# Alex Allauzen

Cours

## AIC-TC-Assignments

## TC4: Second-Order HMM for typos correction

The goal is to design a model to correct typos in texts without a dictionary.

In this problem, a state refers to the correct letter that should have been typed, and an observation refers to the actual letter that is typed. Given a sequence of outputs/observations (i.e., actually typed letters), the problem is to reconstruct the hidden state sequence (i.e., the intended sequence of letters). Thus, data for this problem looks like:

```
[('t', 't'), ('h', 'h'), ('w', 'e'), ('k', 'm')]
[('f', 'f'), ('o', 'o'), ('r', 'r'), ('m', 'm')]
```

The first example is misspelled: the observation is **thwk** while the correct word is **them**. The second example is correctly typed.

Data for this problem was generated as follows: starting with a text document, in this case, the Unabomber's Manifesto, which was chosen not for political reasons, but for its convenience being available on-line and of about the right length, all numbers and punctuation were converted to white space and all letters converted to lower case. The remaining text is a sequence only over the lower case letters and the space character, represented in the data files by an underscore character. Next, typos were artificially added to the data as follows: with 90% probability, the correct letter is transcribed, but with 10% probability, a randomly chosen neighbor (on an ordinary physical keyboard) of the letter is transcribed instead. Space characters are always transcribed correctly. In a harder variant of the problem, the rate of errors is increased to 20%.

You can download the datasets in this archive file. This archive contains 4 pickles: train10 and test10 constitute the dataset with 10% or spelling errors, while train20 and test20 the one with 20% or errors.

## TODO:

- (3pts) Dry run: Train a first-order HMM using the training data. This is basically what we did in lab sessions for POS-tagging. Compute the error rate (at the character level) and compare this results with the dummiest classifier that just do nothing. You can also compute the number of errors your model can correct and the number of errors your model creates.
- (7pts) Second Order HMM: To improve the performance, we can increase the order of the HMM. Implement a second Order model for this task (this means that the probability of the next state depends on the current state and the previous one as well). A convenient way to implement a second order HMM, is to think about it as a variable change.
- (5pts) Critics and evolution: This model is somehow limited. For instance, it only handles substitution errors. Could you describe a way to extend this model to also handle noisy insertion of characters ?
- (5pts) Same question for deletion (or omitted characters) ?
- (5pts) Unsupervised training: Propose and discuss some ideas to do unsupervised training for this task. For this question you should provide details on : what kind of data, which parameters will be involved, ...

For the last three questions, be as precise as you can. An implementation is not mandatory.

## Submission Checklist

You can rely on existing libraries such as nltk of sklearn and you can use other programming languages. You must send an archive with: - the source code (without external libraries of course) necessary to reproduce your results - a report that presents, explains, analyses and discusses your results.

## Grading Scheme:

Just follow the TODO list. Extra points will be given for an implementation for the last three questions (5pts each)

FIND IT

Go

HomePage
Research
Stages/internships
Teaching (in French only)
–> L2 Apprentissage Artificiel
–> TER M1
–> AIC: TC Algorithme d'inférence

LINKS

Machine Translation @ LIMSI
Spoken Language Processing Group
LIMSI
Université Paris Sud