CS 4740 Introduction to NLP
Spring 2013
Sequence Tagging & Annotation

Proposal: Due via CMS by Tue, Apr 2nd, 1:25pm
Results and Report: Due via CMS by Tue, Apr 9th 1:25pm

1 Introduction

So far, you have learned two basic building blocks: language modeling and word-sense disambiguation. Now, you are to implement a model that tags relevant information to the text data. For example, **Part-Of-Speech tagging (POS)** annotates a particular part of speech to each word in a given text. As the same word may have different parts of speech based on its syntactic role, POS is not a trivial job. Another famous example in the field of natural language processing is **Named Entity Recognition (NER)**, which typically annotates the property or category of each word such as *person, location*, and *organization*. As expected, these could also be utilized as building blocks for more high-level NLP tasks.

If merely considering these problems as classifications, it is hard to grasp the impact of sequential influence dwelling inherently in natural languages. Temporal structure (*e.g., the previous word of the target word to annotate, the previous sentence of the current sentence, and so on*) plays a key role for appropriate annotation in many cases. In other words, simple multi-class classification will ignore the statistical dependencies across the time domain, which is given intrinsically in natural languages.

This project is an highly open-ended assignment in which you are to implement and experiment a sequential tagging method that models a temporal transition for the proper annotation. Particularly, the goal is to predict the underlying sentiment[1] of each sentence given in the movie review data through your model. You will mainly implement a **Hidden Markov Model (HMM)** for this project. If you decide to use an existing package or toolkit rather than implement a sequence tagger from the scratch, you are expected to perform extensive experiments with varying various parameters.

[1] If you want an introduction to sentiment classification, read the paper: `http://www.cs.cornell.edu/home/llee/papers/sentiment.pdf`

2 Goal and Dataset

Sentiment classification is one of the interesting high-level NLP tasks that parses out the underlying attitude of a speaker or writer given in the text. Traditionally, people tried to classify the document-level polarity (*i.e., positive vs negative*) or the document-level multi scales (*e.g., star-ratings*). Rather than focusing on the global sentiment of the entire document, here you are going to predict the sentiment of each sentence (or possibly smaller phrase).

1. We will provide two sets of IMDB movie reviews[2] written by Dennis Schwartz and Scott Renshaw. Each of the training data consists of 200 and 150 relatively long reviews respectively. The test data consists of 47 reviews for both cases written by Dennis and Scott respectively.

2. Each review starts with the [*review-id/rating*]. The *review-id* corresponds to the name of html file in the IMDB webpage.[3] The *rating* is extracted from the corresponding review. **Note that two authors use different rating scales**.

3. Normally, each line in the datasets corresponds to one sentence in the original review. However, if the original sentence has multiple underlying sentiments (*e.g., via adversative conjunctions*), each part is separated into an independent line so that no controversial senses exist in a single line.

4. For each line, the sentiment tag <*sentiment-tag*> is placed at the end. There are five different tag values such that *2=highly praised*, *1=something good*, *0=neutral or objective*, *-1=something that needs improvement*, and *-2=strong aversion*. These values are erased in the testing data.

5. In order to preserve the *paragraph structure*, we use {} as a prefix to indicate the start of a new paragraph. You may play with the paragraph-level sentiment as an optional extension.

6. The dataset is largely preprocessed: removed all the punctuations and non-alphabetic symbols (even numbers, hyphens, and apostrophes), and changed upper case letters to lower case to facilitate your work. If you have a trouble in understanding a review given in the datasets, see the original review at the IMDB website as described in the footnote. Reading the original reivew at the IMDB website may help the readability for your annotation assignment which belongs to the part one of this project.

7. Since the paragraph information is separately added after achieving the sentence-level sentiments, some of them might be incorrect with respect to the original review given in the IMDB website. It would help all students if you fix those problems when doing the annotation assignment.

