SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

50.007 Machine Learning, Fall 2020

Design Project

Due 11 Dec 2020 (Week 13 Fri), 5pm

This project will be graded by Joel Ong
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 update your group information on the previously posted Google sheets if you still haven't.
- 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 6 Nov 2020 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 after that.
- 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

Many start-up companies are interested in developing automated systems for analyzing sentiment information associated with social media data. Such sentiment information can be used for making important decisions such as making product recommendations, predicting social stance and forecasting financial market trends.

The idea behind sentiment analysis is to analyze the natural language texts typed, shared and read by users through services such as Twitter and Weibo and analyze such texts to infer the users' sentiment information towards certain targets. Such social texts can be different from standard texts that appear, for example, on news articles. They are often very informal, and can be very noisy. It is very essential to build machine learning systems that can automatically analyze and comprehend the underlying sentiment information associated with such informal texts.

In this design project, we would like to design our sequence labelling model for informal texts using the hidden Markov model (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.

The files for this project are in `EN.zip`, `SG.zip`, `CN.zip` (the latter two will be available on later, 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 one token per line with token and tag separated by space and a single empty line that separates sentences.

For `EN`, the tags are phrase types. The format is:

```
I B-NP
just B-ADVP
felt B-VP
another B-NP
aftershock I-NP
a B-NP
few I-NP
seconds I-NP
ago B-ADVP
. O
```

where labels such as `B-VP, I-NP` are used to indicate **B**eginning and the **I**nside of phrases, and `O` indicates the token is **O**utside of any phrase. The tags like `NP, VP, ADVP` indicate noun phrase, verb phrase, adverb phrase, etc.

For `SG` and `CN`, the tags are entities with an associated sentiment.

```
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` 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, there are also tags for `B-negative, I-negative` and `B-neutral, I-neutral`.

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. This includes:

- A phrase-chunking system using annotations provided by others (`EN`).
- A named-entity and sentiment analysis system for two different languages from scratch, using our own annotations (`SG` and `CN`).

1 Part 1 (*15 points*, due 6 Nov 2020 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.

Essentially, we are interested in annotating all major entities together with their sentiment information.

Everyone has been assigned 500 posts to annotate. Please go to **Project > Annotation Links** on eDimension and click the link corresponding to your student ID. Click on "brat" at the top right and Login with

Username	your student ID
Password	your student ID

Start annotating by highlighting the spans that you want to mark. The interface is straightforward to use, but if you have questions there is a manual here: `http://brat.nlplab.org/manual.html`

Detailed instructions on the annotation can be found in **Project > Annotation Guidelines**. Enjoy!

Disclaimer: to grant us the right to re-distribute your annotated data to your 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 Joel Ong 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):

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

(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:

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

(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 `#UNK#`.

Set k to 0.5, implement this fix into your function for computing the emission parameters.

(10 points)

- Implement a simple system that produces the tag

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

for each word x in the sequence.

For all the datasets `EN`, `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:

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

The recall score is defined as follows:

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

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

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.*

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 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.

  (15 points)

4 Part 4 *(20 points)*

- Use the estimated transition and emission parameters, implement an algorithm to find the 3rd best output sequences. Clearly describe the steps of your algorithm in your report.

  Run the algorithm on the development sets `EN/dev.in` only. Write the outputs to `EN/dev.p4.out`. Report the precision, recall and F scores for the outputs for both languages.

  Hint: find the top-3 best sequences using dynamic programming by modifying the original 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 an improved sentiment analysis system for tweets using any model you like. Please explain clearly the model/method that you used for designing the new system. 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`. Write your output to `EN/dev.p5.out`. Report the precision, recall and F scores of your new systems.

  (10 points)

- We will evaluate your system's performance on the held out test set `EN/test.in`. The test set will only be released on 9 Dec 2020 at 5pm (48 hours before the deadline). Use your new system to generate the output. Write your outputs to `EN/test.p5.out`.

  The system that achieves the overall highest F score on the 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? Note that you are allowed to look into the scientific literature for 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] http://www.aclweb.org/anthology/W02-1001

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`

* For groups that do not have any members who know Chinese, we will not grade the output for `CN`, only that the code works.