SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

01.112 Machine Learning, Fall 2018

Design Project

Due 9 Dec 2018, 5pm

This project will be graded by Ngo Van Mao
Please submit your work to eDimension.

Please form groups for this project early, and start this project early.

Instructions

Please read the following instructions carefully before you start this project:

- This is a group project. You are allowed to form groups in any way you like, but each group must consist of either 2 or 3 people. Please submit your group information to eDimension latest by Friday 26 Oct 2018 5pm.
- You are strictly NOT allowed to use any external resources or machine learning packages. You will receive 0 for this project if you do so.
- Part 1 deadline is Friday 2 Nov 2018 5pm. **Please start working on part 1 as early as possible** as this part is done **individually**, and you do not need to form a team before you start. Annotated training and development set will be shared with you all on 9 Nov 2018.
- Each group should submit code together with a report summarizing your work, and give clear instructions on how to run your code. Please also submit your system's outputs. Your output should be in the same column format as that of the training set.

Project Summary

In an interview with Michael I. Jordan, the Pehong Chen Distinguished Professor in the Department of Electrical Engineering and Computer Science and the Department of Statistics at the University of California, Berkeley and a giant on machine learning, he was asked "If you got a billion dollars to spend on a huge research project that you get to lead, what would you like to do?". He answered: "I'd use the billion dollars to build a NASA-size program focusing on natural language processing".

Indeed, one of the most challenging problems within industry and academia is to design intelligent systems that are capable of comprehending human languages. Natural language processing (NLP) is regarded as one of the most challenging yet most promising directions within the field of artificial intelligence in the next 10 years. In this project, we will be working together on building some basic models for solving several simple NLP problems.

Many problems within the field of NLP are essentially structured prediction problems, among which sequence labeling is the simplest class of problems. The hidden Markov model (HMM) that we have learned in class is a simple structured prediction model. In this design project, we would like to design our sequence labelling model for informal texts using the HMM that we have learned in class. We hope that your sequence labelling system for informal texts can serve as the very first step towards building a more complex, intelligent sentiment analysis system for social media text. Specifically, we will focus on building two NLP systems – a *sentiment analysis* system as well as a *phrase chunking* system for Tweets.

The files for this project are in the files `EN.zip`, `FR.zip`, as well as `SG.zip`, `CN.zip` (the latter two will be available on 9 Nov 2018, after we all have finished part 1). For each dataset, we provide a labelled training set `train`, an unlabelled development set `dev.in`, and a labelled development set `dev.out`. The labelled data has the format of one token per line with token and tag separated by tab and a single empty line that separates sentences.

The format for the `SG` dataset (as well as `FR` and `CN` datasets) can be something like the following:

```
Best O
Deal O
Chiang B-positive
mai I-positive
Tours I-positive
, O
The O
North O
of O
Thailand B-neutral
To O
Get O
special O
Promotion O
and O
free O
Transfer O
roundtrip O
. O
Contact O
: O
... O
http://t.co/sSn1OBTZ O
```

where labels such as `B-positive, I-positive` are used to indicate **B**eginning and the **I**nside of the entities which are associated with a positive sentiment. `O` is used to indicate the **O**utside of any entity. Similarly for `B-negative, I-negative` and `B-neutral, I-neutral`, which are used to indicate entities which are associated with negative and neutral sentiment, respectively.

The format for the `EN` dataset can be something like the following:

```
Cant B-VP
wait I-VP
```

```
for B-PP
the B-NP
ravens I-NP
game I-NP
tomorrow B-NP
.... O
go B-VP
ray B-NP
rice I-NP
!!!!!!! O
```

where labels such as `B-VP, I-VP` are used to indicate **B**eginning and the **I**nside of a **V**erb Phrase. Similarly for other tags such as `B-NP, I-NP` and `B-PP, I-PP`, which are used to indicate spans which are associated with noun phrases and propositional phrases etc. `O` is used to indicate the **O**utside of any phrase.

Overall, our goal is to build a sequence labelling system from such training data and then use the system to predict tag sequences for new sentences. Specifically:

- We will be building two sentiment analysis systems for two different languages from scratch, using our own annotations.
- We will be building yet another two NLP systems (one for sentiment analysis and the other for phrase chunking) for two different languages using annotations provided by others.

1 Part 1 (*15 points*, due 2 Nov 2018 at 5pm. Please budget your time well.)

The first and most important step towards building a supervised machine learning system is to get annotated data. This is also often one of the most difficult and most challenging steps in building a practical machine learning system, as we will see in this project. To allow each of us to have a full end-to-end experience on how challenging it is to build a practical supervised machine learning system, in the first part of this project, we will work together to get annotated data for performing sentiment analysis from social media data (some of them are collected from local social media users). You will receive 10 points if you complete the annotation. An additional 5 points will typically be awarded to you too unless we found the quality of your annotation is unacceptable. We will use an automatic approach to assess the quality of your annotations. Your annotations will be compiled and distributed to your fellow students in order to complete Part 2 and Part 3 of this project.

