Probabilistic Models for Decision Making

September 2018

Project 4: HMM-based correction of misspelled tweets

Umberto Azadi 794276

Contents

1 Datasets Creation

1.1 Twitter Accounts

Three Twitter accounts have been selected for "training purposes", while five accounts have been selected for "testing purposes", as summarised in Table 1 . This choice is motivated by the will to see how well the trained HMM would be able to handle tweets wrote from different persons, with the underlying assumption that different people will most likely use different words and different sentence structures while they are writing their tweets.

Furthermore, the tweets that have downloaded and used are always written by the owner of the account, i.e. all the retweets have been filtered during the downloading process.

Account Tag	Num of Tweets Downloaded	Num of Tweet used for Training	Num of Tweet used for Testing
@NASA	2424	2333	45
@BarackObama	2855	2333	45
@CNN	2922	2334	45
@nytimes	171	0	158
@BillGates	193	0	157

Table 1: Twitter account selected and their usage

1.2 Error Introduction

The errors have been introduced by selecting a percentage (p) of characters that have to be changed. Therefore when a tweet with n chars has been given in input to the Error Introduction Procedure, the first thing that it does is select $int(n \cdot p)$ random numbers, which represent the indexes of the chars that have to be changed. Then, in order to simulate a more realistic process of error introduction, the selected chars are changed based on the following rule:

Let be $s_{correct}$, $s_{error} \in$ (ASCII letters $\cup$ Digits $\cup$ {" ", "_", "#", "@"}), where s_{error} is randomly chosen, then s_{error} will substitute $s_{correct}$ in the misspelled version of that tweet with a probability equal to:

$$\frac{1}{\sqrt{(x_{correct} - x_{error})^2 + (y_{correct} - y_{error})^2}} \tag{1}$$

Where:

- the s_{error} character will be sampled randomly until a change is accepted;
- the x and y values of each symbol can be retrieved through the following table:

coordinates	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10	10.5	11
1	1		2		3		4		5		6		7		8		9		0		'
2	q		w		e		r		t		y		u		i		o		p		
2.5		Q		W		E		R	_	T		Y		U		I		O		P	
3	a		s		d		f		g		h		j		k		l		@		
3.5		A		S		D		F		G		H		J		K		L			
4	z		x		c		v		b		n		m						#		
4.5		Z		X		C		V		B		N		M							
5									<space>												

1.3 Training sets

The training file was composed of 7000 tweets downloaded from three different twitter accounts, as it can be seen in Table 1. In order to evaluate the performance of the model as training data grows, five training sets have been created, each one with respectively the first 1400, 2800, 4200, 5600, 7000 tweets of the original training file. Then for every tweet in each training set, the error introduction procedure, described in Section 1.2, has been run three times, with respectively a percentage of error of 0.1, 0.2 and 0.3.

As results of this operation, a total of 15 training sets have been created and the information about them are summarised in the following table:

Training Set ID	Number of Tweets	Number of Lines	Percentage of Error
1	1400	2800	0.1
2	2800	5600	0.1
3	4200	8400	0.1
4	5600	11200	0.1
5	7000	14000	0.1
6	1400	2800	0.2
7	2800	5600	0.2
8	4200	8400	0.2
9	5600	11200	0.2
10	7000	14000	0.2
11	1400	2800	0.3
12	2800	5600	0.3
13	4200	8400	0.3
14	5600	11200	0.3
15	7000	14000	0.3

Table 2: Training sets

1.4 Testing sets

The testing file was composed of 450 tweets downloaded from five different Twitter accounts, as it can be seen in Table 1. Then for every tweet in the testing file the error introduction procedure, described in Section 1.2, has been run four times, with respectively a percentage of error of 0.05, 0.1, 0.2 and 0.3.

