CS4740/5740 Fall 2019 Project 2
Metaphor Detection with Sequence Labeling Models

Final report: due via Gradescope and CMS
by Friday, 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 **Metaphor Detection (MD)**. See Figure 1 for an example, in which the label associated with each word (whether the word is metaphorically used) is annotated.

The experts started **examining** the Soviet Union with a microscope to study perceived changes.
Rockford teachers are honored for saving a *drowning* student.
You're **drowning** in student loan debt.

Figure 1: Example for metaphorical uses. Metaphorical usages of the target word are bold faced, and literal usages are italicized.

For this project, you will implement the sequence labeling approaches for MD:

- Model 1: **Hidden Markov Model (HMM)**
- Model 2: You are allowed to use the models that you know (e.g., MaxEnt, NB, feedforward neural network) to get the observation probability matrix, **feature engineering** is strongly suggested (e.g., word metaphorical use frequency, pre-trained word embeddings, sentence-level features, etc.) The Workshop on Figurative Language Processing [1] provides good insights on feature selection for this task. You can also refer to J&M book [2]; For decoding, implementation for viterbi is required.

Jurafsky & Martin reading on HMMs and MEMMs can be found in Ch. 8.3 – 8.5. You can find more information about the MD task in [3] and from the paper's reference list.

2 Task and Dataset

The MD task uses the $0, 1$ labeling scheme. Each token is associated with a label 0 if it is not metaphorical, label 1 if it is metaphorically used. Each sentence comes with its MD label and Part-Of-Speech (POS) tags. For example, Table 5 specifies an example sentence in the training data: "hackles", "rising", and "bounced" are labeled as metaphorical word. All other words are non-metaphorical. In order to evaluate your system's performance, you will upload your predictions for the test set to Kaggle.

Word	he	continued	,	hackles	rising	.
MD label	0	0	0	1	1	0
POS tag	PRON	VERB	PUNCT	NOUN	VERB	PUNCT

Word	he	gave	us	a	cheque	,	which	bounced	.
MD label	0	0	0	0	0	0	0	1	0
POS tag	PRON	VERB	PRON	DET	NOUN	PUNCT	ADJ	VERB	PUNCT

Table 1: Examples for the metaphor detection sequence labeling task. Given a sentence, your models should predict the MD label (0 or 1) for each word. The POS tags are also provided, you can try using them for feature engineering to improve your classifiers' performance (optional).

3 Model 1 HMM Implementation

In this section, you will implement a **HMM model** 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. This is fine, **but you should write the HMM code by yourself.**

2. You may calculate a Metaphor Tag sequence probability in a log form. For example, the transition probability for bigram can be calculated as follows:

$$\prod_{i=1,n} P(t_i|t_{i-1}) = exp(\sum_{i=1,n} logP(t_i|t_{i-1}))$$

   It could be beneficial to add weight to the transition matrix (making it less or more important during decoding) – for example, using λ * $logP(t_i|t_{i-1})$ to replace the original $logP(t_i|t_{i-1})$. **Be sure to describe the details of your experimental designs in the report.**

3. 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.

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 programmes to detect plagiarism.**

4 Model 2 Implementation

In this section, you will implement a **Model of your choice** for this task. More specifically, your model should consist of a classifier (e.g., MaxEnt, NB, feedforward neural network) with viterbi decoding at test time. You are not required to do the implementation for the classifier on your own, and you can use other toolkits for this.

1. Similar to section 3, you may use any programming language and any preprocessing tools for this part. You can either use off-the-shelf code for the MaxEnt (i.e. multinomial logistic regression) classifier or write the MaxEnt code yourself. **You must**

implement the viterbi algorithm for the model on your own. If you wrote good code for the HMM portion of the assignment, this should not be too hard.

2. You use the classifier to learn a probability $P(t_i|features)$. You may replace either the emission probability – $P(t_i|w_i)$ or the transition probability – $P(t_i|t_{i-1})$ in HMM with it, or you may replace the emission probability * transition probability – $P(t_i|w_i) * P(t_i|t_{i-1})$ in HMM with it. **Be sure to describe the details of your experimental designs in the report.**

3. **Justify the feature set** you prefer to use for the implementation. Feature engineering is very important for this task. Why do you think these features make sense for this task? Provides the references to particular papers if you've got ideas from there.