[2] The original (not preprocessed) data is downloadable at `www.cs.cornell.edu/people/pabo/movie-review-data`

[3] If the review-id is 27275, you can find the original review at the IMDB webpage www.imdb.com/reviews/272/27275.html

3 Implementation

You should implement either the **Hidden Markov Model (HMM)** or the **Conditional Random Field (CRF)** for this task.

1. Implement the sequence tagging algorithm by yourself. We encourage you to implement HMMs rather than CRFs unless you have a sufficient background knowledge about it. The HMM tutorial given in the footnote could be useful for your understanding and implementation.[4] In case that you decide to implement a CRF, clarify your preparation in the proposal.

2. If you decide to try both, start first with HMMs. You may use any programming language that you would like and any preprocessing tools that you want to use. For instance, it is fine to utilize a toolkit extracting n-gram information for your model.

3. If using an existing HMM/CRF toolkit or package to run your experiments, figuring out the usage is a part of this assignment. If you have a question about the model that you choose, visit the office hours. **In any cases, you need to brainstorm what kind of experiments would be interesting to run given your choice of algorithm**.

4. Similar to the previous project, it could be beneficial to reserve some portions of the training dataset for validation purpose. Describe the details of your experimental designs on the report.

5. **Develop baseline systems to be compared with your own model.** In addition to the trivial baselines such as random guess and dominant prediction[5], you can implement any baseline systems. One simple option is to classify certain words frequently used for each sentiment level, and to perform the prediction based only on whether a test sentence includes those words. Note that baseline systems must be compared to your system in reporting the result and analysis.

4 Scoring and Extension

If available, we are going to launch a Kaggle competition per each author to predict the sentence-level sentiments on the testing dataset. In this case, the output format must consist of only one prediction value out of [-2, -1, 0, 1, 2] for each main line in the test data. For example, if the testing input is given like the following:

$[xxxx1/a-]$
... <>
... <>

[4] `http://nlp.stanford.edu/fsnlp/promo/hmm-chap.ps`
[5] vote uniformly to the most frequent sentiment regardless of the sentence input

... <>

[$xxxx2/c+$]
... <>
... <>
... <>
... <>

your output should only contain 7 lines where each line has a single prediction value. When you upload your predictions to Kaggle, you will see the accuracy on the test set. If the competitions are successfully launched, they will be ended at the project due date.

Here are several optional extensions you can implement and experiment. **Doing at least one extension is mandatory. Having more than one extension will be counted as bonus points with respect to the degree of your implementation and experimental results. Note that any of extensions could be used for Kaggle competition**.

1. Implement a secondary baseline system such as Naive-Bayes model or Support Vector Machine (SVM). For SVM, you could utilize the library such as *svmlight*[6] created by Prof. Thorsten Joachims.

2. Implement a secondary sequence tagging system that is different from your primary implementation. You could try either MEMMs or CRFs. If you want to improve performance over the HMM, CRF is likely to be the right direction.

3. Predict the movie rating via sentence-level and paragraph-level sentiment information. In addition to the training set that has sentence-level sentiments, you will be provided paragraph-level sentiments after finishing the part one of this project. It would be highly interesting to see how the lower-level sentiments will propagate to the upper-level sentiments.

4. Improve your sentence-level prediction task by incorporating paragraph-level sentiments or document-level sentiments (*i.e., the rating information*). In this case, it would be interesting to see how the backward propagation works from high-level sentiments to the lower-level sentiments.

5 Proposal

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

[6] `http://svmlight.joachims.org/`

- Which model (HMM vs CRF) are you planning to implement? (If you try to implement CRF, briefly explain your preparation for understanding the model since we did not cover it in class)
- Explain the algorithmic key points of your model. Especially think about which are hidden variables & observed variables on our setting, and what are the corresponding model parameters.
- Brainstorm which feature you would incorporate to learn emission probabilities. Support your design decisions based on the real examples given in the dataset.
- State which extension are you planning to do. While you can end up with doing different extensions, it will help us to provide you of feedback.

