Your name:

Machine Learning, Fall 2014, Final Exam

README FIRST:

1. Please write your name in the box above on this task sheet and in the box of the stapled writing sheets!
2. All problems of this exam relate to *trigram models*, a simple kind of model of stochastic symbol sequences or word sequences. Trigram models are explained in the box below, make sure you understand them before you proceed to the tasks. The tasks use the notation introduced in the trigram model explanation box.
3. There are eight tasks, the points of which sum to 100. In addition, Task 5 may yield 5 bonus points and Task 8 up to 20 bonus points. If you reach more than 100 points in total with these bonus points, the excess points will be counted toward the course grade.

Trigram models explained. Here is a brief explanation of a standard machine learning model of a language (say, English), called a *trigram model*. Computing a trigram model of English texts is done in the following steps:

1. Procure a (large) collection of written texts, for instance all of the English Wikipedia articles. This makes for the (raw) training data D_{raw}.
2. Define a *vocabulary* V_0, a set of English words and interpunctuation symbols (like !, ?, .) that you want to cover by your model. This set may easily have a size of 100,000 or more (because it includes proper and geographic names, grammatical flexation forms, etc.). Replace any word in your training data D_{raw} which is not in V_0 by an artificial placeholder word, like `unknownWord`. Add this placeholder word to your vocabulary to get $V = V_0 \cup \{\texttt{unknownWord}\}$.
3. Concatenate all your training texts (with words not in V_0 replaced by `unknownWord`) into one long sequence $w_1\ w_2\ ...\ w_N$, where each w_i is an element of V. Call this the training data D. Note: for all English Wikipedia articles, N has been estimated to be 3,302,400,000. For simplicity we'll be assuming in the tasks below $|V| = 100{,}000$ and $N = 3{,}000{,}000{,}000$.
4. A *trigram* is a sequence of (any) three words. Thus $w_1\ w_2\ w_3$ is the first trigram in D, $w_2\ w_3\ w_4$ is the second trigram in D, etc. The set of all possible trigrams over V is $V^3 = \{(u, v, w) \mid u, v, w \in V\}$. We will call this set T. Let $t_1, t_2, ..., t_{|V|^3}$ be an enumeration (ordered listing) of T.
5. A *trigram language model* $\mathcal{M}$ is simply a representation of a probability distribution over T. Since T is finite, mathematically speaking this distribution is a probability vector p of dimension $|V|^3$. For the i-th trigram $t_i = (u, v, w)$ we write $P(t_i)$ or $P(u, v, w)$ or $p(i)$ to denote the probability of t_i assigned by the model $\mathcal{M}$.

Task 1 (creating awareness that trigrams are not trivial) The most straightforward way to obtain a trigram model $\mathcal{M}$ from our Wikipedia-based D is to do a frequency count: for any $t \in T$, let $\#t$ denote the number of occurences of t in D. Then $\mathcal{M}$ is a map $\mathcal{M}: T \rightarrow [0,1]$, $\mathcal{M}(t) = \#t / (N-2)$. A convenient representation of this map is to write it as a probability vector p, where $p(i) = \mathcal{M}(t_i)$. Questions:

(a, 2 pts) What is the ratio of nonzero over zero entries in p? (an upper bound stated in 1-digit precision is good enough).
(b, 2 pts) Give a similar ratio for the case when not trigrams, but bigrams (sequences of length 2) would be used.
(c, 2 pts) Give a similar ratio for a model based on "unigrams" (sequences of length 1).
(d, 2 pts) Give a rough lower bound of the storage size (in bytes) needed to represent the trigram distribution p, assuming only the nonzero entries $p(i)$ of p are stored in the format of a pair $(i, \log(p(i)))$, where the $\log(p(i))$ component is stored in a 32-bit precision format. (Note: $\log_2(10^{15}) \approx 50$; a byte corresponds to 8 bits).

Task 2 (basic uses of a trigram model) Assume a trigram model $p(i) = \mathcal{M}(t_i)$ is given. This model may be used to model many interesting probabilities relating to word sequences, if one makes the following independence assumption:

(*) $P(X_n = w_n \mid X_1 = w_1, ..., X_{n-1} = w_{n-1}) = P(X_n = w_n \mid X_{n-2} = w_{n-2}, X_{n-1} = w_{n-1})$.

