Your name: ____________________________________________

Your NetID: ____________________________________________

- Please write your name & NetID on the top of every page you use.
- This is a closed-book exam. No cheat sheets or other written notes, text books or electronic devices (cell phones, tablets, laptops, etc.) are allowed.
- Please write your answers in black or blue pen, or pencil, and use the spaces provided.
- **Question types:**
  - Define X: Provide a mathematical/formal definition of X
  - Explain/Describe what X is/does: Use plain English to say what X is/does
  - Compute X: Return X, and show the steps required to calculate it
  - Show/Prove that X is true/false/: This requires a (typically very simple) proof.
  - We expect your answers to be concise.

Your score

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basics, Morphology</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Language modeling</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>POS tagging, HMMs and Sequence Labeling</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Lexical semantics</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td><strong>20</strong></td>
<td></td>
</tr>
</tbody>
</table>
Part 1: Basics, Morphology .................................................. (5 points)

[1 point]  (a) List four main components of the NLP pipeline, and describe what they do in your own words.

Solution: I’d like to see something like the following four steps: tokenization (0.25) - POS tagging (0.25) - syntactic parsing (0.25) - semantic analysis (0.25) (Variations may include other steps such as coreference resolution, discourse parsing, etc.)

[1 point]  (b) Describe Zipf’s Law and discuss its implications for natural language processing.

Solution: Zipf’s law: a few word types are very frequent, long tail of very many very low-frequency word types. (0.5) Implications for NLP: we have to be able to handle unseen events/words in the test data. (0.5)

[1 point]  (c) Separate the words unthinkable and producing into morphemes, and explain what morphological processes are at work in each of these cases.

Solution: un-think-able (0.25): derivational morphology (in both cases) (0.25) produc-ing (0.25): inflectional morphology (0.25)

[2 points]  (d) Draw a finite state transducer that maps English singular nouns to their plural form. In addition to the standard rule that the plural adds an -s to the end of words, your FST should capture the rule that English nouns that end in a consonant followed by -y form the plural by changing the y to ies (party → parties), whereas nouns that end in a vowel (a, e, i, o, u) followed by -y just add an s (day → days). You do not need to capture any other exceptions. You can use the symbols C and V for consonants and vowels (and you can assume that any letter is either a consonant or a vowel).

Solution: Base case: every letter inside the word gets transduced to itself (0.5) Every letter besides y that finishes a word (goes to an accepting state) gets transduced to itself, followed by s (0.5) -Cy ending: C gets transduced to itself, y gets transduced to ies (0.5) -Vy ending: V gets transduced to itself, y gets transduced to ys (0.5). Note that we need three distinct states for the penultimate letter of the word. If somebody gets only one side of a transduction rule right, they get half credit for that rule.
Part 2: Language modeling ......................................................... (4 points)

[1 point]  (a) Describe how a unigram language model computes the probability of a sentence, and explain how you can make sure you obtain a distribution over the sentences in the language.

Solution: Definition: $P(w) = \prod_i P(w_i)$ (0.5)
Use an end-of-sentence symbol (0.25) to account for sentences of varying length (0.25) (don’t subtract points for including a start-of-sentence symbol)

[1 point]  (b) Provide the mathematical definition of a trigram language model. Define how the estimate its parameters (without smoothing).

Solution: $P(w^{(1)}...w^{(n)}) = \prod_i P(w^{(i)}|w^{(i-1)},w^{(i-2)})$ (0.5 – don’t care whether notation uses superscripts or subscripts, as long as definitions are clear)

$P(w^{(i)} = w|w^{(i-1)} = w', w^{(i-2)} = w'') = \frac{freq(w''w'w)}{freq(w'')} (0.5 – again, don’t care about notation details)$

[1 point]  (c) Describe how linear interpolation and absolute discounting can be used to smooth distributions (say, to smooth a trigram model with a bigram model). You can provide mathematical definitions or draw a picture if you’d like to.

Solution: Linear interpolation: a weighted average of two distributions ($\lambda P(\cdot) + (1-\lambda)P(\cdot)$), (0.5 points). Absolute discounting: we take away a constant term from the probability mass of each event (0.5 points). (this may be tricky – be generous in grading).

[1 point]  (d) Describe what the metrics of perplexity and word error rate capture when used to evaluate language models.

Solution: Perplexity: does the model assign high probability to unseen data? (0.5 points)
WER: how well can the LM model be used in a task/how useful is the LM for a task (such as speech recognition)? Alternative answers: how many mistakes does the model make, string edit distance between predicted and correct string. (0.5 points)
(a) Explain what a confusion matrix is, and describe what kind of insights can you get from a confusion matrix that you can’t get from a raw accuracy score.

**Solution:** Confusion matrix: which tag was mistagged as which other tag. (0.5) Qualitative analysis: what kinds of errors are made. (0.5)

(b) Provide the formal definition of a (bigram) HMM, and prove that we can use this model to find the best (most likely) sequence of tags $t$ for a string of words $w$.

**Solution:** $P(w, t) = P(t_1)P(w_1|t_1)\prod_{i=2}^{n}P(t_i|t_{i-1})P(w_i|t_1)$ (0.5 points; it’s okay if they don’t have a special case for the first tag). $t* = \text{argmax}_t P(t|w) = \text{argmax}_t \frac{P(t,w)}{P(w)} = \text{argmax}_t P(t,w)$ (0.5)

(c) Define how to learn the parameters of a bigram HMM from labeled (POS-tagged) text.

