Assignment 4

CMSC 473/673 — Introduction to Natural Language Processing

Due Friday October 25th, 11:59 PM

<table>
<thead>
<tr>
<th>Item</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assigned</td>
<td>Sunday October 13th</td>
</tr>
<tr>
<td>Due</td>
<td>Friday October 25th</td>
</tr>
<tr>
<td>Topic</td>
<td>Maximum Entropy Models: Feature Design</td>
</tr>
<tr>
<td>Points</td>
<td>120</td>
</tr>
</tbody>
</table>

In this assignment you will gain an understanding of maximum entropy (log-linear) models and appropriate feature design.

As with past assignments, you are to complete this assignment on your own: that is, the code and writeup you submit must be entirely your own. However, you may discuss the assignment at a high level with other students or on the discussion board. Note at the top of your assignment who you discussed this with or what resources you used (beyond course staff, any course materials, or public Piazza discussions).

The following table gives the overall point breakdown for this assignment.

<table>
<thead>
<tr>
<th>Question</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points</td>
<td>40</td>
<td>20</td>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>

**What To Turn In**  Turn in a writeup in PDF format that answer the questions; turn in all requested code necessary to replicate your results. Be sure to include specific instructions on how to build (compile) your code. Answers to the following questions should be long-form. Provide any necessary analyses and discussion of your results.

**How To Submit**  Submit the assignment on the submission site:

[https://www.csee.umbc.edu/courses/undergraduate/473/f19/submit](https://www.csee.umbc.edu/courses/undergraduate/473/f19/submit)

Be sure to select “Assignment 4.”
Full Questions

(Q1) (40 points) Work through the online tutorial at
https://www.csee.umbc.edu/courses/undergraduate/473/f19/loglin-tutorial/
It will take several hours to work through the lessons. Please be sure to read the instructions box for each lesson; it provides crucial context and guidance. In the instructions box for each lesson, there are questions posed; you should think about them as you work through the tutorial, but you do not have to turn in your responses to them (unless the following questions specifically say so). Answer, and turn in your responses for, the following questions.

(a) Why are the initial two features in lesson 1 useful? (That is, why are those features able to discriminate/model anything at all?)

(b) Write out, as a formula, the normalization constant \( Z \) for lesson 1.

(c) Answer the four questions of lesson 2.

(d) Describe how the “log-likelihood” and “matching” games (defined in lessons 1 and 3) are related. When you make a good move in the log-likelihood game, does it seem to help or hurt the matching game? How about vice-versa?

(e) Why is the added feature in lesson 4 important? Was it the only additional feature that could have been added? If so, explain; if not, what was another feature?

(f) Compare and contrast the intended effects of \( \ell_2 \) vs. \( \ell_1 \) vs. no regularization in lesson 8.

(g) Summarize, in your own words, the tradeoff(s) between the number of “tokens” (shapes) seen and \( C \), the regularization coefficient.

(h) Discuss how, and explain why, lesson 9 and 10 differ.

(i) Write out, with a formula, the unnormalized probability of a striped circle in the English context, defined in lesson 11.

(j) Write out, with formulas, the normalization constants for lesson 11.

(k) Describe the end effect(s) of adding features for contexts, as shown in lesson 14. Compare the setup and end results, including optimal log-likelihood and the overall sensitivity of the objective to individual feature weights, in 14 with that of 15. Discuss the (real-world) implications of this. When using log-linear models, what should you do (or not do)?

(l) In the bigram modeling (16) or classification (17) lessons, how well can your model fit the observed data? What are some of the highest and lowest weighted features? How does your answer change as you change the amount of observed data, the regularization, or both? You should play with both lessons, but you only need to answer this question (part (Q1)l) for either lesson 16 or lesson 17. Please indicate which lesson you’re referring to. (Of course, you are welcome to answer for both lessons.)