As results of this operation, a total of 4 testing sets have been created and the information about them are summarised in the following table:

Testing Set ID	Number of Tweets	Number of Lines	Percentage of Error	Training sets tested
1	450	900	0.05	All 15 of them
2	450	900	0.1	1 - 5
3	450	900	0.2	6 - 10
4	450	900	0.3	11 - 15

Table 3: Testing sets

2 HMM Description

The HMM designed has 66 states, which represent the characters that the HMM can use to correct a tweet, and each one them have 67 possible related observations, which represent the characters that the user had typed. Specifically, the characters related to each state and observable are reported in the following table:

States	<space>, @, #, ', 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G, H, I, J, K, L, M, N, P, Q, R, S, T, U, V, W, X, Y, Z, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z
Observable values	_, <space>, @, #, ', 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G, H, I, J, K, L, M, N, P, Q, R, S, T, U, V, W, X, Y, Z, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z

It is possible to see that the only character that can be observed but it is not a state is "_", and that is because the character "_" has been selected to represent the absence of a character. Therefore, during the training step, this character will be kept in the misspelled tweets, in order to make the HMM learn how many times a character is forgotten by the user. While, during the testing step, this character will be removed in the misspelled tweets in order to make it consistent with the meaning that has been assigned to it.

2.1 Training Procedure

This section contains the explanation regarding how the three matrices (Transition, Emission, Initial) that characterise an HMM are obtained. Each one of the matrices is estimated through a 3 steps procedure, summarized in Figure 1.

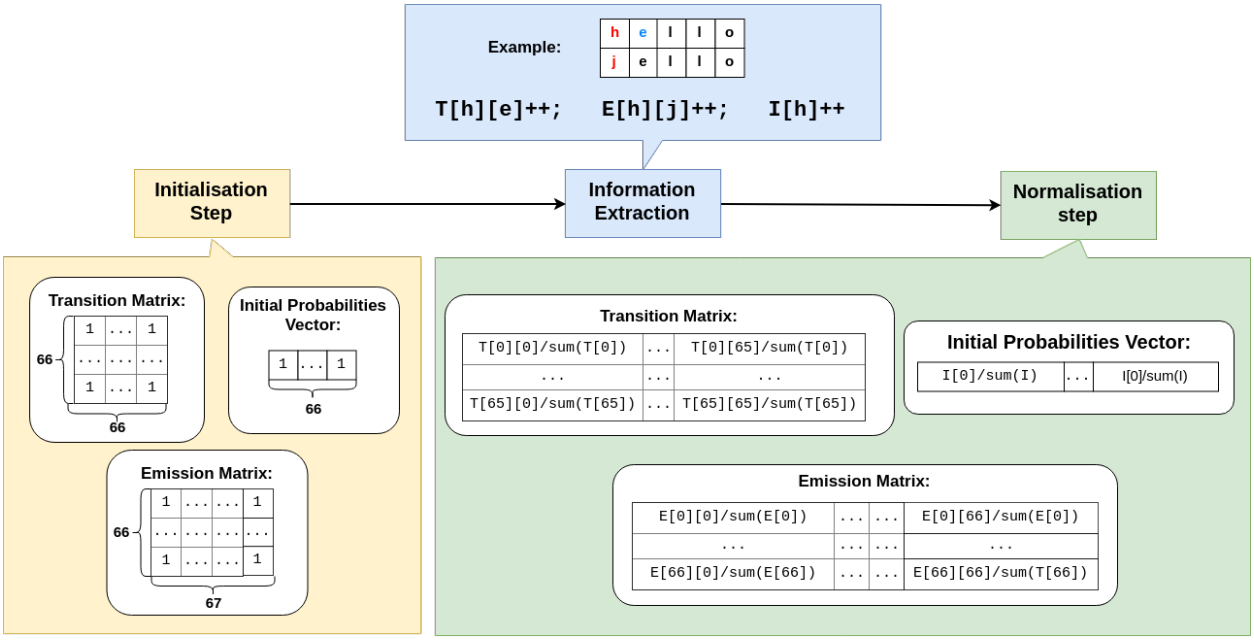


Figure 1: Training Procedure

1. Initialisation step:

   Transition Matrix: a square matrix with 66 rows is created and each cell is initialised with the default value 1

