Praveen Kumar - 111986420 - CSE628 Assignment2 - Report

System configuration:

OS - MAC OS 10.13.6
Python - 2.7.15

Need num2words to run feat_gen.py file

Commands:

For CRF:

python data.py --model crf --test
./conlleval.pl -r -d \\t < ./predictions/twitter_dev.crf.pred

For memm:

python data.py --model memm --test
./conlleval.pl -r -d \\t < ./predictions/twitter_dev.lr.pred

To check feature generation file:

python feat_gen.py

Question 1) Description of your viterbi implementation:

- Added start_scores to trans_scores
- Added start_scores to emission_scores
- Two zero matrices are initialised one both with dimension's of emission_scores (one named as em_scores) and the other with type of int32 named as back_ptrs
- 0th record of em_scores is set as start_scores + 0th record of em_scores.

- For all the indices across the N tokens, i do the following

  1) trans_plus_score = trans_scores + expanded dim of scores[i-1] with row = 1
  2) back_ptrs[kth row] = row wise max value of trans_plus_score
  3) em_scores[kth row] = max value of trans_plus_score + em_scores[kth

row]

- Generate the most likely path viterbi = row wise max (end_scores + em_scores[-1])
- Viterbi score = max sum of end_scores + em_scores[-1]

- For each back_ptr element in reversed back_ptrs

 1) append(back_ptr[v[-1]])

- Finally reverse the viterbi list and return it and the viterbi score

——

Question 2) Description of the added features:

In preprocessing step - replace tab with space and also remove extra line spaces
Features addition step -

    positive_emotion = { "<3",":D", ":d", ":dd", ":P",":p","8)","8-)", ":-)", ":)", ";)", "(-:","(:",":')", "xD", "XD","yay!", "yay","yaay","yaaay", "yaaaay", "yaaaaay", "Yay!","Yay","Yaay","Yaaay", "Yaaaay", "Yaaaaay", "Hurray", "Hurraay", "Hurraaay"}
    negative_emotion = {":/",">",":'(",":-(", ":(", ":s", ":-s", "-_-","-.-"}

    emotion = "IS_NOT_AN_EMOTION"
    if word.split(" ")[0] in positive_emotion: **——> Checking if it is a positive emotion**
        emoji = "IS_POSITIVE_EMOTION"
    elif word.split(" ")[0] in negative_emotion: **——> Checking if it is a negative emotion**
        emoji = "IS_NEGATIVE_EMOTION"
    ftrs.append(emotion)