Please visit the following site for the annotation interface:

`http://ml-project.statnlp.org/annotation.php?id=/**YOUR_ID_HERE**/`

You then need to key in your student ID to proceed to the next step for annotation. For example, if your student ID is 1001949, please visit the following link for annotation:

`http://ml-project.statnlp.org/annotation.php?id=1001949`

Alternatively, visit the following site and type in your student ID:

`http://ml-project.statnlp.org/annotation.php`

You need to log in to start the annotation process. The user name and password are both your student ID. We also provide a sample collection of annotated data, which is available here:

`http://ml-project.statnlp.org/annotation.php?id=1000000`

Essentially, we are interested in annotating all major entities together with their sentiment information. Detailed instructions on the annotation can be found at *http://ml-project.statnlp.org/annotation.pdf*. The annotation interface is straightforward to use, but if you have questions there is a manual here:

`http://brat.nlplab.org/manual.html`

Disclaimer: to grant us the right to re-distribute your annotated data to your other fellow classmates and for potential future usage, by submitting your annotations online, you agree that your annotated data will be in public domain unless otherwise stated. Please contact Ngo Van Mao if you have questions or doubts on this.

2 Part 2 *(25 points)*

Recall that the HMM discussed in class is defined as follows:

$$p(x_1, \ldots, x_n, y_1, \ldots, y_n) = \prod_{i=1}^{n+1} q(y_i|y_{i-1}) \cdot \prod_{i=1}^{n} e(x_i|y_i) \tag{1}$$

where $y_0 = \texttt{START}$ and $y_{n+1} = \texttt{STOP}$. Here q are transition probabilities, and e are emission parameters. In this project, x's are the natural language words, and y's are the tags (such as `O`, `B-positive`).

- Write a function that estimates the emission parameters from the training set using MLE (maximum likelihood estimation):

$$e(x|y) = \frac{\texttt{Count}(y \rightarrow x)}{\texttt{Count}(y)}$$

(5 points)

- One problem with estimating the emission parameters is that some words that appear in the test set do not appear in the training set. One simple idea to handle this issue is as follows. We introduce a special word token `#UNK#`, and make the following modifications to the computation of emission probabilities:

$$e(x|y) = \begin{cases} \frac{\texttt{Count}(y \rightarrow x)}{\texttt{Count}(y)+k} & \text{If the word token } x \text{ appears in the training set} \\ \frac{k}{\texttt{Count}(y)+k} & \text{If word token } x \text{ is the special token } \texttt{\#UNK\#} \end{cases}$$

(This basically says we assume from any label y there is a certain chance of generating `#UNK#` as a rare event, and empirically we assume we have observed that there are k occurrences of such an event.)

During the testing phase, if the word does not appear in the training set, we replace that word with the special word token `#UNK#`.

Set k to 1, implement this fix into your function for computing the emission parameters.

This method is called *smoothing*. The resulting emission parameters are called *smoothed* emission parameters. Let us assume such a smoothing method for the emission parameters is always used in the subsequent questions of this project (you may also do so in part 5 if you choose to).

(10 points)

- Implement a simple sequence labeling system that produces the tag

$$y^* = \arg\max_y e(x|y)$$

for each word x in the sequence.

For all the four datasets `EN`, `FR`, `CN`, and `SG`, learn these parameters with `train`, and evaluate your system on the development set `dev.in` for each of the dataset. Write your output to `dev.p2.out` for the four datasets respectively. Compare your outputs and the gold-standard outputs in `dev.out` and report the precision, recall and F scores of such a baseline system for each dataset.

The precision score is defined as follows:

$$\texttt{Precision} = \frac{\text{Total number of correctly predicted entities/phrases}}{\text{Total number of predicted entities/phrases}}$$

The recall score is defined as follows:

$$\texttt{Recall} = \frac{\text{Total number of correctly predicted entities/phrases}}{\text{Total number of gold entities/phrases}}$$

where a gold entity/phrase is a true entity/phrase that is annotated in the reference output file, and a predicted entity/phrase is regarded as correct if and only if it matches exactly the gold entity/phrase (*i.e.*, both their *boundaries* and *sentiment* are exactly the same for the sentiment analysis task, and both their *boundaries* and *phrase types* are exactly the same as for the phrase chunking task).

Finally the F score is defined as follows:

$$\texttt{F} = \frac{2}{1/\texttt{Precision} + 1/\texttt{Recall}}$$

Note: in some cases, you might have an output sequence that consists of a transition from `O` *to* `I-negative` *(rather than* `B-negative`*). For example, "*`O I-negative I-negative O`*". In this case, the second and third words should be regarded as one entity with negative sentiment. Similarly for the chunking task.*

You can use the evaluation script shared with you to calculate such scores. However it is strongly encouraged that you understand how the scores are calculated.