In addition to submitting 1 page proposal document, **you also have to annotate several paragraph-level sentiments given currently as blank in the training and test data**. Through this annotation assignment, you will get better understanding about how to collect the gold-standard labels as well as the structure of our movie review data.

- For each paragraph, there is an empty prefix {} currently in the dataset. You are to tag an appropriate sentiment inside the curly brackets. The tagging value should be one of [-2, -1, 0, 1, 2] with respect to the same criteria given in section 2.4. DO NOT PUT any other characters or whitespaces other than the sentiment value.
- Each team will annotate roughly 10-15 reviews. We will shortly email you the actual parts to annotate as soon as you formed a group via CMS. **Thus form a group as early as possible.** We highly encourage you to annotate them *collectively rather than individaully.* The usual way to get gold-standard labels in NLP is to get several human votes for each example, and finalize it to the most frequent votes. Here we will shortly simulate this process.
- These data will be collected right after the proposal deadline, and will be released again via CMS after merging all the labels from your annotations. **Take your time to figure out the most plausible labels rather than merely averaging the sentence-level sentiments given as training examples. Note that these paragraph-level sentiments could highly contribute to the performance of your system in Kaggle competition as described in the extensions**.

6 Report

You should submit a document of 5-6 pages that contains the following sections: (You can include additional sections if you wish to)

1. **General**

   (a) "Configuration" Clearly describe the parts you have implemented on your own and the packages you used from other libraries. Explain any non-trivial design decisions with the intuition.

   (b) "Data Processing" If you have any motivation that make you preprocess the dataset further, explain the details. (You don't have to explain the data-structures unless they are highly non-trivial largely contributing to the efficiency of your system)

2. **Sequence Tagging Model**

   (a) "Implementation Details" Explain your model and approaches that you desgined through this assignment.

   (b) "Experiments" Describe the motivations and procedures of the experiments that you ran. Clearly state what were your hypotheses and what were your exepectations.

   (c) "Results" Summarize the performancse of your system on both the training 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 with including your observation and analysis.

   (d) "Competition Score" If we launch Kaggle competition, include your team name and the screenshot of your best score from Kaggle.

3. **Extensions**
   Explain the extensions 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. If not, also explain why it was not useful. (You don't have to attach Kaggle competition score again here. Just attach the best score to the section 2.(d))

4. **Individual Member Contribution**
   Briefly explain the contribution of an individual group member. You could report if working loads are unfairly distributed.

7 Guideline

1. This is also a relatively big project if you plan to make several extensions. Due to the size, we suggest that you work in groups of at least four, ideally five. We may allow you to form a group of six students in case that your proposal shows reasonable amount of implementation plan. Recruiting people from various backgrounds could be beneficial for both implementation and analysis for this assignment.

2. IT IS ALLOWED to use other prebuilt sequence-tagging system to implement or fortify your own system. In this case, state clearly in the proposal and the report. Then you will be epxected to perform more experiments than other teams.

3. Don't be ridiculously inefficient. You are not supposed to spend times optimizing your codes, 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 this property.

4. **Form a group via CMS immediately and start annotation assignment as early as possible**. It will bring you an intuition about how to plan your project that is indeed what you have to write as a proposal.

5. **If we launch Kaggle competition, submitting to Kaggle is not optional, but mandatory for every team**. Detailed information about Kaggle competition will be updated via Piazza.

6. Grading

   - Proposal: [20 pts] (10 pts for propsoal, 10 pts for annotation)
   - Implementation: [40 pts]
   - Report: [40 pts]
   - Optional Extensions: [20 pts] (10 pts for the 1st, 10 pts for the rest)

7. What to submit

   - Annotation (txt file via TA's email address)
   - Proposal (pdf file into CMS, hardcopy at class)
   - Source code (only the code that YOU wrote, not for any of the packages that you used).
   - Prediction output (the csv files you submit to Kaggle)
   - The report (pdf file into CMS, hardcopy submission will be notified later)
   - Archive all of these except the proposal, and upload it to CMS.