   Emission Matrix: a matrix with 66 rows and 67 columns is created and each cell is initialised with the default value 1

   Initial Probabilities Vector: a vector with 66 cells is created and each cell is initialised with the default value 1

2. Information Extraction from the training set:

   Transition Matrix: for each character, `x`, in the training set, it is taken the character immediately after it, `y`. Then the value of the cell obtained by searching the row related to character `x` and the column related to character `y` is increased by one.

   Emission Matrix: for each character, `x`, in the training set, it is taken the related character in the misspelled version of the same tweet, `y`. Then the value of the cell obtained by searching the row related to character `x` and the column related to character `y` is increased by one. Since it will happen that the far highest value of each row will be found in the cell that has the row and the column related to the same character, *it is also possible to set an upper bound for that specific cell.*

   Initial Probabilities Vector: for each tweet in the training set, it is taken the first character of it, `x`, and the value of the cell related to `x` is increased by one.

3. Normalisation step:

   Transition Matrix: the value of each cell of the matrix is replaced with the value of that cell divided by the sum of all the values of the cells in the same row.

   Emission Matrix: the value of each cell of the matrix is replaced with the value of that cell divided by the sum of all the values of the cells in the same row.

   Initial Probabilities Vector: the value of each cell of the vector is replaced with the value of that cell divided by the sum of all the values of the cells of the vector.

It is important to highlight that thanks to:

- the Initialisation step all the cells of the three matrices contain a value that is strictly greater than 0;
- the Normalisation step if one character occurs very rarely, then the probabilities in the related rows of each one of the three matrices should be almost uniform.

2.2 Correction Procedure

Once the HMM is trained, i.e. the three matrices are created through the procedure described in Section 2.1, it can be used to correct two types of misspell errors: *(i) wrong character error* (Ex. hallo instead of hello) and *(ii) missing character error* (Ex. hllo instead of hello). Both these approach are also described in Figure 2.

The detection of the misspelled words is accomplished through a vocabulary, that will contain a set of 1000 words plus all the words found in the training set. Therefore any word in the tweets that is not found in the vocabulary will be considered as misspelled.

