THE UNIVERSITY OF SYDNEY

ASSIGNMENT

THREE

Viterbi Tagging

Due date: Friday June 3rd, 5pm

This assignment is worth 10% *of your final assessment.*

In this assignment, you will build a named entity recognition (NER) system – a structured classifier for predicting the best sequence of named entity tags for an input sentence. Or, you will survey the state of the art in NER, and additionally evaluate and analyse the output of three existing systems.

Option I (programming)

- Implement the *Viterbi algorithm* for predicting the best tag sequence according to a learnt model.
- Explore possible feature sets and perform experiments comparing them.
- Describe your experiments, results and analysis in a report.

For Option I, your submission should include:
- **your report (~3 pages, not including tables/diagrams, Adobe PDF or Microsoft Word);**
- **a tarball/zipfile containing your code and instructions on how to run it.**

Option II (non-programming)

- Write a survey of at least six influential NER papers.
- Evaluate and analyse the output of three NER systems (provided).
- Describe your experiments, results and analysis in a report.

For Option II, your submission should include:
- **your survey (~3 pages, Adobe PDF or Microsoft Word);**
- **your report (~2 pages, not including tables/diagrams Adobe PDF or Microsoft Word).**

Send your submission to `comp5046@it.usyd.edu.au` before the deadline.

Academic Plagiarism declaration

By submitting this assignment you declare the following:

I declare that I have read and understood the University of Sydney Student Plagiarism: Coursework Policy and Procedure, and except where specifically acknowledged, the work contained in this assignment/project is my own work, and has not been copied from other sources or been previously submitted for award or assessment.

I understand that understand that failure to comply with the Student Plagiarism: Coursework Policy and Procedure can lead to severe penalties as outlined under Chapter 8 of the University of Sydney By-Law 1999 (as amended). These penalties may be imposed in cases where any significant portion of my submitted work has been copied without proper acknowledgement from other sources, including published works, the internet, existing programs, the work of other students, or work previously submitted for other awards or assessments.

I realise that I may be asked to identify those portions of the work contributed by me and required to demonstrate my knowledge of the relevant material by answering oral questions or by undertaking supplementary work, either written or in the laboratory, in order to arrive at the final assessment mark.

I acknowledge that the School of Information Technologies, in assessing this assignment, may reproduce it entirely, may provide a copy to another member of faculty, and/or communicate a copy of this assignment to a plagiarism checking service or in-house computer program, and that a copy of the assignment may be maintained by the service or the School of IT for the purpose of future plagiarism checking.

The Named Entity Recognition Task

Input

Input to the NER task consists of raw text, e.g.:

```
U.N. official Ekeus heads for Baghdad.
```

The typical preprocessing stack for NER includes sentence boundary detection, word tokenisation, part-of-speech tagging and syntactic chunking, e.g.:

```
U.N.          NNP  I-NP  I-ORG
official      NN   I-NP  O
Ekeus         NNP  I-NP  I-PER
heads         VBZ  I-VP  O
for           IN   I-PP  O
Baghdad       NNP  I-NP  I-LOC
.             .    O     O
```

Here, each word has been put on a separate line (with a double newline between sentences). The first item on each line is a word, the second is a part-of-speech (POS) tag, the third is a syntactic chunk tag, and the fourth is the gold standard NER tag.

You will receive a data set where this preprocessing has already been performed.

Output

Output from the NER task consists of a tag for each word, which should be added as a new column, e.g.:

```
U.N.          NNP  I-NP  I-ORG  I-ORG
official      NN   I-NP  O      O
Ekeus         NNP  I-NP  I-PER  I-LOC
heads         VBZ  I-VP  O      O
for           IN   I-PP  O      O
Baghdad       NNP  I-NP  I-LOC  I-LOC
.             .    O     O      O
```

The chunk tags and the named entity tags have the format `I- TYPE` which means that the word is inside a phrase of type `TYPE`. Only if two phrases of the same type immediately follow each other, the first word of the second phrase will have tag `B- TYPE` to show that it starts a new phrase. A word with tag `O` is not part of a phrase.

Do not use gold standard NER tags as features. Your classifier would just learn to output the gold tag, resulting in unrealistically good performance!

Data Set

The data is a collection of news wire articles (1996-7) from Reuters, which was developed for the Computational Natural Language Learning (CoNLL) 2003 shared task. It uses the typical set of four entity types: person (`PER`), organisation (`ORG`), location (`LOC`), miscellaneous (`MISC`).

We will supply a URL separately where the prepared data can be downloaded. By downloading the data you agree 1) to only use it for this assignment, 2) to delete all copies after you are finished with the assignment, and 3) not to distribute it to anybody.