    if word.startswith("https://") :   **——> Checking if it has a secured url**
        if len(word[9:])!=0:
            ftrs.append("IS_A_SECURED_URL")
        else:
            ftrs.append("IS_NOT_A_SECURED_URL")
    elif word.startswith("http://") :   **——> Checking if it has a url**
        if len(word[8:])!=0:

```
            ftrs.append("IS_A_URL")
        else:
            ftrs.append("IS_NOT_A_URL")

    if "!"  in word :                              ——> Checking if it has an
exclamation mark
        ftrs.append("HAS_A_EXCLAIMATION_MARK")

    if word.startswith("@") :                      ——> Checking if it has a header
        if len( word[1:] ) != 0:
            ftrs.append("IS_HEADER")
        else:
            ftrs.append("IS_NOT_HEADER")

    if "?"  in word :                                                     ——>
Checking if it has a question mark
        ftrs.append("HAS_A_QUESTION_MARK")

    if word.startswith("#") :                      ——> Checking if it has a hashtag
        if  len( word[1:] )!=0:
            ftrs.append("IS_A_HASHTAG")
        else:
            ftrs.append("IS_NOT_A_HASHTAG")

    w = word.split(" ")[0]

    if w.endswith("ed") :                          ——> Checking if the word is
ending with a particular suffix
        ftrs.append("ENDS_WITH_ED")
    elif w.endswith("ing") :
        ftrs.append("ENDS_WITH_ING")
    elif w.endswith("s") :
        ftrs.append("ENDS_WITH_S")
    elif w.endswith("es") :
        ftrs.append("ENDS_WITH_ES")
    elif w.endswith("ous") :
        ftrs.append("ENDS_WITH_OUS")

    elif w.endswith("able") :
        ftrs.append("ENDS_WITH_ABLE")
    elif w.endswith("al") :
        ftrs.append("ENDS_WITH_AL")
    elif w.endswith("an") :
        ftrs.append("ENDS_WITH_AN")
    elif w.endswith("ar") :
        ftrs.append("ENDS_WITH_AR")

    elif w.endswith("ent") :
        ftrs.append("ENDS_WITH_ENT")
    elif w.endswith("ful") :
        ftrs.append("ENDS_WITH_FUL")
```

```
elif w.endswith("ic") :
    ftrs.append("ENDS_WITH_IC")
elif w.endswith("ical") :
    ftrs.append("ENDS_WITH_ICAL")

elif w.endswith("ine") :
    ftrs.append("ENDS_WITH_INE")
elif w.endswith("ile") :
    ftrs.append("ENDS_WITH_ILE")
elif w.endswith("ive") :
    ftrs.append("ENDS_WITH_IVE")
elif w.endswith("less") :
    ftrs.append("ENDS_WITH_LESS")

elif w.endswith("ous") :
    ftrs.append("ENDS_WITH_OUS")
elif w.endswith("some") :
    ftrs.append("ENDS_WITH_SOME")
elif w.endswith("ty") :
    ftrs.append("ENDS_WITH_TY")
elif w.endswith("ly") :
    ftrs.append("ENDS_WITH_LY")

elif w.endswith("ie") :
    ftrs.append("ENDS_WITH_IE")
elif w.endswith("or") :
    ftrs.append("ENDS_WITH_OR")
elif w.endswith("ance") :
    ftrs.append("ENDS_WITH_ANCE")
elif w.endswith("ish") :
    ftrs.append("ENDS_WITH_ISH")

elif w.endswith("ion") :
    ftrs.append("ENDS_WITH_ION")
elif w.endswith("ce") :
    ftrs.append("ENDS_WITH_CE")
elif w.endswith("ge") :
    ftrs.append("ENDS_WITH_GE")
elif w.endswith("ite") :
    ftrs.append("ENDS_WITH_ITE")

elif w.endswith("acy") :
    ftrs.append("ENDS_WITH_ACY")
elif w.endswith("asy") :
    ftrs.append("ENDS_WITH_ASY")
elif w.endswith("ize") :
    ftrs.append("ENDS_WITH_IZE")

elif w.endswith("ise") :
    ftrs.append("ENDS_WITH_ISE")
elif w.endswith("yze") :
    ftrs.append("ENDS_WITH_YZE")
```

```
    elif w.endswith("yse") :
        ftrs.append("ENDS_WITH_YSE")
    elif w.endswith("ance") :
        ftrs.append("ENDS_WITH_ANCE")

    elif w.endswith("ence") :
        ftrs.append("ENDS_WITH_ENCE")
    elif w.endswith("ancy") :
        ftrs.append("ENDS_WITH_ANCY")
    elif w.endswith("ency") :
        ftrs.append("ENDS_WITH_ENCY")
    elif w.endswith("ant") :
        ftrs.append("ENDS_WITH_ANT")

    elif w.endswith("ent") :
        ftrs.append("ENDS_WITH_ENT")
    elif w.endswith("ary") :
        ftrs.append("ENDS_WITH_ARY")
    elif w.endswith("ery") :
        ftrs.append("ENDS_WITH_ERY")
    elif w.endswith("ory") :
        ftrs.append("ENDS_WITH_ORY")
    elif w.endswith("y") :
        ftrs.append("ENDS_WITH_Y")

    elif w.endswith("ogue") :
        ftrs.append("ENDS_WITH_OGUE")
    elif w.endswith("og") :
        ftrs.append("ENDS_WITH_OG")
    elif w.endswith("oe") :
        ftrs.append("ENDS_WITH_OE")
    elif w.endswith("ae") :
        ftrs.append("ENDS_WITH_AE")

    elif w.endswith("ence") :
        ftrs.append("ENDS_WITH_ENCE")
    elif w.endswith("ense") :
        ftrs.append("ENDS_WITH_ENSE")
    elif w.endswith("efy") :
        ftrs.append("ENDS_WITH_EFY")
    elif w.endswith("ify") :
        ftrs.append("ENDS_WITH_IFY")
    elif w.endswith("y") :
        ftrs.append("ENDS_WITH_Y")


    ftrs.append("WORD LENGTH :"+str(len(word)-1))
```

——> Checking the word length

```
    ftrs.append("HASH LENGTH :" + str(hash(word.split(" ")[0])))
```

——> Checking the hash length

```
    ftrs.append("BYTE LENGTH OF WORD : " +
num2words(sys.getsizeof(word)).upper())
```

——> Checking byte length

———

Question 3 : Comparison of your features against the basic features

Highlighting the difference in basic vs my features for sents =
[
["I PRON",
"love ADJ",
"and CONJ",
":D ADJ",
"https://www.youtube.com/ X",
"Yay! ADJ",
"http://xyz.com X",
"@abc X",
"#xyz X"]
]