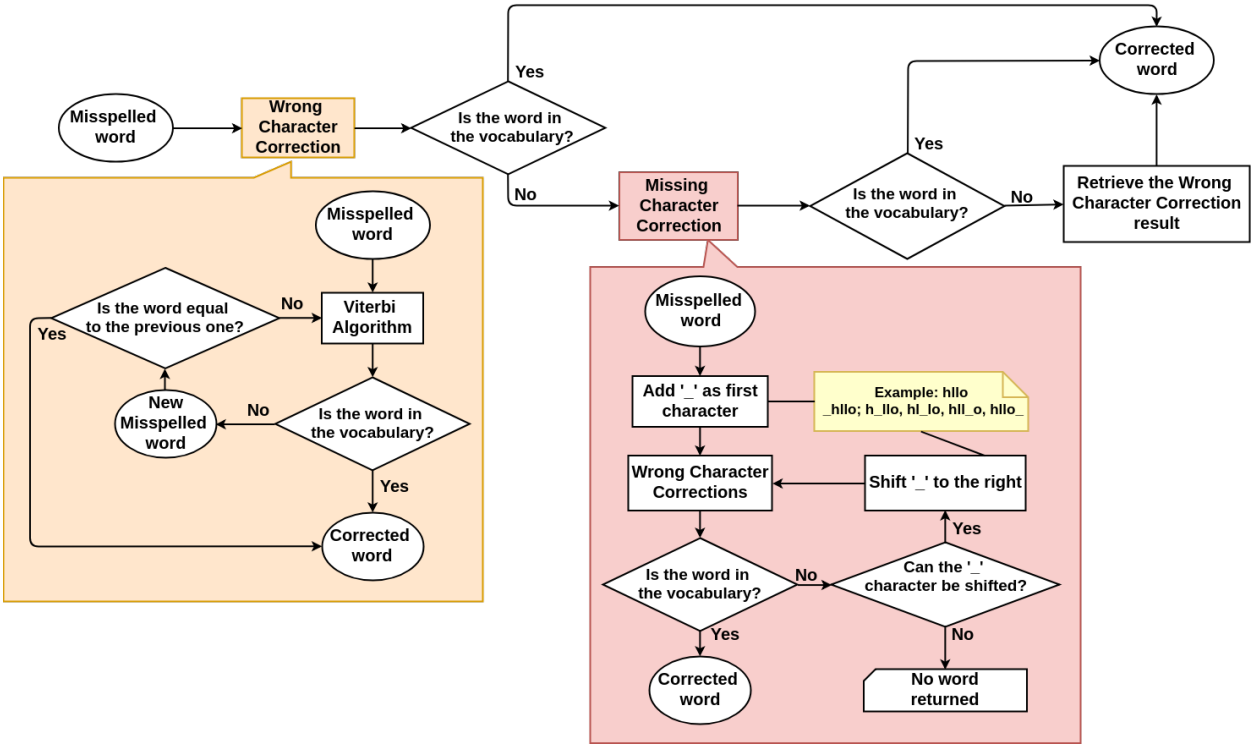


Figure 2: Correction Procedure

Both these attempt are used for each misspelled word:

- *wrong character correction*: this correction strategy consists only to keep applying the Viterbi algorithm until a word in the vocabulary is obtained or for two consecutive times the algorithm outcome is the same word
- *missing character correction*: the character "_" is placed in each possible position, one position at a time (Ex. hllo $\mapsto$ _hllo, h_llo, hl_lo, hll_o, hllo_), and for each one of these words the wrong character correction is applied. The correction stop when a word in the vocabulary has been obtained or if all the possible combination have been applied.

Therefore, the corrected word is:

- the word in the vocabulary obtained through the *wrong character correction*;
- if the word returned by the *wrong character correction* is not included in the vocabulary, the first word in the vocabulary obtained through the *missing character correction*;
- if neither of the character corrections were able to return a word in the vocabulary, then the word obtained through the *wrong character correction* is returned.

  This last choice is motivated by the fact that the words obtained through the *missing character correction* that are not in the vocabulary often are extremely far from a correction. That is because the character "_" will be for sure replaced with another character that in most of the cases will not fit at all within the word, while the remaining cases are the ones in which a word in the vocabulary is obtained.

3 Performance Evaluation

The evaluation process has as objectives the assessment of: *(i)* the goodness of this approach, *(ii)* the predictive skills of the model as training data grows, *(iii)* the predictive skills of the model as the percentage of error grows and *(iv)* the impact, on the predictive skills of the model, caused by setting of an upper bound to the value contained in the cell that has the row and the column related to the same character.
In order to reach these objectives 45 HMMs have been trained (the training procedure is presented in Section 2.1). Each one of them is uniquely identified by a list of two attribute:

- the set with which the HMM has been trained, that could be chosen among a list of 15, which are outlined in Table 2;
- an upper bound to the value contained in the cell that has the row and the column related to the same character. It can be chosen by a list of three possible values: `no upper bound`, `0.5`, `0.75`.