The data is split into subsets:

- training (eng.train) - to be used for training you system;
- development (eng.testa) - to be used for development experiments;
- held-out test (eng.testb) - to be used only once features and algorithms are finalised.

For further information, see `http://www.clips.ua.ac.be/conll2003/ner/`.

Do not download and build the data set from the above URL. It does not contain the raw Reuters data.

Sequence Tagging and the Viterbi Algorithm

Sequence Tagging

Our goal is to predict a tag sequence $t_1^* \dots t_n^*$ given an input sentence $w_1 \dots w_n$:

$$t_1^* \dots t_n^* = \underset{t_1 \dots t_n}{\text{argmax}}\ score(t_1 \dots t_n | w_1 \dots w_n). \tag{3.1}$$

We formulate the score of the tag sequence as the sum over token-level tag scores:

$$score(t_1 \dots t_n | w_1 \dots w_n) \approx \sum_{i=1}^{n} score_{perceptron}(t_i | C_i) \tag{3.2}$$

where each context C_i encodes the active attributes for the corresponding word w_i and $score_{perceptron}(t_i|C_i)$ is obtained using your averaged perceptron implementation from Assignment 2, i.e.:

$$score_{perceptron}(t_i | C_i) = \sum_j w_j f_j(x, y) \tag{3.3}$$

where j is an index into the active attributes encoded by C_i.

To encode tag history, you should include previous predictions in the attributes for each word. For example, you should include a history attribute for the previous prediction $KLASS_{i-1} = X$. This is in contrast to the HMM in Lecture 6 and Section 10.2.2 of Manning & Schütze, where there was an explicit term for the probability of a tag given the previous tag $p(t_i|t_{i-1})$.

Now, we could evaluate Equation 3.2 for all possible taggings $t_1 \dots t_n$ of a sentence of length n, but that would make tagging exponential in the length of the input. For the CoNLL 2003 tag set ($\mathcal{T}$ = {`B- PER`, `I- PER`, `B- ORG`, `I- ORG`, `B- MISC`, `I- MISC`, `O`}), this results in a time complexity of $\mathbf{O}(9^n)$.

The Viterbi Algorithm

The Viterbi algorithm is a dynamic programming algorithm for finding the best sequence of tags in $\mathbf{O}(n \times T^2)$ time and $\mathbf{O}(n \times T)$ space where T is the size of the tag set ($T = |\mathcal{T}|$). Thus, for the CoNLL 2003 task, the Viterbi algorithm has a time complexity of $\mathbf{O}(n \times 9^2)$ and space complexity of $\mathbf{O}(n \times 9)$.

For each test sentence, the Viterbi algorithm first requires filling in a $n \times T$ chart of sequence scores:

1. **for** $i \leftarrow 1$ **to** n **do**
2. **for** $t^k \in \mathcal{T}$ **do**
3. $\delta[i][t^k] = \max_{t^l \in \mathcal{T}} \left(\delta[i-1][t^l] + score_{perceptron}(t_i|C_i) \right)$
4. $\psi[i][t^k] = \underset{t^l \in \mathcal{T}}{\text{argmax}} \left(\delta[i-1][t^l] + score_{perceptron}(t_i|C_i) \right)$
5. **end**
6. **end**

where $\delta[i][t^k]$ stores the *score* of the best tag sequence up to and including tag k at position i, and $\psi[i][t^k]$ stores the *history* (i.e., the tag l at the preceding position $i-1$ that gives $\delta[i][t^k]$).

After the chart is filled in, the Viterbi algorithm works back through the chart to recover the predicted tag sequence:

7. $t_n^* = \underset{t^l \in \mathcal{T}}{\text{argmax}}\ \delta[n][t^l]$
8. **for** $i \leftarrow (n-1)$ **to** 1 **do**
9. $t_i^* = \delta[i+1][t_{i+1}^*]$
10. **end**
11. **return** $t_1^* \dots t_n^*$

Implementation Note

To improve efficiency, it is useful to keep a dictionary of observed tags for each word in the training data. Only considering valid tags reduces the total number of sequences to be evaluated.