4. Remember to use same training and validation split for this section to have a fair comparison with your **HMM** model.

5. We provide some baselines' performance numbers on the validation set in Table below, which can be a target for you to reach. It's okay if your model does not have amazing performance. We value more about the insights behind your model design and feature selection, as well as your analysis for the results.

Model	Precision	Recall	F1
Random Guessing	11.59	49.63	18.79
Lexical Frequency Baseline [3]	**60.79**	47.27	53.18
Our model with simple feature engineering + hard decoding	56.64	**52.26**	**54.36**
Your model	?	?	?

5 Kaggle Competition

We will launch a Kaggle competition for the task. Your submission to Kaggle should be a csv file similar to the **sample_out.csv** file that is included in the project folder. The first line is a fixed header, The first column is the index for the word, and the predicted label for the word. Suppose the first two sentences in the input files are:

Word	he	continued	,	hackles	rising	.
MD label	0	0	0	1	1	0
POS tag	PRON	VERB	PUNCT	NOUN	VERB	PUNCT

Word	he	gave	us	a	cheque	,	which	bounced	.
MD label	0	0	0	0	0	0	0	1	0
POS tag	PRON	VERB	PRON	DET	NOUN	PUNCT	ADJ	VERB	PUNCT

Then your output should look like this:

idx,label
1,0
2,0
3,0
4,1

5,1
6,0
7,0
8,0
9,0
10,0
11,0
12,0
13,0
14,1
15,0
...

We provide an evaluation script in the folder, please refer to README on how to do the evaluation on validation set. When you upload your predictions file you will see the evaluation results (F1 score) on the test data. Note that once the Kaggle competition closes, you cannot make additional submissions to Kaggle.

6 Report

You should submit a short document that contains the following sections. (You can include additional sections if you wish.)

1. **Sequence Labeling Models**

   (a) **Implementation Details:** Explain how you implemented **HMMs** and **Model 2** (e.g. which algorithms/data structures you used, what features are included for Model 2). 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 (e.g. which features you used for your **Model 2**, why?).

   (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:** Explain how you evaluate the models. Summarize the performance of your system and any variations that you experimented with on both the training/validation and test dataset. Put the results into clearly labeled tables or diagrams and include your observations and analysis. **An analysis on feature selection for model 2 is required – e.g. what features help most, why? 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) **Comparing HMMs and your model 2:** Compare your results for your **HMM** and model 2. Which of them performs better? Which of the features play an important role? How is the transition matrix affecting the decoding. Do error analysis. See if you observe certain error patterns that one model makes and the other does not. Try to justify why/why not?

   (f) **Competition Score:** Include your team name and the screenshot of your best score from Kaggle.

2. **Code Pieces for Implementations**

When describing your implementation details (e.g., Viterbi decoding, feature extraction), we require that you also include corresponding code pieces (maybe in the form of screenshot) in the report.

3. **Individual Member Contribution**

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

7 Grading Guide

- (20 pts) Design and implementation of the HMM model.
- (20 pts) Design and implementation of the model 2 (model of your choice), as well as the feature set and how the feature extraction is done.
- (15 pts) Experiment design and methodology
- (15 pts) Error analysis and comparison of **HMM** model with **model of your choice**.
- (25 pts) Report: organization, clarity and including **the corresponding pieces of code for the implementations in the report**.
- (5 pts) **Submission to Kaggle.** In the report, you should add a screenshot of your team's performance on kaggle leaderboard.

8 What to Submit

- 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)
- Submit the report to Gradescope. (Due Oct 19th, 11:59pm)

References

[1] *Proceedings of the Workshop on Figurative Language Processing*, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[2] Dan Jurafsky and James H. Martin. *Speech & language processing.*

[3] Ge Gao, Eunsol Choi, Yejin Choi, and Luke Zettlemoyer. Neural metaphor detection in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 607–613, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.