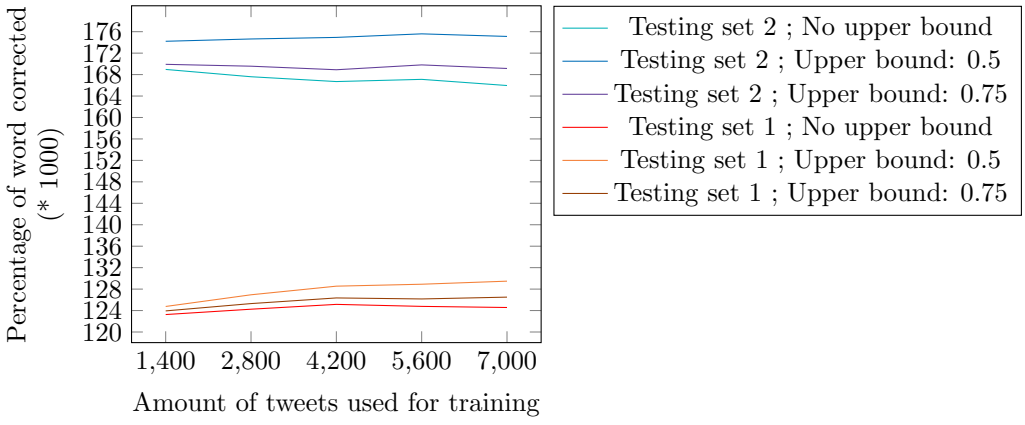
The performance of an HMM has been measured through the metrics defined as the average of the percentages of words successfully corrected by it, which is computed as follows:

1. For each misspelled tweet (i.e. tweet modified through the Error Introduction procedure, see 1.2) the percentage of correct words has been calculated (number of correct word divided by the total number of word).
2. For each corrected tweet (i.e. tweet modified through the Correction procedure, see 2.2) the percentage of correct words has been calculated (number of correct word divided by the total number of word).
3. For each tweet the value obtained through the 2^{nd} step has been subtracted to the value obtained through the 1^{st} step.
4. Finally, it has been computed the mean of all the values obtained through the 3^{rd} step.

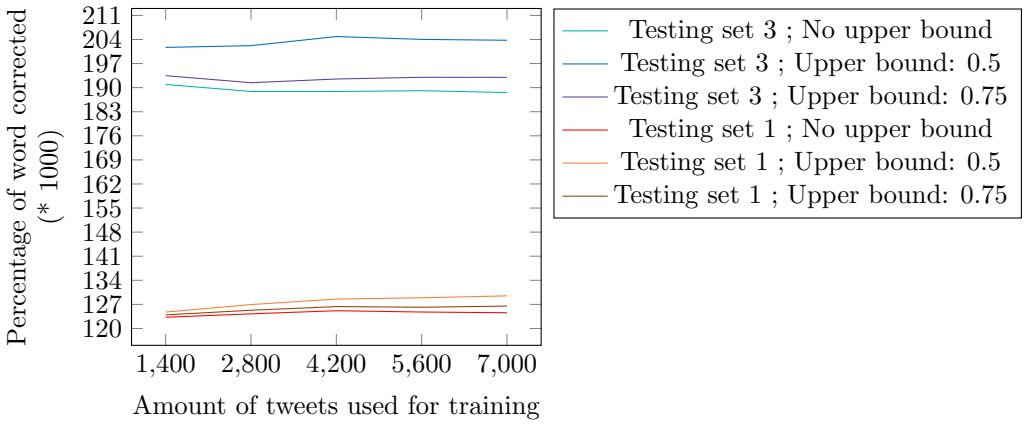
The tweets through which this comparison has been accomplished are the ones contained in the testing sets listed in Table 3.

The results are shown in the following diagrams:

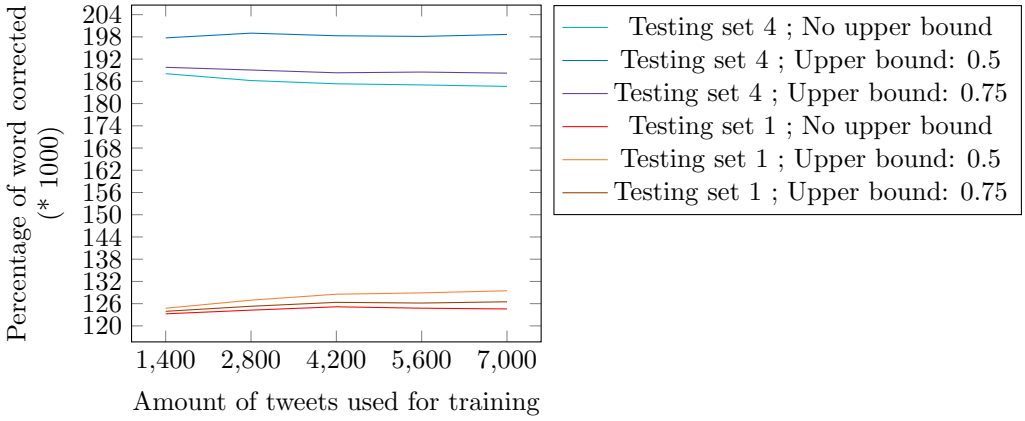
Percentage of words successfully corrected by the HMMs trained with 10% error



Percentage of words successfully corrected by the HMMs trained with 20% error



Percentage of words successfully corrected by the HMMs trained with 30% error



As it is possible to see from the diagrams above:

- the percentage of error that seems to lead to better performance is `20%`
- the upper bound selected for the value contained in the cell that has the row and the column related to the same character that seems to lead to better performance is `0.5`
- the grow of the training set size influence for sure the performance of the HMM, however it not always lead to an improvement.
- Finally, the HMM that achieved the better performance is the HMM trained with the Testing Set number 8 (i.e. the one containing *4200 tweets* with a percentage of error of *20%*) and *0.5* as upper bound selected for the value contained in the cell that has the row and the column related to the same character.

4 GUI

The best HMMs for each percentage of error can be actually used for the tweet correction through a GUI.

4.1 Usage

In order to start the application, the following command has to be typed in the command line:

```
python GUI.py
```

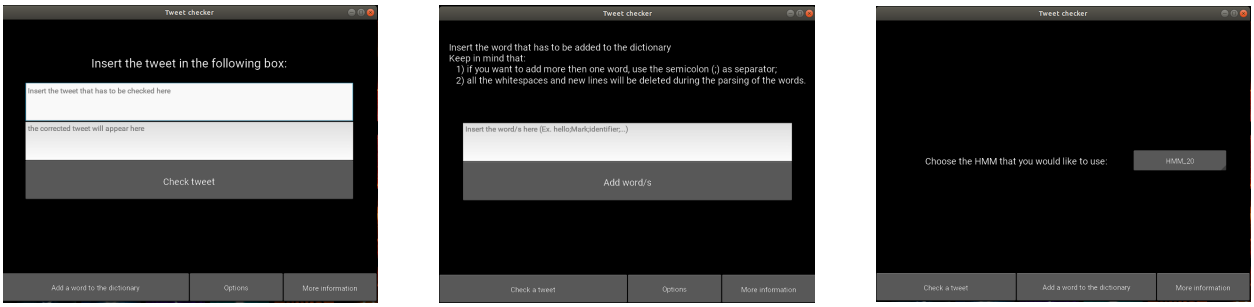
please keep in mind that the following libraries are required:

```
- python 2.7.x
- kivy (v1.10.1)
- ghmm (v0.8)
```

4.2 Description

The GUI is organised is three views:

- **Tweeter Checker View**: that allows the user to insert his tweet and use check it;
- **Dictionary View**: that allows the user to add a word to the dictionary;
- **Option View**: that allows the user to switch the HMM used for the correction. The available choices are the best HMMs for each percentage of error, which are:
  - *HMM_10*: The HMM trained with 7000 tweets, the percentage of error of 10% and the upper bound of 0.5;
  - *HMM_20*: The HMM trained with 4200 tweets, the percentage of error of 20% and the upper bound of 0.5;
  - *HMM_30*: The HMM trained with 7000 tweets, the percentage of error of 30% and the upper bound of 0.5;



(a) Tweeter Checker View (b) Dictionary View (c) Option View

Figure 3: GUI of the project