CS4740/5740 Introduction to NLP
Fall 2017
Named Entity Recognition with HMMs

Proposal: due via Gradescope and CMS by Monday, Oct 2nd, 11:59pm
Final report: due via Gradescope and CMS by Thursday, Oct 19th, 11:59pm

1 Overview

In this project, you will implement a model that identifies relevant information in a text and tags it with the appropriate label. Particularly, the task of this project is **Named Entity Recognition (NER)**, in which the semantic class associated with a set of contiguous tokens corresponding to a *person, location, organization*, etc. is annotated.

For this project, you will implement a **Hidden Markov Model** and use it to detect named entities.

2 Task and Dataset

The NER task uses the IOB encoding scheme. Each token is associated with a label O if it is **O**utside the entity, label B-xxx if it is the first token (i.e. **B**eginning) of entity xxx, and I-xxx if it is within (i.e. **I**nside) entity xxx but not the first of the entity. We will concentrate on four types of named entities: persons (PER), locations (LOC), organizations (ORG) and names of miscellaneous entities (MISC) that do not belong to the previous three groups.

The training data is organized by sentences. Note that it is **not** split into separate training and development/validation sets. You will need to do this yourself as needed. Each sentence is associated with three lines, where the first line contains the tokens, the second line contains the corresponding Part-Of-Speech (POS) tags, and the third line contains the correct labels. For example:

Volkswagen AG won 77,719 registrations , slightly more than a quarter of the total .
NNP NNP VBD CD NNS , RB RBR IN DT NN IN DT NN .
B-ORG I-ORG O O O O O O O O O O O O O

specifies an example sentence in the training data. “Volkswagen AG” is labeled as a ORG (Organization) entity in the sentence. All non-entities are labelled

"O". The test data only has words and POS tags. In order to evaluate your system's performance, you will upload your predictions for the test set to Kaggle.

3 Implementation

You should implement a **Hidden Markov Model (HMM)** for this task.

1. You may use any programming language that you'd like and any preprocessing tools that you can find. It might be possible, for example, to use a toolkit just to extract n-gram information, but to write the HMM code by yourself.

2. It could be beneficial to reserve some portion of the training dataset for validation purposes. Describe the details of your experimental designs in the report.

3. **Develop baseline systems to be compared with your own model**. You are required to implement baseline systems for comparison. One simple option is to first build a lexicon of all named entities that appear in the training corpus, and identify during testing only those named entities that are part of the lexicon. Note that baseline systems should be compared to your system in the report.

4. **Code of Academic Integrity: We encourage collaboration regarding ideas, etc. However, please do not copy code from online or share code with other students. We will be running submissions through MOSS to detect plagiarism.**

4 Kaggle Competition

We will launch a Kaggle competition for the task. Your submission to Kaggle should be a csv file consisting of five lines and two columns. The first line is a fixed header, and each of the rest four lines corresponds to one of the four types of named entities. The first column is the type identifier (PER, LOC, ORG or MISC), and the second column is a list of entities (separated by single space) that you predict to be of that type. Each entity is specified by its starting and ending position (concatenated by a hypen). To make positions unambiguous, we provide the position for each token in the test corpus. Suppose the input contains two sentences:

Renate Goetschl of Austria won the women 's World Cup downhill race
NNP NNP IN NNP VBD DT NNS POS NNP NNP RB NN
0 1 2 3 4 5 6 7 8 9 10 11
ZIFA vice-chairman Vincent Pamire said Grobbelaar would take charge for a match against Tanzania
NNP NN NNP NNP VBD NNP MD VB NN IN DT NN IN NNP

12 13 14 15 16 17 18 19 20 21 22 23 24 25

then your output should look like this:

```
Type,Prediction
PER,0-1 14-15 17-17
LOC,3-3 25-25
ORG,12-12
MISC,8-9
```

The standard measures to report for NER are recall, precision, and F1 score (also called F-measure) evaluated **at the entity level** (not at the token level). Precision is $\frac{|C \cap P|}{|P|}$ and Recall is $\frac{|C \cap P|}{|C|}$ and F1 is $\frac{2Prec \times Recall}{Prec + Recall}$, where P and C are the sets of predicted and correct name entities respectively. When you upload your predictions you will see the evaluation results (mean F1 score of the four entity types) on half of the test data. You will be able to see your score on the other half of the test set after the submission deadline.

