; found: 2114 phrases; correct: 1784. accuracy: 84.39%; precision: 84.39%; recall: 84.39%; FB1: 84.39 .: precision: 94.34%; recall: 98.43%; FB1: 96.34 265 ADJ: precision: 73.47%; recall: 36.36%; FB1: 48.65 49 ADP: precision: 91.72%; recall: 88.08%; FB1: 89.86 145 ADV: precision: 93.83%; recall: 58.91%; FB1: 72.38 81 CONJ: precision: 100.00%; recall: 92.86%; FB1: 96.30 39 DET: precision: 99.17%; recall: 91.54%; FB1: 95.20 120 NOUN: precision: 72.71%; recall: 89.56%; FB1: 80.26 590 NUM: precision: 85.19%; recall: 67.65%; FB1: 75.41 27 PRON: precision: 99.44%; recall: 92.27%; FB1: 95.72 180 PRT: precision: 89.29%; recall: 87.72%; FB1: 88.50 56 VERB: precision: 79.64%; recall: 85.36%; FB1: 82.40 388 X: precision: 81.03%; recall: 77.05%; FB1: 78.99 174

./conlleval.pl -r -d \\t < ./predictions/twitter_dev.lr.pred with basic and added feature	processed 2114 tokens with 2114 phrases; found: 2114 phrases; correct: 1811. accuracy: 85.67%; precision: 85.67%; recall: 85.67%; FB1: 85.67 .: precision: 95.82%; recall: 99.21%; FB1: 97.49 263 ADJ: precision: 71.93%; recall: 41.41%; FB1: 52.56 57 ADP: precision: 91.16%; recall: 88.74%; FB1: 89.93 147 ADV: precision: 88.89%; recall: 55.81%; FB1: 68.57 81 CONJ: precision: 100.00%; recall: 90.48%; FB1: 95.00 38 DET: precision: 99.17%; recall: 92.31%; FB1: 95.62 121 NOUN: precision: 73.73%; recall: 90.81%; FB1: 81.38 590 NUM: precision: 85.19%; recall: 67.65%; FB1: 75.41 27 PRON: precision: 95.83%; recall: 94.85%; FB1: 95.34 192 PRT: precision: 91.23%; recall: 91.23%; FB1: 91.23 57 VERB: precision: 83.06%; recall: 85.36%; FB1: 84.20 372 X: precision: 89.35%; recall: 82.51%; FB1: 85.80 169

CRF with Basic and Added Features	
Result with basic features on DEV data	### Dev evaluation Token-wise accuracy 84.5789971618 Token-wise F1 (macro) 83.75275036 Token-wise F1 (micro) 84.5789971618 Sentence-wise accuracy 9.82142857143 precision recall f1-score support . 0.96 0.98 0.97 254 ADJ 0.62 0.56 0.59 99 ADP 0.85 0.87 0.86 151 ADV 0.84 0.62 0.71 129 CONJ 0.98 0.95 0.96 42 DET 0.98 0.92 0.95 130 NOUN 0.79 0.86 0.82 479 NUM 0.80 0.71 0.75 34 PRON 0.97 0.94 0.96 194 PRT 0.86 0.86 0.86 57 VERB 0.80 0.83 0.81 362 X 0.81 0.80 0.80 183 avg / total 0.85 0.85 0.84 2114
Result with basic and Added features on DEV data	### Dev evaluation Token-wise accuracy 85.8088930937 Token-wise F1 (macro) 84.0995255504 Token-wise F1 (micro) 85.8088930937 Sentence-wise accuracy 9.82142857143 precision recall f1-score support

	. 0.96 0.99 0.98 254 ADJ 0.67 0.53 0.59 99 ADP 0.82 0.89 0.85 151 ADV 0.82 0.59 0.68 129 CONJ 0.93 0.95 0.94 42 DET 0.99 0.92 0.95 130 NOUN 0.79 0.87 0.83 479 NUM 0.71 0.74 0.72 34 PRON 0.94 0.95 0.95 194 PRT 0.89 0.88 0.88 57 VERB 0.86 0.85 0.85 362 X 0.87 0.85 0.86 183 avg / total 0.86 0.86 0.86 2114
# of features generated with basic features	Twitter pos data loaded. .. # train sents 379 .. # dev sents 112 .. # test sents 295 Classes: 12 ['.' 'ADJ' 'ADP' 'ADV' 'CONJ' 'DET' 'NOUN' 'NUM' 'PRON' 'PRT' 'VERB' 'X'] -- 0 features added. -- 1000 features added. -- 2000 features added. -- 3000 features added. -- 4000 features added. -- 5000 features added. -- 6000 features added. -- 7000 features added. -- 8000 features added. -- 9000 features added. -- 10000 features added. -- 11000 features added. -- 12000 features added. -- 13000 features added. -- 14000 features added. 379 14712 Number of weights 176712
# of features generated with basic and added features	Twitter pos data loaded. .. # train sents 379 .. # dev sents 112 .. # test sents 295 Classes: 12 ['.' 'ADJ' 'ADP' 'ADV' 'CONJ' 'DET' 'NOUN' 'NUM' 'PRON' 'PRT' 'VERB' 'X'] -- 0 features added. -- 1000 features added. -- 2000 features added. -- 3000 features added. -- 4000 features added. -- 5000 features added. -- 6000 features added. -- 7000 features added. -- 8000 features added.