You may work through and discuss the lessons with others. However, you must answer and write up responses to the above on your own.
Consider a unigram maxent model \( p(z) \propto \exp(\theta^T f(z)) \), defined over \( V \) types. Define \( V \) binary features \( f_k \), one for each type \( k \), as

\[
f_k(z) = \begin{cases} 
1 & \text{if } z = k \\
0 & \text{otherwise.}
\end{cases}
\] (1)

Therefore, there are \( V \) different features; correspondingly, there are \( V \) different weights. Features like these are called lexical features. Notice that we’re indexing both the feature vector and feature weights by the vocabulary items.

(a) Write out a formula for \( p(a) \) (the probability of the character “a”). It must be properly normalized and all dot products must be simplified as much as possible.

(b) If you trained this model’s log-likelihood without any regularization, what are, up to a constant, the optimal weights \( \theta_k^* \)?

(Hint 1: think about other unigram models we’ve covered.)

(Hint 2: the constant in “up to a constant” may be defined with respect to some fixed training data.)

(c) Is the optimal \( \theta^* \) guaranteed to be finite? If not, when will it not be finite?

(Q3) (30 points) Read and write a half page summary and review of Rosenfeld (1994). In addition to discussing the basic methodology and findings of this paper, identify findings you found interesting, surprising, or confusing. What is the overall takeaway (for you) from this paper? As part of your summary, briefly compare the statistical techniques used by Rosenfeld to those used by Church and Hanks (1989).

The full BibTeX citation is

@inproceedings{rosenfeld1994hybrid,
  title={A Hybrid Approach to Adaptive Statistical Language Modeling},
  author={Rosenfeld, Ronald},
  booktitle={Proceedings of the Workshop on Human Language Technology},
  pages={76--81},
  year={1994},
  organization={Association for Computational Linguistics}
}

(While many of the acronyms are defined, a couple aren’t. CSR, mentioned at the beginning of Section 5, stands for “continuous speech recognition.” ARPA—the Advanced Research Projects Agency—was a U.S. government agency that provided some research funding; it was the predecessor to the current DARPA—the Defense Advanced Research Projects Agency.)

(Q4) (30 points) This question guides you through different types of feature design, as related to generatively trained vs. discriminatively trained models. For each of the following scenarios, provide some example features (or feature templates). Describe at least two features/feature templates for each scenario; your description must be both in English (prose) and either mathematically (e.g., with functions, as in (1)) or in sufficient pseudo-code. See Appendix A for an extended example.

---

1 For example, our vocabulary is over the words “foo” and “bar.” There are two features, \( f_{\text{foo}}(z) \) and \( f_{\text{bar}}(z) \), that act accordingly: \( f_{\text{foo}}(\text{foo}) = 1 \), \( f_{\text{foo}}(\text{bar}) = 0 \), \( f_{\text{bar}}(\text{bar}) = 1 \), and \( f_{\text{bar}}(\text{foo}) = 0 \).

2 Single spaced, regular font, and one column is fine.
This question talks about “linguistic annotations” $z_{i,j}$ for a particular word token $w_{i,j}$. Each linguistic annotation $z_{i,j}$ could be, e.g., the word’s part-of-speech tag, or possibly a collection of linguistic properties. As an example of this, in the UD data files, the part-of-speech tags are in the third and fourth columns, while the linguistic properties are in the sixth column. For example, the following (Spanish) UD word $w_{i,4}$ has two types of parts-of-speech tags (columns 4 & 5), morphological features (column 6), and syntactic features (columns 7-9). You could consider any combination of these columns to form $z_{i,j}$. That is, $z_{i,j}$ does not have to be a single label; it could itself have a structure, e.g., $z_{i,4,\text{POS}} = \text{VERB}$ and $z_{i,4,\text{MOOD}} = \text{Cnd}$.  

<table>
<thead>
<tr>
<th>WORD</th>
<th>LEMMA/FORM</th>
<th>POS</th>
<th>POS</th>
<th>MORPH. FEATURES</th>
<th>SYNTAX</th>
<th>Misc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>pediría</td>
<td>VERB</td>
<td>VERB</td>
<td>Mood=Cnd</td>
<td>Number=Sing</td>
<td>Person=3</td>
</tr>
</tbody>
</table>

For this question, you can consider $z_{i,j}$ to be whatever is easiest to help you define the features/model. If it’s helpful to assume that each $z_{i,j}$ is a part-of-speech tag, then you can assume that; if instead it’s helpful to assume that $z_{i,j}$ is a complex collection of annotations, that’s okay too. The point of this question is to get you thinking about feature design, how you can incorporate additional information into your classification model, and the differences between discriminatively trained models vs. generatively trained models.

