

Due date: November 6 13, 2006

Late submission : 10% per day.

Submission: You should submit a paper listing and report AND an electronic version of your code.

Language Models: In this assignment, you will experiment with language models to:

1. generate pseudo-random text, and
2. recognize a text's genre or author.

Your program should first build a character-based model and a word-based model from given input texts. Your program should work for various orders of LM (ex. unigram, bigram, ...).

1. Pseudo-Random Text generation:

Pick electronic texts from different textual genres (e.g. sports news, horoscopes, cooking recipes...) and train your models on them. Then, generate pseudo-random text according to your models. To generate the texts, you can output the first N characters (or words) of the original training text, then repeatedly generate successive pseudo-random "grams" according to your LM. Stop generating when you have as many "grams" as the input.

For example, if your training text is:

Today is highly favorable for work of a mental nature. Not everything ...

and you have built a tri-gram character model, then your program would generate:

Tod + a series of pseudo-random characters according to the probability distribution in your LM.

Experimentation: Once your code runs, perform 2 experiments and analyze your results. For example:

- What is the effect of the order of LM on the quality of the generated texts?
- What if you train on horoscopes and start generating from cooking recipes?
- Does a character-LM produce better results than a word-based LM?
- ...

For the experimentations, let your imagination and intuition guide you. The point is to experiment with different situations, then report your findings.

2. Genre or Author Recognition:

Now you will use your LMs to identify either a text's genre (e.g. scientific article, news article, joke, instructions...) or the author of a text (you decide).

Pick electronic texts from different textual genres or of the same genre, but different authors¹. Train your models on them, and see if you can identify the textual genre or the author of new documents.

Experimentation: Once your code runs, perform 2 experiments and analyze your results. For example:

- What is the effect of the order of LM on the quality of the recognition?
- What is the effect of the training size on the recognition?
- ...

¹ A good source of online books is the Project Gutenberg http://www.gutenberg.org/wiki/Main_Page

Note: You can implement your own language model or modify any freely available code on the Web². On the course Web page, I have some code for character-based and word-based languages models. You are welcome to use them, but make sure you specify where your code comes from in your report.

Report: Write a report (5 to 10 pages) to describe your code and your experimentation. Your report must describe:

- The program (1-2 page maximum):
 - o Describe the code itself (its source, modifications done, choice of programming language,...)
 - o Indicate the instructions necessary to run your code (files, commands...)
- The hypothesis and experimental setup (~1-2 page):
 - o Describe the questions that you experimented with and the methodology used
 - o Describe your corpora briefly (size, source, language, ...)
 - o Describe the methods that you used (do not re-explain the theory, but just the parameters you used; order, smoothing technique...)
- Analysis of the results (~4-6 pages)

Please note: Your report should **not** be a detailed explanation of your code (data structures ...). It should be an analytical report describing your experiments and an analysis of the results.

Submission:

Both the code and the report must be submitted electronically AND in paper.

- Paper submission. In class, you must submit a listing of your program & results and the report.
- Electronic submission. Through the Electronic Submission Form (<https://eas.encs.concordia.ca/eas/authentication.jsp>), you must submit the code of your programs & results and an electronic version of your report.

² see <http://opennlp.sourceforge.net/> for example