Highlighted the difference by highlighting with red colour

Basic features -

I PRON : ['BIAS', 'SENT_BEGIN', u'WORD=I PRON', u'LCASE=i pron', 'IS_UPPER', 'NEXT_BIAS', u'NEXT_WORD=love ADJ', u'NEXT_LCASE=love adj']

love ADJ : ['BIAS', u'WORD=love ADJ', u'LCASE=love adj', 'PREV_BIAS', 'PREV_SENT_BEGIN', u'PREV_WORD=I PRON', u'PREV_LCASE=i pron', 'PREV_IS_UPPER', 'NEXT_BIAS', u'NEXT_WORD=and CONJ', u'NEXT_LCASE=and conj']

and CONJ : ['BIAS', u'WORD=and CONJ', u'LCASE=and conj', 'PREV_BIAS', u'PREV_WORD=love ADJ', u'PREV_LCASE=love adj', 'NEXT_BIAS', u'NEXT_WORD=:D ADJ', u'NEXT_LCASE=:d adj', 'NEXT_IS_UPPER']

:D ADJ : ['BIAS', u'WORD=:D ADJ', u'LCASE=:d adj', 'IS_UPPER', 'PREV_BIAS', u'PREV_WORD=and CONJ', u'PREV_LCASE=and conj', 'NEXT_BIAS', u'NEXT_WORD=https://www.youtube.com/ X', u'NEXT_LCASE=https://www.youtube.com/ x']

https://www.youtube.com/ X : ['BIAS', u'WORD=https://www.youtube.com/ X', u'LCASE=https://www.youtube.com/ x', 'PREV_BIAS', u'PREV_WORD=:D ADJ', u'PREV_LCASE=:d adj', 'PREV_IS_UPPER', 'NEXT_BIAS', u'NEXT_WORD=Yay! ADJ', u'NEXT_LCASE=yay! adj']

Yay! ADJ : ['BIAS', u'WORD=Yay! ADJ', u'LCASE=yay! adj', 'PREV_BIAS', u'PREV_WORD=https://www.youtube.com/ X', u'PREV_LCASE=https://www.youtube.com/ x', 'NEXT_BIAS', u'NEXT_WORD=http://xyz.com X', u'NEXT_LCASE=http://xyz.com x']

http://xyz.com X : ['BIAS', u'WORD=http://xyz.com X', u'LCASE=http://xyz.com x', 'PREV_BIAS', u'PREV_WORD=Yay! ADJ', u'PREV_LCASE=yay! adj', 'NEXT_BIAS', u'NEXT_WORD=@abc X', u'NEXT_LCASE=@abc x']

@abc X : ['BIAS', u'WORD=@abc X', u'LCASE=@abc x', 'PREV_BIAS', u'PREV_WORD=http://xyz.com X', u'PREV_LCASE=http://xyz.com x', 'NEXT_BIAS', 'NEXT_SENT_END', u'NEXT_WORD=#xyz X', u'NEXT_LCASE=#xyz x']

#xyz X : ['BIAS', 'SENT_END', u'WORD=#xyz X', u'LCASE=#xyz x', 'PREV_BIAS', u'PREV_WORD=@abc X', u'PREV_LCASE=@abc x']

My features :

I PRON : ['BIAS', 'SENT_BEGIN', u'WORD=I PRON', u'LCASE=i pron', 'IS_UPPER', 'IS_NOT_AN_EMOTION', 'WORD LENGTH :5', 'HASH LENGTH :9344028104', u'BYTE LENGTH OF WORD : SIXTY-TWO', 'NEXT_BIAS', u'NEXT_WORD=love ADJ', u'NEXT_LCASE=love adj', 'NEXT_IS_NOT_AN_EMOTION', 'NEXT_WORD LENGTH :7', 'NEXT_HASH LENGTH :5971043325596305668', u'NEXT_BYTE LENGTH OF WORD : SIXTY-SIX']

love ADJ : ['BIAS', u'WORD=love ADJ', u'LCASE=love adj', 'IS_NOT_AN_EMOTION', 'WORD LENGTH :7', 'HASH LENGTH :5971043325596305668', u'BYTE LENGTH OF WORD : SIXTY-SIX', 'PREV_BIAS', 'PREV_SENT_BEGIN', u'PREV_WORD=I PRON', u'PREV_LCASE=i pron', 'PREV_IS_UPPER', 'PREV_IS_NOT_AN_EMOTION', 'PREV_WORD LENGTH :5', 'PREV_HASH LENGTH :9344028104', u'PREV_BYTE LENGTH OF WORD : SIXTY-TWO', 'NEXT_BIAS', u'NEXT_WORD=and CONJ', u'NEXT_LCASE=and conj', 'NEXT_IS_NOT_AN_EMOTION', 'NEXT_WORD LENGTH : 7', 'NEXT_HASH LENGTH :1453079729200098176', u'NEXT_BYTE LENGTH OF WORD : SIXTY-SIX']

and CONJ : ['BIAS', u'WORD=and CONJ', u'LCASE=and conj', 'IS_NOT_AN_EMOTION', 'WORD LENGTH :7', 'HASH LENGTH : 1453079729200098176', u'BYTE LENGTH OF WORD : SIXTY-SIX', 'PREV_BIAS', u'PREV_WORD=love ADJ', u'PREV_LCASE=love adj', 'PREV_IS_NOT_AN_EMOTION', 'PREV_WORD LENGTH :7', 'PREV_HASH LENGTH :5971043325596305668', u'PREV_BYTE LENGTH OF WORD : SIXTY-SIX', 'NEXT_BIAS', u'NEXT_WORD=:D