	-- 9000 features added. -- 10000 features added. -- 11000 features added. -- 12000 features added. -- 13000 features added. -- 14000 features added. -- 15000 features added. -- 16000 features added. -- 17000 features added. -- 18000 features added. -- 19000 features added. -- 20000 features added. -- 21000 features added. -- 22000 features added. 379 22853 Number of weights 274404
./conlleval.pl -r -d \\t < ./predictions/twitter_dev.crf.pred with basic feature	processed 2114 tokens with 2114 phrases; found: 2114 phrases; correct: 1788. accuracy: 84.58%; precision: 84.58%; recall: 84.58%; FB1: 84.58 .: precision: 95.77%; recall: 98.03%; FB1: 96.89 260 ADJ: precision: 62.50%; recall: 55.56%; FB1: 58.82 88 ADP: precision: 85.16%; recall: 87.42%; FB1: 86.27 155 ADV: precision: 84.21%; recall: 62.02%; FB1: 71.43 95 CONJ: precision: 97.56%; recall: 95.24%; FB1: 96.39 41 DET: precision: 98.35%; recall: 91.54%; FB1: 94.82 121 NOUN: precision: 78.74%; recall: 85.80%; FB1: 82.12 522 NUM: precision: 80.00%; recall: 70.59%; FB1: 75.00 30 PRON: precision: 97.33%; recall: 93.81%; FB1: 95.54 187 PRT: precision: 85.96%; recall: 85.96%; FB1: 85.96 57 VERB: precision: 79.63%; recall: 83.15%; FB1: 81.35 378 X: precision: 81.11%; recall: 79.78%; FB1: 80.44 180
./conlleval.pl -r -d \\t < ./predictions/twitter_dev.crf.pred with basic and added feature	processed 2114 tokens with 2114 phrases; found: 2114 phrases; correct: 1814. accuracy: 85.81%; precision: 85.81%; recall: 85.81%; FB1: 85.81 .: precision: 96.18%; recall: 99.21%; FB1: 97.67 262 ADJ: precision: 66.67%; recall: 52.53%; FB1: 58.76 78 ADP: precision: 81.82%; recall: 89.40%; FB1: 85.44 165 ADV: precision: 81.72%; recall: 58.91%; FB1: 68.47 93 CONJ: precision: 93.02%; recall: 95.24%; FB1: 94.12 43 DET: precision: 99.17%; recall: 91.54%; FB1: 95.20 120 NOUN: precision: 79.17%; recall: 87.27%; FB1: 83.02 528 NUM: precision: 71.43%; recall: 73.53%; FB1: 72.46 35 PRON: precision: 94.36%; recall: 94.85%; FB1: 94.60 195 PRT: precision: 89.29%; recall: 87.72%; FB1: 88.50 56 VERB: precision: 85.56%; recall: 85.08%; FB1: 85.32 360 X: precision: 86.59%; recall: 84.70%; FB1: 85.64 179

8. Comparison points:

a. Which methods give the highest accuracy, and by how much?

CRF gives highest accuracy compared to LR

Details are furnished about accuracy in the table in point 7 above with DEV data and basic features (highlighted in cyan) and added features (highlighted in yellow) between LR and CRF.

b. Further, can you find/create sentences which highlight your features over the basic ones?

Yes, the details are furnished in the table in point 6 above. Newly added features in the example sentence are highlighted in yellow.

c. Are there sentences for which CRF is much better than logistic regression? Why is it better on these?

Yes, if I add below feature then CRF gives me much better result as with logistic regression.

```
# HASH length
ftrs.append("HAS HASH_" + str(hash(word.split(" ")[0])) + "_LENGTH")
```

This is because adding the above feature function helps the CRF and LR both to get 22,000 straight from 14,000, which then helps CRF to increase its accuracy.