(a) Define features for a word trigram language model $p(w_1, w_2, \ldots, w_M) = \prod_i p(w_i | w_{i-2}, w_{i-1})$, where each trigram factor is a maximum entropy model: $p(z | x, y) \propto \exp(\theta^\top f(x, y, z))$. (Hint: think of some of the smoothing techniques we studied for language models.)

(b) Given sentences $s_i = w_{i1}w_{i2}\ldots w_{iM}$ and labels $l_i$ for each sentence, define features for a maxent likelihood model $p(s_i | l_i)$ that can be used in a generative classifier of the label.

(c) Given sentences $s_i = w_{i1}w_{i2}\ldots w_{iM}$, their labels $l_i$ and linguistic annotations $z_i = (z_{i,j})_j$ for each word $w_{i,j}$, define features for a maxent likelihood model $p(s_i, z_i | l_i)$ that can be used in a generative classifier of the label.

This classifier optimizes $p(l | s_i, z_i) \propto p(s_i, z_i | l)p(l)$; that is, given both a sentence and its corresponding linguistic annotations as input, predict a label $l$ based on its prior probability and the likelihood of both the sentence and those linguistic annotations (given $l$). That is, the label must be able to explain both the words and linguistic annotations; put another way, from just the label, you should be able to generate (sample) both the sentence and the linguistic annotations.

(d) Given sentences $s_i = w_{i1}w_{i2}\ldots w_{iM}$, their labels $l_i$ and linguistic annotations $z_i = (z_{i,j})_j$ for each word $w_{i,j}$, define features for a maxent likelihood model $p(s_i | l_i, z_i)$ that can be used in a generative classifier that conditions on $z_i$.

This classifier optimizes $p(l | s_i, z_i) \propto p(s_i | z_i, l)p(l)$; that is, given both a sentence and its corresponding linguistic annotations as input, predict a label $l$ based on its prior probability and the likelihood of the sentence (given $l$ and the linguistic annotations). In contrast to (c) the linguistic annotations can help generate the words, but the annotations themselves do not need to be generated. (Let this guide the features you design.)

(e) Given sentences $s_i = w_{i1}w_{i2}\ldots w_{iM}$, their labels $l_i$ and linguistic annotations $z_i = (z_{i,j})_j$ for each word $w_{i,j}$, define features for a discriminately trained classifier $p(l | s_i, z_i) \propto \exp(\theta^\top f(s_i, z_i, l))$. 


Appendix A  Feature Design

A.1  Basic Word/Character Feature Design

Let’s say we were dealing with a generative word unigram language model $p(w_1, w_2, \ldots, w_M) = \prod_i p(w_i)$, where each unigram factor is a maximum entropy model: $p(z) \propto \exp (\theta^T f(z))$. One type of feature template is defined by $[$1$. Another type could involve character $m$-grams, that iterate over the character $m$-grams in the word $z$. Combining these, and using binary lexical word unigram and binary character bi-gram features, result in modeling

$$p(\text{cat}) \propto \exp \left( \theta^{(w)}_{\text{cat}} + \theta^{(c)}_{\hat{c}} + \theta^{(c)}_{\text{ca}} + \theta^{(c)}_{\text{at}} + \theta^{(c)}_{t\$} \right).$$

(2)

This equation has a lot going on: first, the character bigram features extract relevant character bigrams. Second, the “$\hat{c}$” and “$t\$” indicate word start/end boundaries: this helps identify character spans that may be indicative (e.g., past tense verbs in English tend to end in “-ed”). This is similar, but not identical, to how you need start and end types when doing language modeling. Third, the superscripts $(w)$ and $(c)$ distinguish word features from character features: especially for short words, such as “at,” we can a lexical word feature/weight $\theta^{(w)}_{\text{at}}$ separate from a character bigram feature/weight $\theta^{(c)}_{\text{at}}$. 