ADJ', u'NEXT_LCASE=:d adj', 'NEXT_IS_UPPER', 'NEXT_IS_NOT_AN_EMOTION', 'NEXT_WORD LENGTH :5', 'NEXT_HASH LENGTH :7424044602067048', u'NEXT_BYTE LENGTH OF WORD : SIXTY-TWO']

:D ADJ : ['BIAS', u'WORD=:D ADJ', u'LCASE=:d adj', 'IS_UPPER', 'IS_NOT_AN_EMOTION', 'WORD LENGTH :5', 'HASH LENGTH :7424044602067048', u'BYTE LENGTH OF WORD : SIXTY-TWO', 'PREV_BIAS', u'PREV_WORD=and CONJ', u'PREV_LCASE=and conj', 'PREV_IS_NOT_AN_EMOTION', 'PREV_WORD LENGTH : 7', 'PREV_HASH LENGTH :1453079729200098176', u'PREV_BYTE LENGTH OF WORD : SIXTY-SIX', 'NEXT_BIAS', u'NEXT_WORD=https://www.youtube.com/ X', u'NEXT_LCASE=https://www.youtube.com/ x', 'NEXT_IS_NOT_AN_EMOTION', 'NEXT_IS_A_SECURED_URL', 'NEXT_WORD LENGTH :25', 'NEXT_HASH LENGTH :-1422413245927943911', u'NEXT_BYTE LENGTH OF WORD : ONE HUNDRED AND TWO']

https://www.youtube.com/ X : ['BIAS', u'WORD=https://www.youtube.com/ X', u'LCASE=https://www.youtube.com/ x', 'IS_NOT_AN_EMOTION', 'IS_A_SECURED_URL', 'WORD LENGTH :25', 'HASH LENGTH :-1422413245927943911', u'BYTE LENGTH OF WORD : ONE HUNDRED AND TWO', 'PREV_BIAS', u'PREV_WORD=:D ADJ', u'PREV_LCASE=:d adj', 'PREV_IS_UPPER', 'PREV_IS_NOT_AN_EMOTION', 'PREV_WORD LENGTH :5', 'PREV_HASH LENGTH :7424044602067048', u'PREV_BYTE LENGTH OF WORD : SIXTY-TWO', 'NEXT_BIAS', u'NEXT_WORD=Yay! ADJ', u'NEXT_LCASE=yay! adj', 'NEXT_IS_NOT_AN_EMOTION', 'NEXT_HAS_A_EXCLAIMATION_MARK', 'NEXT_WORD LENGTH :7', 'NEXT_HASH LENGTH :821293140639314160', u'NEXT_BYTE LENGTH OF WORD : SIXTY-SIX']

Yay! ADJ : ['BIAS', u'WORD=Yay! ADJ', u'LCASE=yay! adj', 'IS_NOT_AN_EMOTION', 'HAS_A_EXCLAIMATION_MARK', 'WORD LENGTH :7', 'HASH LENGTH : 821293140639314160', u'BYTE LENGTH OF WORD : SIXTY-SIX', 'PREV_BIAS', u'PREV_WORD=https://www.youtube.com/ X', u'PREV_LCASE=https://www.youtube.com/ x', 'PREV_IS_NOT_AN_EMOTION', 'PREV_IS_A_SECURED_URL', 'PREV_WORD LENGTH :25', 'PREV_HASH LENGTH :-1422413245927943911', u'PREV_BYTE LENGTH OF WORD : ONE HUNDRED AND TWO', 'NEXT_BIAS', u'NEXT_WORD=http://xyz.com X', u'NEXT_LCASE=http://xyz.com x', 'NEXT_IS_NOT_AN_EMOTION', 'NEXT_IS_A_URL', 'NEXT_WORD LENGTH :15', 'NEXT_HASH LENGTH :-5750080177430901690', u'NEXT_BYTE LENGTH OF WORD : EIGHTY-TWO']