You should include the final evaluation result in your final report before the Kaggle competition ends. Note that once the Kaggle competition closes, you cannot make additional submissions to Kaggle.

5 Extensions

Here are several extensions you can implement and experiment with. **Doing at least one extension is mandatory. Note that all the extensions can be used in the Kaggle competition**.

1. Experiment with different orders of n-gram-based features: will bigrams be adequate or will trigrams (or 4-grams, etc.) be better?
2. Use toolkit(s) to try other algorithms, or to compare different variations of HMMs / CRFs.
3. Implement a secondary sequence tagging system that is different from your primary implementation, e.g. MEMMs or CRFs if your primary implementation is HMMs.
4. Something else of your choosing. In this case, explain what you would like to do in your proposal, and we may give feedback on it or suggest that you try something else instead.

6 Proposal

Describe your sequence-tagging system and implementation plan in 1 page. You should consider

- Explain the algorithmic key points of your model. Especially think about which are hidden variables and observed variables for our setting, and what are the corresponding model parameters.
- What are some pros and cons of using HMMs for this task? Can you think of other techniques that may be effective?
- State which extension you are planning to do. How do you think this will help solve the problems you raised in the previous bullet? While you might end up implementing different extensions, it will help us to provide you feedback.

In addition to submitting a 1-page proposal document, you must also submit code for at least one baseline system and report its performance on the partial test set via Kaggle. Include details if you split the training set into training and validation parts.

7 Report

You should submit a short document (5-6 pages will suffice) that contains the following sections. (You can include additional sections if you wish.)

1. **Sequence Tagging Model**
   (a) **Implementation Details.** Make clear which sequence tagging method(s) that you selected. Make clear which parts were implemented from scratch vs. obtained via an existing package. Explain and motivate any design choices providing the intuition behind them.
   (b) **Pre-Processing.** Explain and motivate the pre-processing steps you apply to the data.
   (c) **Experiments.** Describe the motivations and methodology of the experiments that you ran. Clearly state what were your hypotheses and what were your expectations.
   (d) **Results.** Summarize the performance of your system and any variations that you experimented with on both the training/validation and test dataset. **Note that you have to compare your own system to at least one other non-trivial baseline system.** Put the results into clearly labeled tables or diagrams and include your observations and analysis. An error analysis is required – e.g. what sorts of errors occurred, why? When did the system work well, when did it fail and any ideas as to why? How might you improve the system?
   (e) **Competition Score** Include your team name and the screenshot of your best score from Kaggle.

2. **Extensions**

   Explain the extension(s) that you decided to do. Include the implementation details, the experiments, and the results similar to the previous section. If your extensions contribute to the competition score, analyze why they help. If not, try to explain why it was not useful.

3. **Individual Member Contribution**

   Briefly explain the contribution of an individual group member. Report if working loads are unfairly distributed.

8 Grading Guide

- (10 pts) Proposal
- (35 pts) Design and implementation of the sequence-tagging system.
- (10 pts) Experiment design and methodology
- (30 pts) Report: organization, clarity and quality of the report.
- (10 pts) At least one extension: implementation, experiments and discussion.
- (5 pts) **Submission to Kaggle.** (not optional!)

8.1 Things to avoid

Don't be ridiculously inefficient. You are not supposed to spend too much time optimizing your code, but it SHOULD NOT take forever either. Bigram Viterbi is $O(sm^2)$ where s is the length of the sentence and m is the number of tags. Your implementation should have similar efficiency.

9 What to Submit

9.1 Part One: Proposal and Baseline System

- Proposal (one-page pdf file) including results and description of your baseline.
  - Description of baseline and motivation
  - Results and analysis of baseline model
- Code for at least one baseline system.
  - README file detailing how to run your code.
- Archive all of the above in a zip file, and upload it to CMS. (due Oct 2nd, 11:59pm)

9.2 Part Two: Final Submission

- Your team name on Kaggle **- include this at the top of your report. -2 if not present.**
- Source code with adequate comments, and executables (only include code that you wrote yourselves, DO NOT include code from existing toolkits/packages)
- README file detailing how to run your code.
- Prediction output (the file you submitted to Kaggle)
- Report (pdf file)
- Archive all of the above in a zip file, and upload it to CMS. (due Oct 19th, 11:59pm)