Typical NER Features

As mentioned above, you should encode tag history by including previous predictions in your attributes, e.g.:

Condition	Contextual predicate (C_i)
$\forall w_i$	$\text{KLASS}_{i-1} = X$

Other typical attributes for NER include the words and parts of speech in a window of two tokens either side of the current one, e.g.:

Condition	Contextual predicates (C_i)
$\forall w_i$	$w_i = X$, $w_{i-1} = X$, $w_{i-2} = X$, $w_{i+1} = X$, $w_{i+2} = X$
$\forall w_i$	$\text{POS}_i = X$, $\text{POS}_{i-1} = X$, $\text{POS}_{i-2} = X$, $\text{POS}_{i+1} = X$, $\text{POS}_{i+2} = X$

Other Features

You should explore some other possible features for NER. For example, you might consider a few of the following questions:

- how could you handle infrequent words?
- how could you incorporate external lists of common locations or common person names?
- how could you capture common word types in a general way?
- how could you model sentence-level information?
- how could you incorporate tag information for tokens to the right?

Hint: Lecture 7 (Information Extraction) contains a larger list of example NER features.

Precision, Recall and F-score for Sequences

Along with the data set, we will provide you with the `conlleval` perl script for scoring your predicted tag sequences against the gold standard. This script expects data on `stdin` in the format described above. The second-to-last column should contain the gold standard NER tag and the last column should contain your predicted NER tag.

Option I (programming)

The assessment for Option I is about how well you can:

- implement Viterbi, extending your averaged perceptron to handle sequence prediction;
- implement and clearly/concisely describe basic features;
- implement and clearly/concisely describe additional features;
- devise and clearly/concisely describe sound experiments;
- devise and clearly/concisely describe insightful error analysis.

The assessment **is not** about the quality of your code. However, your code may be consulted to verify that it is original and consistent with your report.

Implementation

You are free to use a programming language of your choice to implement the assignment.

Report

The report should describe which features you included, and identify which types of features were most important for your classifier's accuracy. It should also characterise the kinds of errors the system made.

The report **should not** focus on your implementation of the Viterbi algorithm.

Features:

- What features does your tagger implement?
- Why are they important and what information do you expect them to capture?
- Are these new features or can you attribute them to the literature?

Experiments / results:

- Which features are most important?
  e.g., performance of different feature combinations
  e.g., subtractive analysis for many feature subsets
- What is the impact of sequence modelling?
  e.g., performance with and without Viterbi
- How does your system perform with respect to a simple baseline?
  e.g., majority class
  e.g., classification with only obvious features, for instance, words
- How does your system perform with respect to the best results reported at the shared task?

Error analysis:

- What are the typical tag confusions made by your system?
- What are the characteristic errors made by your system?
  e.g., manually characterise the errors on a sample of your predictions
  e.g., hypothesise and/or implement some possible solutions
- Are there problems/inconsistencies in the data set that affect your system's performance?

Although in general the choice of how to present your results is up to you, you *must* include micro-averaged Precision, Recall and F-Measure statistics. Development experiments should use the development data sub set and final results should be reported on the held-out test set.

Option II (non-programming)

The assessment for Option I is about how well you can:

- perform a literature search and clearly/concisely provide an interesting comparison of approaches (survey);
- clearly/concisely demonstrate understanding of features and models (survey);
- clearly/concisely convey insight about relative advantages and disadvantages of approaches (survey);
- devise and clearly/concisely describe sound experiments (report);
- devise and clearly/concisely describe insightful error analysis (report).

Survey of NER

You should choose three system description papers from the CoNLL 2003 shared task, which can be found on the task web page (`http://www.clips.ua.ac.be/conll2003/ner/`) under the **CoNLL-2003 Shared Task Papers** heading. The easiest way to choose is to take those that perform best, though you may also want to choose papers such that you end up with a more diverse set of models and/or feature sets.

You should also choose three other NER papers that represent the current state of the art. The first should be:

> Lev Ratinov and Dan Roth. 2009. Design Challenges and Misconceptions in Named Entity Recognition. In *Proceedings of the 13th Conference on Computational Natural Language Learning*, pages 147–155. `http://www.aclweb.org/anthology/W/W09/W09-1119.pdf`

It is up to you to find and choose the other two papers. One of these can be a paper that is seminal (i.e., represents very early or important work on NER).

The survey should not simply summarise each paper. It will be necessary to describe the approaches, but the survey should focus on comparing features/models and providing insight into the state of the art.

Experiment and Report

You should evaluate and analyse the English output for the three CoNLL systems that you choose for your survey. Output for CoNLL 2003 shared task systems can be found on the task web page (`http://www.clips.ua.ac.be/conll2003/ner/`) under the **CoNLL-2003 Shared Task Papers** heading. Follow the **system output** links. Note that you are only required to evaluate and analyse the output for the English data.

You should follow the report instructions for Option I as much as possible. You won't be able to talk about the features you implemented, but you can compare features for your chosen systems. You should try to answer as many of the experimental questions as possible. And you should perform a thorough error analysis.