HW1 NLP WET- REPORT

Yonathan Mizrahi – 209948728 & Raphaël Bellahsen – 931188684

1. Task division

Raphaël worked on preprocessing.py and implemented all the relevant features. Yonathan implemented the Viterbi algorithm. The training and the testing stages were done by both of us.

2. Train

A. List of the Features

Please find below the list of all the features we implemented in our models:

Type of features	***Number of features***	***Model***
$f_{100}(x,y) = \begin{cases} 1 & if\ current\ word\ is\ x\ and\ tag\ is\ y \\ 0 & otherwise \end{cases}$	Model 1: 15,415 Model 2: 1,520	1,2
$f_{101}, f_{102}(x,y) = \begin{cases} 1 & if\ prefix/suffix\ is\ x\ and\ tag\ is\ y \\ 0 & otherwise \end{cases}$ $1 \leq prefix, suffix \leq 4$	F101: Model 1: 13,265 Model 2: 2,258 F102: Model 1: 22,393 Model 2: 3,452	1,2
$f_{103}(x,y,z) = \begin{cases} 1 & if < t_{-2}, t_{-1}, t > = < x, y, z > \\ 0 & otherwise \end{cases}$	Model 1: 8,150 Model 2: 1,042	1,2
$f_{104}(x,y) = \begin{cases} 1 & if < t_{-1}, t > = < x, y > \\ 0 & otherwise \end{cases}$	Model 1: 1,060 Model 2: 290	1,2
$f_{105}(x) = \begin{cases} 1 & if < t > = < x > \\ 0 & otherwise \end{cases}$	Model 1: 44 Model 2: 30	1,2
$f_{106}(x,y) = \begin{cases} 1 & if\ previous\ word\ is\ x\ and\ tag\ is\ y \\ 0 & otherwise \end{cases}$	Model 1: 32,132 Model 2: 2,278	1,2
$f_{107}(x,y) = \begin{cases} 1 & if\ next\ word\ is\ x\ and\ tag\ is\ y \\ 0 & otherwise \end{cases}$	Model 1: 30,793 Model 2: 2,190	1,2
$f_{NumInside}(x,y) = \begin{cases} 1 & if\ word\ x\ contains\ a\ digit\ tag\ is\ y \\ 0 & otherwise \end{cases}$	Model 1: 1,248 Model 2: 141	1,2
$f_{CapitalInside}(x,y) = \begin{cases} 1 & if\ capital\ letter\ in\ x\ and\ tag\ is\ y \\ 0 & otherwise \end{cases}$	Model 1: 445 Model 2: 191	1,2

$f_{Num}(x,y) = \begin{cases} 1 & if\ word\ x\ start\ with\ a\ digit\ tag\ is\ y \\ 0 & otherwise \end{cases}$	Model 1: 1,218 Model 2: 33	1,2
$f_{Capital}(x,y) = \begin{cases} 1 & if\ capital\ letter\ in\ x[0]\ and\ tag\ is\ y \\ 0 & otherwise \end{cases}$	Model 1: 3,929 Model 2: 326	1,2

$f_{Composed}(x,y) = \{1\ if\ word\ x\ contains\ '-'\ and\ the\ tag\ is\ y\}$	Model 1: 1018 Model 2 : 180	1,2
$f_{suffixPrevWord}, f_{prefixPrevword}(x,y) = \begin{cases} 1 & ifprefix\ or\ suffix\ or\ previous\ word\ \ is\ x\ and\ tag\ is\ y \\ 0 & otherwise \end{cases}$ $1 \leq prefix, suffix \leq 4$	Model 1: Prefix: 38008 Suffix: 47826 Model 2: Prefix: 5155 Suffix: 4451	1,2
$f_{NounAdj}(x,y) = \{1\ if\ prev\ tag\ is\ NN\ and\ cur\ tag\ is\ JJ\}$	Model 1: 94 Model 2: 45	

A total of 217,038 features for model 1, and 22,520 for model 2.

B. Description of the trained model

For model 1: We started by implementing features $f_{100}\ to\ f_{107}, f_{Capital}, f_{Num}$ cited in the assignment directives. All other features were implemented based on our confusion matrix output and on best practices found on the internet (the website reference for each feature is available in our code). We have performed many trainings and tests to determine the occurrence threshold for the features. We obtain our best results with threshold = 1, lambda = 0.4, and beam search = 3. After searching on the internet and reading a few of papers there is on the subject we found that most of functions increase a little bit of accuracy but not that much. However, we decide to take the same function as f_{101}, f_{102} to implement this on the previous word. Also, based on confusion matrix we created, we implemented the function fAdjNoun to reduce our missing between NN and JJ.

For model 2: We split randomly our train.wtag file into a train and test files. After, we played our features already available for model 1. Finally, we kept them all and obtained our best results with threshold = 1, beam size = 3, and lambda = 0.4.

Hardware Configuration	Processor: 2,2 GHz 6-Core Intel Core i7 Memory: 16 GB 2400 MHz DDR4
Training (+ testing) length for Model 1	51 Minutes
Training (+ testing) length for Model 2	10 min
Accuracy on Model 1 (given file test1.wtag)	**95.45%**

Accuracy on Model 2 (generated test2.wtag)	**93.82%**

3: Inference

To reduce dramatically the running time, we implemented a Viterbi using beam_search.

4: Test

A. Results on Model 1

Accuracy on the test set of Model 1: 95.38%.

```
Accuracy.: 95.45294169533386 %
Ten Worst Elements: ['VBG', 'VBD', 'VBZ', 'NNPS', 'RB', 'VBN', 'IN', 'NNP', 'JJ', 'NN']
```

Below can be found the confusion matrix of these 10 worst tags. Lines are predicted tags and columns are actual tags. Orange cells are the worst 2 tags of our trained model on the test set.

	VBG	**VBD**	**VBZ**	**NNPS**	**RB**	**VBN**	**IN**	**NNP**	**JJ**	**NN**
VBG	366	0	0	0	0	0	1	1	1	3
VBD	0	796	2	0	0	32	0	1	7	2
VBZ	2	0	456	0	0	0	0	0	0	0
NNPS	0	0	0	28	0	14	0	0	0	4
RB	0	0	0	0	725	0	44	0	14	11
VBN	2	0	0	0	0	434	0	0	37	4
IN	0	32	0	0	0	0	2492	1	2	2
NNP	2	0	0	27	0	0	3	1912	18	29
JJ	5	2	0	4	16	30	0	23	1313	82
NN	27	2	0	0	7	2	2	18	84	3153

The main mistakes our model made are between nouns and adjectives, as well as generally on any noun or adjective (a lot of false positives and false-negatives), as can be seen in the confusion matrix. This is a known issue in NLP tasks. In fact, it is very difficult to differentiate unknown adjectives from nouns based on structural features only (cause the word is unseen). Often the structure of the sentence is confusing and given the context, the same word can be seen as a noun or an adjective.

B. Results on Model 2

To evaluate the second model, we split our train.wtag file into a train and test files
The accuracy obtained was: 92.598%

```
Accuracy.: 92.59818731117825 %
Ten Worst Elements: ['VBN', 'DT', 'WDT', 'VBD', 'VBZ', 'RB', 'VB', 'CD', 'NN', 'JJ']
```

5: Competition

We used the same model we trained for the test sets. e could get different results on the competition files. We can see many reasons to that: First , the competition files could be very different from the train/test files (from other origins) and then may contain a lot of unknown words, and context. Furthermore, about Model 2 the problem could be worse cause of the size of the train set is largely smaller, and the test set was from the same origin. Estimation for competition files: Model 1: 85-90 % / Model 2: 75-80 %.