9. Result of LR and CRF

For LR:

Gourabs-MacBook-Pro:Assignment2 gourabbhattacharyya$ python data.py --model lr --test
Twitter pos data loaded.
.. # train sents 379
.. # dev sents 112
.. # test sents 295
(7381,)
-- 0 features added.
-- 1000 features added.
-- 2000 features added.
-- 3000 features added.
-- 4000 features added.
-- 5000 features added.
-- 6000 features added.
-- 7000 features added.
-- 8000 features added.
-- 9000 features added.
-- 10000 features added.
-- 11000 features added.
-- 12000 features added.

```
-- 13000 features added.
-- 14000 features added.
-- 15000 features added.
-- 16000 features added.
-- 17000 features added.
-- 18000 features added.
-- 19000 features added.
-- 20000 features added.
-- 21000 features added.
-- 22000 features added.
Features computed
(7381, 22853)
### Train evaluation
Token-wise accuracy 99.3090367159
Token-wise F1 (macro) 99.1928146247
Token-wise F1 (micro) 99.3090367159
Sentence-wise accuracy 87.598944591
             precision    recall  f1-score   support

          .       1.00      1.00      1.00       901
        ADJ       0.99      0.97      0.98       341
        ADP       0.99      0.99      0.99       549
        ADV       0.99      0.98      0.99       401
       CONJ       0.99      0.99      0.99       161
        DET       0.99      1.00      0.99       426
       NOUN       0.99      0.99      0.99      1685
        NUM       0.99      0.98      0.99       142
       PRON       1.00      1.00      1.00       671
        PRT       1.00      1.00      1.00       207
       VERB       0.99      1.00      1.00      1215
          X       1.00      0.99      0.99       682

avg / total       0.99      0.99      0.99      7381

### Dev evaluation
Token-wise accuracy 85.6669820246
Token-wise F1 (macro) 84.3768537715
Token-wise F1 (micro) 85.6669820246
Sentence-wise accuracy 12.5
             precision    recall  f1-score   support

          .       0.96      0.99      0.97       254
        ADJ       0.72      0.41      0.53        99
        ADP       0.91      0.89      0.90       151
        ADV       0.89      0.56      0.69       129
       CONJ       1.00      0.90      0.95        42
        DET       0.99      0.92      0.96       130
       NOUN       0.74      0.91      0.81       479
        NUM       0.85      0.68      0.75        34
       PRON       0.96      0.95      0.95       194
        PRT       0.91      0.91      0.91        57
       VERB       0.83      0.85      0.84       362
          X       0.89      0.83      0.86       183
```