http://xyz.com X : ['BIAS', u'WORD=http://xyz.com X', u'LCASE=http://xyz.com x', 'IS_NOT_AN_EMOTION', 'IS_A_URL', 'WORD LENGTH :15', 'HASH LENGTH :-5750080177430901690', u'BYTE LENGTH OF WORD : EIGHTY-TWO', 'PREV_BIAS', u'PREV_WORD=Yay! ADJ', u'PREV_LCASE=yay! adj', 'PREV_IS_NOT_AN_EMOTION', 'PREV_HAS_A_EXCLAIMATION_MARK', 'PREV_WORD LENGTH :7', 'PREV_HASH LENGTH :821293140639314160', u'PREV_BYTE LENGTH OF WORD : SIXTY-SIX', 'NEXT_BIAS', u'NEXT_WORD=@abc X', u'NEXT_LCASE=@abc x', 'NEXT_IS_NOT_AN_EMOTION', 'NEXT_IS_HEADER', 'NEXT_WORD LENGTH :5', 'NEXT_HASH LENGTH :7637656320375785316', u'NEXT_BYTE LENGTH OF WORD : SIXTY-TWO']

@abc X : ['BIAS', u'WORD=@abc X', u'LCASE=@abc x', 'IS_NOT_AN_EMOTION', 'IS_HEADER', 'WORD LENGTH :5', 'HASH LENGTH :7637656320375785316', u'BYTE LENGTH OF WORD : SIXTY-TWO', 'PREV_BIAS', u'PREV_WORD=http://xyz.com X', u'PREV_LCASE=http://xyz.com x', 'PREV_IS_NOT_AN_EMOTION', 'PREV_IS_A_URL', 'PREV_WORD LENGTH :15', 'PREV_HASH LENGTH :-5750080177430901690', u'PREV_BYTE LENGTH OF WORD : EIGHTY-TWO', 'NEXT_BIAS', 'NEXT_SENT_END', u'NEXT_WORD=#xyz X', u'NEXT_LCASE=#xyz x', 'NEXT_IS_NOT_AN_EMOTION', 'NEXT_IS_A_HASHTAG', 'NEXT_WORD LENGTH :5', 'NEXT_HASH LENGTH :-2164168021658837856', u'NEXT_BYTE LENGTH OF WORD : SIXTY-TWO']

#xyz X : ['BIAS', 'SENT_END', u'WORD=#xyz X', u'LCASE=#xyz x', 'IS_NOT_AN_EMOTION', 'IS_A_HASHTAG', 'WORD LENGTH :5', 'HASH LENGTH :-2164168021658837856', u'BYTE LENGTH OF WORD : SIXTY-TWO', 'PREV_BIAS', u'PREV_WORD=@abc X', u'PREV_LCASE=@abc x', 'PREV_IS_NOT_AN_EMOTION', 'PREV_IS_HEADER', 'PREV_WORD LENGTH :5', 'PREV_HASH LENGTH :7637656320375785316', u'PREV_BYTE LENGTH OF WORD : SIXTY-TWO']

———

Question 4: **Comparison of MEMM and CRFs**

CRF on dev data with basic features:

python data.py --model crf
Twitter pos data loaded.
.. # train sents 379
.. # dev sents 112
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
Starting training
iteration 0
avg loss: 0.429752 w: [[ 0. -1.  0. ...  0.  0.  0.]]
effective learning rate: 1.000000
iteration 1
avg loss: 0.225986 w: [[ 0. -1.  0. ...  0.  0.  0.]]
effective learning rate: 1.000000
iteration 2
avg loss: 0.154722 w: [[ 0. -1.  0. ...  0.  0.  0.]]
effective learning rate: 1.000000
iteration 3
avg loss: 0.109606 w: [[0. 1. 0. ... 0. 0. 0.]]
effective learning rate: 1.000000
iteration 4
avg loss: 0.069096 w: [[ 0.  1. -1. ...  0.  0.  0.]]
effective learning rate: 1.000000
iteration 5
avg loss: 0.050942 w: [[ 1.  0.  0. ... -1.  0.  0.]]
effective learning rate: 1.000000
iteration 6
avg loss: 0.036174 w: [[ 1.  0.  0. ... -1.  0.  0.]]
effective learning rate: 1.000000
iteration 7
avg loss: 0.026419 w: [[ 1.  1.  1. ... -1.  0.  0.]]
effective learning rate: 1.000000
iteration 8
avg loss: 0.029264 w: [[ 1.  0.  0. ... -1.  0.  0.]]
effective learning rate: 1.000000
iteration 9
avg loss: 0.020593 w: [[ 1.  1.  0. ... -1.  0.  0.]]
effective learning rate: 1.000000
iteration 10
avg loss: 0.014632 w: [[ 1.  0.  0. ... -1.  0.  0.]]
effective learning rate: 1.000000
iteration 11
avg loss: 0.015581 w: [[ 1.  0.  0. ... -1.  0.  0.]]
effective learning rate: 1.000000
iteration 12
avg loss: 0.014361 w: [[ 1.  0.  0. ... -1.  0.  0.]]
effective learning rate: 1.000000
iteration 13
avg loss: 0.006639 w: [[ 2.  0.  0. ... -1.  0.  0.]]
effective learning rate: 1.000000
iteration 14
avg loss: 0.007993 w: [[ 2.  0. -1. ... -1.  0.  0.]]
effective learning rate: 1.000000
iteration 15
```

avg loss: 0.010839 w: [[2. -1. 0. ... -1. 0. 0.]]
effective learning rate: 1.000000
iteration 16
avg loss: 0.007452 w: [[2. -1. 0. ... -1. 0. 0.]]
effective learning rate: 1.000000
iteration 17
avg loss: 0.007181 w: [[2. -1. 0. ... -1. 0. 0.]]
effective learning rate: 1.000000
Train evaluation
Token-wise accuracy 99.71548570654383
Token-wise F1 (macro) 99.71914040429829
Token-wise F1 (micro) 99.71548570654383
Sentence-wise accuracy 94.72295514511873

	precision	recall	f1-score	support
.	1.00	1.00	1.00	901
ADJ	1.00	0.99	0.99	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
micro avg	1.00	1.00	1.00	7381
macro avg	1.00	1.00	1.00	7381
weighted avg	1.00	1.00	1.00	7381

Dev evaluation
Token-wise accuracy **84.15326395458845**
Token-wise F1 (macro) 83.36271018265512
Token-wise F1 (micro) 84.15326395458845
Sentence-wise accuracy 10.714285714285714

	precision	recall	f1-score	support
.	0.95	0.98	0.97	254
ADJ	0.68	0.48	0.56	99
ADP	0.89	0.89	0.89	151
ADV	0.85	0.60	0.70	129
CONJ	0.95	0.95	0.95	42
DET	0.98	0.92	0.95	130
NOUN	0.78	0.85	0.81	479
NUM	0.81	0.74	0.77	34
PRON	0.96	0.94	0.95	194
PRT	0.86	0.86	0.86	57
VERB	0.77	0.84	0.80	362
X	0.80	0.78	0.79	183
micro avg	0.84	0.84	0.84	2114
macro avg	0.86	0.82	0.83	2114

```
weighted avg       0.84      0.84      0.84      2114

(venv) Praveens-MacBook-Pro:Assignment2_for_students praveenkumar$
./conlleval.pl -r -d \\t < ./predictions/twitter_dev.crf.pred
processed 2114 tokens with 2114 phrases; found: 2114 phrases; correct: 1779.
accuracy:  84.15%; precision:  84.15%; recall:  84.15%; FB1:  84.15
                .: precision:  94.70%; recall:  98.43%; FB1:  96.53  264
              ADJ: precision:  67.61%; recall:  48.48%; FB1:  56.47  71
              ADP: precision:  88.74%; recall:  88.74%; FB1:  88.74  151
              ADV: precision:  84.62%; recall:  59.69%; FB1:  70.00  91
             CONJ: precision:  95.24%; recall:  95.24%; FB1:  95.24  42
              DET: precision:  98.35%; recall:  91.54%; FB1:  94.82  121
             NOUN: precision:  77.76%; recall:  85.39%; FB1:  81.39  526
              NUM: precision:  80.65%; recall:  73.53%; FB1:  76.92  31
             PRON: precision:  95.79%; recall:  93.81%; FB1:  94.79  190
              PRT: precision:  85.96%; recall:  85.96%; FB1:  85.96  57
             VERB: precision:  77.49%; recall:  83.70%; FB1:  80.48  391
                X: precision:  79.89%; recall:  78.14%; FB1:  79.01  179
(venv) Praveens-MacBook-Pro:Assignment2_for_students praveenkumar$
```

CRF on dev data with basic+advanced features:

```
python data.py --model crf
Twitter pos data loaded.
.. # train sents 379
.. # dev sents 112
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 22865
Number of weights 274548
Starting training
iteration 0
avg loss: 0.390597 w: [[ 0. -1. -1. ...  0.  0.  0.]]
effective learning rate: 1.000000
iteration 1
avg loss: 0.209050 w: [[ 1. -1.  0. ...  0.  0.  0.]]
effective learning rate: 1.000000
iteration 2
avg loss: 0.132638 w: [[ 1. -1. -2. ...  0.  0.  0.]]
effective learning rate: 1.000000
iteration 3
avg loss: 0.098903 w: [[ 1. -1. -1. ...  0.  0.  0.]]
effective learning rate: 1.000000
iteration 4
avg loss: 0.075464 w: [[ 1. -1. -2. ...  0.  0.  0.]]
effective learning rate: 1.000000
iteration 5
avg loss: 0.058664 w: [[ 2.  2. -1. ...  0.  0.  0.]]
effective learning rate: 1.000000
iteration 6
avg loss: 0.039832 w: [[ 2.  1. -1. ...  0.  0.  0.]]
effective learning rate: 1.000000
iteration 7
avg loss: 0.041729 w: [[ 2.  2. -2. ...  0.  0.  0.]]
effective learning rate: 1.000000
iteration 8
avg loss: 0.028858 w: [[ 2.  0. -2. ...  0.  0.  0.]]
effective learning rate: 1.000000
iteration 9
avg loss: 0.022084 w: [[ 2.  0. -2. ...  0.  0.  0.]]
effective learning rate: 1.000000
iteration 10
avg loss: 0.017342 w: [[ 2.  0. -2. ...  0.  0.  0.]]
effective learning rate: 1.000000
iteration 11
avg loss: 0.015174 w: [[ 2. -1. -2. ...  0.  0.  0.]]
effective learning rate: 1.000000
iteration 12
avg loss: 0.014361 w: [[ 2.  0. -1. ...  0.  0.  0.]]
effective learning rate: 1.000000
iteration 13
avg loss: 0.011110 w: [[3. 0. 0. ... 0. 0. 0.]]
effective learning rate: 1.000000
iteration 14
avg loss: 0.009077 w: [[ 3.  0. -1. ...  0.  0.  0.]]
effective learning rate: 1.000000
iteration 15
avg loss: 0.010703 w: [[ 3.  0. -1. ...  0.  0.  0.]]
effective learning rate: 1.000000
iteration 16
avg loss: 0.007587 w: [[ 3.  0. -1. ...  0.  0.  0.]]
effective learning rate: 1.000000
```

iteration 17
avg loss: 0.007858 w: [[3. 0. -1. ... 0. 0. 0.]]
effective learning rate: 1.000000
Train evaluation
Token-wise accuracy 99.78322720498576
Token-wise F1 (macro) 99.79042005534997
Token-wise F1 (micro) 99.78322720498576
Sentence-wise accuracy 95.77836411609498

	precision	recall	f1-score	support
.	1.00	1.00	1.00	901
ADJ	1.00	0.99	0.99	341
ADP	1.00	1.00	1.00	549
ADV	1.00	1.00	1.00	401
CONJ	1.00	1.00	1.00	161
DET	1.00	1.00	1.00	426
NOUN	0.99	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
micro avg	1.00	1.00	1.00	7381
macro avg	1.00	1.00	1.00	7381
weighted avg	1.00	1.00	1.00	7381

Dev evaluation
Token-wise accuracy **85.99810785241249**
Token-wise F1 (macro) 84.63228749432923
Token-wise F1 (micro) 85.99810785241247
Sentence-wise accuracy 14.285714285714285

	precision	recall	f1-score	support
.	0.96	0.98	0.97	254
ADJ	0.67	0.52	0.58	99
ADP	0.86	0.89	0.88	151
ADV	0.82	0.71	0.76	129
CONJ	0.95	0.95	0.95	42
DET	0.98	0.91	0.94	130
NOUN	0.79	0.88	0.83	479
NUM	0.79	0.68	0.73	34
PRON	0.95	0.93	0.94	194
PRT	0.89	0.88	0.88	57
VERB	0.85	0.85	0.85	362
X	0.84	0.83	0.83	183
micro avg	0.86	0.86	0.86	2114
macro avg	0.86	0.83	0.85	2114
weighted avg	0.86	0.86	0.86	2114

Memm on dev data with basic features:

```
python data.py --model memm
Twitter pos data loaded.
.. # train sents 379
.. # dev sents 112
(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)
/Users/praveenkumar/Desktop/Sem 1/NLP/venv/lib/python2.7/site-packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/praveenkumar/Desktop/Sem 1/NLP/venv/lib/python2.7/site-packages/sklearn/linear_model/logistic.py:459: FutureWarning: Default multi_class will be changed to 'auto' in 0.22. Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
### Train evaluation
Token-wise accuracy 98.19807614144425
Token-wise F1 (macro) 97.74689850492388
Token-wise F1 (micro) 98.19807614144425
Sentence-wise accuracy 71.50395778364116
```

	precision	recall	f1-score	support
.	0.99	1.00	1.00	901
ADJ	0.99	0.92	0.95	341
ADP	0.98	0.98	0.98	549
ADV	0.98	0.96	0.97	401
CONJ	0.99	0.98	0.98	161
DET	0.98	1.00	0.99	426
NOUN	0.98	0.98	0.98	1685
NUM	0.97	0.91	0.94	142
PRON	1.00	0.99	1.00	671
PRT	0.99	0.99	0.99	207

```
       VERB       0.98      0.99      0.99      1215
          X       0.97      0.98      0.97       682

   micro avg       0.98      0.98      0.98      7381
   macro avg       0.98      0.97      0.98      7381
weighted avg       0.98      0.98      0.98      7381