**Solution:** $P(w|t) = \frac{C(w,t)}{C(t)}$ (0.5) and $P(t'|t) = \frac{C(t',t)}{C(t)}$ (0.5). Don’t get hung up on notation (e.g. for $C(t,t')$ vs $C(tt')$).
Now, remember that for any given HMM, the forward-backward algorithm allows us to compute the conditional probability of a particular tag $t_i$ in the $i$-th position of a particular sentence $w^{(1)}...w^{(i)}...w^{(n)}$, i.e. $P(t^{(i)} = t | w^{(1)}...w^{(i)}...w^{(n)})$ as well as the conditional probability of a particular tag bigram $P(t^{(i-1)} = t', t^{(i)} = t | w^{(1)}...w^{(i)}...w^{(n)})$.

Can you explain (either in plain English, or just in math) how these conditional probabilities can be used to accomplish our task of learning the parameters of a bigram HMM from a raw text?

**Solution:** We cannot use observed counts, but have to use an iterative approach that goes over the training data multiple times (0.25) and uses expected counts (according to the current HMM) instead of observed counts to compute a new HMM in each iteration (0.25). Expected counts: for each position, the expected counts are given by the conditional probabilities, e.g. $\langle c(t^{(i)} = t) \rangle = P(t^{(i)} = t | w^{(1)}...w^{(i)}...w^{(n)})$ (0.25) and then we have to sum up these counts across the corpus and renormalize to get the new HMM(0.25).

**Viterbi algorithm.** Explain and/or define 1) what the trellis in the Viterbi algorithm is, 2) how you initialize the trellis for Viterbi for an $n$-word sentence, 3) how you fill in each internal cell, and 4) how you obtain the result that Viterbi returns to the user. Assume a bigram HMM.

**Solution:** Initialization: the trellis is an $n \times T$ table (words $\times$ tags) (0.25) in which each cell $[i][j]$ contains the probability of the best (highest probability) tag sequence for the prefix $w_1...w_i$ that ends in tag $t_j$ (0.25). Fill each cell $\text{trellis}[1][t]$ with $\pi(t)P(w_1|t)$. (0.5 – many of you forgot $P(w_1|t)$, so you got only 0.25)

Internal cell: To fill each cell $\text{trellis}[i][t]$, compute $t^* = \arg\max_{t'} \text{trellis}[i-1][t']P(t'|t)$. (0.25). Fill $\text{trellis}[i][t]$ with $\text{trellis}[i-1][t^*]P(t'|t')P(w_i|t)P(w_1|t)$ (0.25) and a backpointer to $\text{trellis}[i-1][t^*]$ (0.25).

Result: compute $t^* = \arg\max_{t'} \text{trellis}[n][t']$ (0.25), follow the backpointers in $\text{trellis}[n][t^*]$ to retrieve best tag sequence (0.25).
(f) Describe how you might annotate the sentence *Roger Ebert went to the University of Illinois at Urbana-Champaign* for named entity recognition (i.e. to extract the names of persons, locations and organizations from text). Discuss what problems your annotators might run into.

**Solution:** 0.25 points: correct BIO labels for the words for some analysis, but no differentiation between different kinds of entities 0.25 points: distinguish different kinds of named entities 0.5 points: discuss “University of Illinois at Urbana-Champaign” : issues may include location vs organization, bracketing

(g) Define MEMMs, and explain why they are better than HMMs for sequence labeling tasks such as named entity recognition.

**Solution:** 0.5 points: mathematical definition $P(t_i|w_i, t_{i-1}) = \frac{\exp(f(t_i,w_i,t_{i-1}))}{\sum_j \exp(f(t_i,w_i,t_{i-1}))}$; 0.5 points: feature functions are richer / handle unknown words better than HMMs, which treat each word as an atomic symbol and can’t handle overlapping/non-independent features.
Part 4: Lexical semantics ......................................................... (3 points)

[1 point] (a) Provide the mathematical definition of pointwise mutual information (pmi), and explain why high pmi is a better indicator than high co-occurrence counts that two words are strongly associated with each other.

Solution: Definition: \( pmi(w, w') = \log \frac{P(w, w')}{P(w)P(w')} \). PMI is high when two words co-occur much more often than we would expect by chance. Co-occurrence counts depend very much on the frequency of the words by themselves. (0.5)

[1 point] (b) Please explain (in plain English) how Brown clusters are induced (you can provide mathematical definitions if you find them helpful). Discuss whether Brown clusters would be useful for POS tagging.

Solution: I want to see something about how each word belongs to a single class (0.25), how we merge classes to maximize the mutual information of adjacent classes (0.25) and that this is not useful for POS tagging (0.25) because it doesn’t capture POS ambiguities (0.25).

[1 point] (c) Explain what we mean by the distributional hypothesis, and explain how this hypothesis is captured by distributional similarities.

Solution: Distributional hypothesis: words that appear in similar contexts are similar to each other (0.5). Distributional similarities measure how similar two words are two each other by representing each word as a vector of the contexts in which it appears (0.25), and capture the similarity of words as the similarity/distance of their vectors (0.25).