avg / total 0.86 0.86 0.85 2114

For CRF:

```
Gourabs-MacBook-Pro:Assignment2 gourabbhattacharyya$ python data.py --model crf --test
Twitter pos data loaded.
.. # train sents 379
.. # dev sents 112
.. # test sents 295
Classes: 12 ['.' 'ADJ' 'ADP' 'ADV' 'CONJ' 'DET' 'NOUN' 'NUM' 'PRON' 'PRT' 'VERB' 'X']
-- 0 features added.
-- 1000 features added.
-- 2000 features added.
-- 3000 features added.
-- 4000 features added.
-- 5000 features added.
-- 6000 features added.
-- 7000 features added.
-- 8000 features added.
-- 9000 features added.
-- 10000 features added.
-- 11000 features added.
-- 12000 features added.
-- 13000 features added.
-- 14000 features added.
-- 15000 features added.
-- 16000 features added.
-- 17000 features added.
-- 18000 features added.
-- 19000 features added.
-- 20000 features added.
-- 21000 features added.
-- 22000 features added.
379 22853
Number of weights 274404
Starting training
iteration 0
avg loss: 0.438152 w: [[ 1.25892541 -1.25892541 -2.51785082 ...  0.          0.
   0.        ]]
effective learning rate: 1.258925
iteration 1
avg loss: 0.240753 w: [[ 2.52990703  0.0120562  -2.51785082 ...  0.          0.
   0.        ]]
effective learning rate: 1.270982
iteration 2
avg loss: 0.176941 w: [[ 2.52990703 -1.27003265 -2.51785082 ...  0.          0.
   0.        ]]
effective learning rate: 1.282089
iteration 3
avg loss: 0.131419 w: [[ 2.52990703 -1.27003265 -1.2254586  ...  0.          0.
```

```
  0.        ]]
effective learning rate: 1.292392
iteration 4
avg loss: 0.108522 w: [[ 2.52990703 -1.27003265  0.07654685 ...  0.          0.
  0.        ]]
effective learning rate: 1.302005
iteration 5
avg loss: 0.081967 w: [[ 3.84092645  1.3520062  -2.54549199 ...  0.          0.
  0.        ]]
effective learning rate: 1.311019
iteration 6
avg loss: 0.065303 w: [[ 3.84092645  1.3520062  -2.54549199 ...  0.          0.
  0.        ]]
effective learning rate: 1.319508
iteration 7
avg loss: 0.044709 w: [[ 3.84092645  0.02447452 -2.54549199 ...  0.          0.
  0.        ]]
effective learning rate: 1.327532
iteration 8
avg loss: 0.042542 w: [[ 3.84092645  0.02447452 -1.21035063 ...  0.          0.
  0.        ]]
effective learning rate: 1.335141
iteration 9
avg loss: 0.033464 w: [[ 3.84092645  1.36685417 -1.21035063 ...  0.          0.
  0.        ]]
effective learning rate: 1.342380
iteration 10
avg loss: 0.026419 w: [[ 3.84092645  0.01757132 -1.21035063 ...  0.          0.
  0.        ]]
effective learning rate: 1.349283
iteration 11
avg loss: 0.021813 w: [[ 3.84092645  0.01757132 -1.21035063 ...  0.          0.
  0.        ]]
effective learning rate: 1.355882
iteration 12
avg loss: 0.020458 w: [[ 3.84092645  1.37977569 -2.572555   ...  0.          0.
  0.        ]]
effective learning rate: 1.362204
iteration 13
avg loss: 0.016935 w: [[ 5.20919953  0.01150261 -2.572555   ...  0.          0.
  0.        ]]
effective learning rate: 1.368273
iteration 14
avg loss: 0.016935 w: [[5.20919953 0.01150261 0.17566262 ... 0.         0.         0.        ]]
effective learning rate: 1.374109
iteration 15
avg loss: 0.012464 w: [[ 5.20919953  0.01150261 -1.20406704 ...  0.          0.
  0.        ]]
effective learning rate: 1.379730
iteration 16
avg loss: 0.013142 w: [[ 5.20919953 -1.37364908 -1.20406704 ...  0.          0.
  0.        ]]
effective learning rate: 1.385152
```

```
iteration 17
avg loss: 0.010026 w: [[ 5.20919953 -1.37364908 -1.20406704 ...  0.          0.
   0.        ]]
effective learning rate: 1.390389
iteration 18
avg loss: 0.012464 w: [[ 5.20919953 -1.37364908 -2.59952193 ...  0.          0.
   0.        ]]
effective learning rate: 1.395455
iteration 19
avg loss: 0.008535 w: [[ 5.20919953 -1.37364908 -2.59952193 ...  0.          0.
   0.        ]]
effective learning rate: 1.400360
iteration 20
avg loss: 0.013413 w: [[ 5.20919953  0.03146675 -2.59952193 ...  0.          0.
   0.        ]]
effective learning rate: 1.405116
iteration 21
avg loss: 0.009484 w: [[ 5.20919953  0.03146675 -1.1897912  ...  0.          0.
   0.        ]]
effective learning rate: 1.409731
iteration 22
avg loss: 0.005013 w: [[ 5.20919953  0.03146675 -1.1897912  ...  0.          0.
   0.        ]]
effective learning rate: 1.414214
iteration 23
avg loss: 0.004335 w: [[ 5.20919953  0.03146675 -1.1897912  ...  0.          0.
   0.        ]]
effective learning rate: 1.418572
iteration 24
avg loss: 0.005284 w: [[ 5.20919953  0.03146675 -1.1897912  ...  0.          0.
   0.        ]]
effective learning rate: 1.422813
### Train evaluation
Token-wise accuracy 99.8509687034
Token-wise F1 (macro) 99.8399627452
Token-wise F1 (micro) 99.8509687034
Sentence-wise accuracy 97.3614775726
             precision    recall  f1-score   support

          .       1.00      1.00      1.00       901
        ADJ       1.00      0.99      1.00       341
        ADP       1.00      1.00      1.00       549
        ADV       1.00      0.99      1.00       401
       CONJ       1.00      1.00      1.00       161
        DET       1.00      1.00      1.00       426
       NOUN       1.00      1.00      1.00      1685
        NUM       0.99      1.00      1.00       142
       PRON       1.00      1.00      1.00       671
        PRT       1.00      1.00      1.00       207
       VERB       1.00      1.00      1.00      1215
          X       1.00      1.00      1.00       682

avg / total       1.00      1.00      1.00      7381
```

Dev evaluation

Token-wise accuracy 85.8088930937
Token-wise F1 (macro) 84.0995255504
Token-wise F1 (micro) 85.8088930937
Sentence-wise accuracy 9.82142857143

	precision	recall	f1-score	support
.	0.96	0.99	0.98	254
ADJ	0.67	0.53	0.59	99
ADP	0.82	0.89	0.85	151
ADV	0.82	0.59	0.68	129
CONJ	0.93	0.95	0.94	42
DET	0.99	0.92	0.95	130
NOUN	0.79	0.87	0.83	479
NUM	0.71	0.74	0.72	34
PRON	0.94	0.95	0.95	194
PRT	0.89	0.88	0.88	57
VERB	0.86	0.85	0.85	362
X	0.87	0.85	0.86	183
avg / total	0.86	0.86	0.86	2114

10. Prediction for Test Data

For LR:

twitter_test.lr.pred

For CRF:

twitter_test.crf.pred