### Dev evaluation
Token-wise accuracy 84.38978240302744
Token-wise F1 (macro) 83.33422799705717
Token-wise F1 (micro) 84.38978240302745
Sentence-wise accuracy 8.928571428571429
             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

   micro avg       0.84      0.84      0.84      2114
   macro avg       0.88      0.80      0.83      2114
weighted avg       0.85      0.84      0.84      2114

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

Memm on dev data with basic + my features:

```
python data.py --model memm
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, 22865)
/Users/praveenkumar/Desktop/Sem 1/NLP/venv/lib/python2.7/site-packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/praveenkumar/Desktop/Sem 1/NLP/venv/lib/python2.7/site-packages/sklearn/linear_model/logistic.py:459: FutureWarning: Default multi_class will be changed to 'auto' in 0.22. Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
### Train evaluation
Token-wise accuracy 99.26839181682698
Token-wise F1 (macro) 99.14139182921068
Token-wise F1 (micro) 99.26839181682698
Sentence-wise accuracy 86.80738786279683
             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      0.99      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       1.00      1.00      1.00      1215
          X       1.00      0.99      0.99       682

   micro avg       0.99      0.99      0.99      7381
   macro avg       0.99      0.99      0.99      7381
weighted avg       0.99      0.99      0.99      7381
```

Dev evaluation

```
Token-wise accuracy 86.9441816461684
Token-wise F1 (macro) 85.46539859487362
Token-wise F1 (micro) 86.9441816461684
Sentence-wise accuracy 16.071428571428573
              precision    recall  f1-score   support

          .       0.96      0.99      0.97       254
        ADJ       0.74      0.39      0.51        99
        ADP       0.91      0.89      0.90       151
        ADV       0.89      0.71      0.79       129
       CONJ       1.00      0.90      0.95        42
        DET       0.99      0.92      0.95       130
       NOUN       0.76      0.91      0.83       479
        NUM       0.82      0.68      0.74        34
       PRON       0.98      0.95      0.96       194
        PRT       0.93      0.91      0.92        57
       VERB       0.84      0.87      0.86       362
          X       0.89      0.84      0.86       183

   micro avg       0.87      0.87      0.87      2114
   macro avg       0.89      0.83      0.85      2114
weighted avg       0.87      0.87      0.87      2114
```

Generating Test predictions

```
(venv) Praveens-MacBook-Pro:Assignment2_for_students praveenkumar$

./conlleval.pl -r -d \\t < ./predictions/twitter_dev.lr.pred
processed 2114 tokens with 2114 phrases; found: 2114 phrases; correct: 1838.
accuracy:  86.94%; precision:  86.94%; recall:  86.94%; FB1:  86.94
                .: precision:  95.45%; recall:  99.21%; FB1:  97.30  264
              ADJ: precision:  74.07%; recall:  40.40%; FB1:  52.29  54
              ADP: precision:  90.54%; recall:  88.74%; FB1:  89.63  148
              ADV: precision:  89.22%; recall:  70.54%; FB1:  78.79  102
             CONJ: precision: 100.00%; recall:  90.48%; FB1:  95.00  38
              DET: precision:  99.17%; recall:  91.54%; FB1:  95.20  120
             NOUN: precision:  76.49%; recall:  91.02%; FB1:  83.13  570
              NUM: precision:  82.14%; recall:  67.65%; FB1:  74.19  28
             PRON: precision:  97.87%; recall:  94.85%; FB1:  96.34  188
              PRT: precision:  92.86%; recall:  91.23%; FB1:  92.04  56
             VERB: precision:  84.27%; recall:  87.29%; FB1:  85.75  375
                X: precision:  89.47%; recall:  83.61%; FB1:  86.44  171
```

Comparison points:

1)Which methods give the highest accuracy, and by how much?

Memm gives better accuracy than CRF, memm gave **86.9441816461684** while crf gives **85.99810785241249** (with basic + my features)

and (even with basic features) Memm gives better accuracy than CRF, memm gave **84.38978240302744** while crf gives **84.15326395458845**

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

Yes, sents = [

```
  [ "I PRON",
   "love ADJ",
   "and CONJ",
   ":D ADJ",
   "https://www.youtube.com/ X",
   "Yay! ADJ",
   "http://xyz.com X",
   "@abc X",
   "#xyz X"]
]
```

Please check my answer for 3rd question in which i elaborated this. Highlighted the difference by highlighting with red colour.

3) Are there sentences for which CRF is much better than MEMM? Why is it better on these? Use any graphs, tables, and figures to aid your analysis, including ones generated by conlleval.pl.

Yes, Hash function helps to increase accuracy of CRF over memm. This below line :

```
ftrs.append("HASH LENGTH :" + str(hash(word.split(" ")[0])))
```

Reason - Features added would be increased from 14000 to 220000 which helps CRF in increasing accuracy