(10 points)

3 Part 3 *(20 points)*

- Write a function that estimates the transition parameters from the training set using MLE (maximum likelihood estimation):

$$q(y_i|y_{i-1}) = \frac{\texttt{Count}(y_{i-1}, y_i)}{\texttt{Count}(y_{i-1})}$$

  Please make sure the following special cases are also considered: $q(\texttt{STOP}|y_n)$ and $q(y_1|\texttt{START})$.

  (5 points)

- Use the estimated transition and emission parameters, implement the Viterbi algorithm taught in class to compute the following (for a sentence with n words):

$$y_1^*, \ldots, y_n^* = \arg\max_{y_1,\ldots,y_n} p(x_1, \ldots, x_n, y_1, \ldots, y_n)$$

  For all datasets, learn the model parameters with `train`. Run the Viterbi algorithm on the development set `dev.in` using the learned models, write your output to `dev.p3.out` for the four datasets respectively. Report the precision, recall and F scores of all systems.

  Note: in case you encounter potential numerical underflow issue, think of a way to address such an issue in your implementation.

  (10 points)

4 Part 4 *(20 points)*

- The HMM discussed in class makes a simple first-order assumption, where the next state only depends on the previous state in the generative process. However, it is possible to extend the model discussed in class to have second-order dependencies. In other words, the HMM can be parameterised in the following way:

$$p(x_1, \ldots, x_n, y_1, y_2, \ldots, y_n) = \prod_{i=1}^{n+1} q(y_i|y_{i-2}, y_{i-1}) \cdot \prod_{i=1}^{n} e(x_i|y_i)$$

  where we define $y_{-1} = y_0 = \texttt{START}$ and $y_{n+1} = \texttt{STOP}$.

  In other words, the transition probabilities are changed from $q(y_i|y_{i-1})$ to $q(y_i|y_{i-2}, y_{i-1})$ now. Describe the Viterbi algorithm used for decoding such a second-order HMM model and implement it. In other words, describe and implement the dynamic programming algorithm that computes the following efficiently for such an HMM:

$$(y_1^*, y_2^*, \ldots, y_n^*) = \arg\max_{y_1,y_2,\ldots,y_n} p(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n)$$

  Train your models on `EN/train` and `FR/train`. Run your new decoding algorithm on the development sets `EN/dev.in` and `FR/dev.in` only. Write the outputs to `EN/dev.p4.out` and `FR/dev.p4.out`. Report the precision, recall and F scores for the outputs for both datasets.

Hint: How to train your model under the new assumptions? How to learn the new transition parameters? Any changes to the emission parameters? What will be the time complexity of the new Viterbi algorithm?

(20 points)

5 Part 5 – Design Challenge *(20 points)*

- Now, based on the training and development set, think of a better design for developing improved NLP systems for tweets using any model you like. Please explain clearly the model/method that you used for designing the new systems and provide clear documentations on how to run the code. We will check your code and may call you for an interview if we have questions about your code. Please run your system on the development set `EN/dev.in` and `FR/dev.in`. Write your outputs to `EN/dev.p5.out` and `FR/dev.p5.out`. Report the precision, recall and F scores of your new systems for these two languages.

  Note that the two systems designed for the two NLP tasks are allowed to be different.

  (10 points)

- We will evaluate your systems' performance on two held out test sets `EN/test.in` and `FR/test.in` respectively. The test sets will only be released on 6 Dec 2018 at 5pm (72 hours before the deadline). Use your new system to generate the outputs. Write your outputs to `EN/test.p5.out` and `FR/test.p5.out`.

  We will also evaluate your system's performance on yet another two held out test sets `EN/test2.in` and `FR/test2.in`. These two test sets will only be released to you after the deadline. The system that achieves the overall highest F score on these test sets will be announced as the winner.

  (10 points)

Hints: Can we handle the new words in a better way? Are there better ways to model the transition and emission probabilities? Or can we use a discriminative approach instead of the generative approach? Perhaps using Perceptron?[1]*. Any other creative ideas?*

Items To Be Submitted

Upload to eDimension a single ZIP file containing the following: (Please make sure you have only one submission from each team only.)

- A report detailing the approaches and results
- Source code (.py files) with README (instructions on how to run the code)
- Output files
  - `EN/`
    1. `dev.p2.out`

[1] http://www.aclweb.org/anthology/W02-1001

2. dev.p3.out
3. dev.p4.out
4. dev.p5.out
5. test.p5.out

- FR/
  1. dev.p2.out
  2. dev.p3.out
  3. dev.p4.out
  4. dev.p5.out
  5. test.p5.out
- CN/
  1. dev.p2.out
  2. dev.p3.out
- SG/
  1. dev.p2.out
  2. dev.p3.out