That is, word sequences ("texts") are seen as a Markov chain of order 2, where each word depends only on the two previous words.

Questions:

(a, 12 pts) Using (*), give an analytical expression of the probability $P(X_1 = a, X_2 = b, X_3 = c, X_4 = d, X_5 = e)$ to observe a 5-step sequence *abcd*, in terms of the trigram distribution p.
(b, 6 pts) Using (*), give an analytical expression of the probability $P(X_3 = c \mid X_1 = a, X_2 = b, X_4 = d, X_5 = e)$ in terms of the trigram distribution p.

You may use shorthand notation like $P(a\ b\ c\ d\ e)$, $P(c \mid a\ b\ d\ e)$.

Task 3 (a Bayesian network representation of word sequence statistics) (6 pts) Assuming (*), represent the (in-)dependencies between 5 successive word observations $X_1, X_2, ..., X_5$ in the graphical format of a Bayesian network. Deliverable: a directed graph with nodes $X_1, X_2, ..., X_5$.

Task 4 (zero probabilities are bad) (15 pts) In Task 1 you found out that the straightforward frequency count model $\mathcal{M}: T \rightarrow [0,1]$, $\mathcal{M}(t) = \#t / (N-2)$ leads to a large number of zero probabilities $p(i) = 0$.

Explain why this model $\mathcal{M}$ will be useless in many practical applications. Deliverable format: 2 – 5 sentences in plain English, maybe garnished with a formula or two, giving an example of a computation needed in applications which will be rendered impossible or invalid by (too many) occurrences of $p(i) = 0$.

Task 5 (15 pts, plus max 5 bonus pts) (a quickfix for the zero probability problem) Your findings from task 4 demonstrated that a useful model $\mathcal{M}: T \rightarrow [0,1]$ should assign nonzero probabilities to all $t \in T$. Describe in formal terms a patch for the frequency count model $\mathcal{M}(t) = \#t / (N - 2)$ which ensures that $p(i) > 0$ for all i. Your patch should be more sophisticated than the all-too-dirty patch of just adding some small $\varepsilon > 0$ to all zero components of the basic frequency count model. Hint: one approach is to work in what you found out in Task 1a, 1b, 1c. Any valid solution gives 15 points; really nice solutions get bonus points.

Task 6 (20 pts) (a more sophisticated solution to the zero probability problem) A professional-level machine learning expert would approach the task of learning a trigram model $\mathcal{M}: T \rightarrow [0,1]$ in two stages: (i) fix a type of parametric model and define a loss function, (ii) design an optimization algorithm which finds parameters that minimize the loss. A parametric model type which avoids zero probabilities by design is the softmax: define p_θ by $p_\theta(i) = \exp(a_i) / \sum_i \exp(a_i)$. The parameter set θ of this model is the collection of the $a_i \in \mathbb{R}$. Your task: formally specify a reasonably-looking loss function $L(D, \theta)$, and explain your ideas behind this formula in plain English.

Task 7 (for the fun of it) (3 pts) Give a sequence of 8 English words, such that each subsequence of length 3 would have a high trigram model probability (i.e., it could easily occur in a real text), but such that the overall sequence is nonsensical and/or not grammatical.

Task 8 (HMM versus trigram model) A natural approach for a model of English texts would be to train a HMM on the training sequence D. The set of observables would be V.

(a, 15 pts) How many hidden states would you think should a good HMM trained on D have? Give an order of magnitude for the size $|Q|$ of the hidden state set that you would expect to give a good model, and justify your number in plain English (target size: 3-5 sentences).
(b, pure bonus question, up to 20 bonus points) Compare a HMM text model with a trigram based text model. Choosing the criteria for comparison is up to you: for instance model size, training time, applications, model accuracy, ... Format of delivery: a bullet point list of comparative insights, each bullet point a single sentence (or two sentences at most). Up to 10 such bullet points. Each valid, transparently formulated insight gets 2 bonus points.

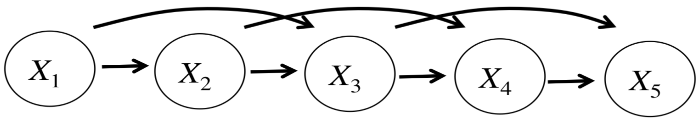
Solution for task 1. (a) D contains about 3 × 10^9 trigrams, p has 10^15 entries, so there can't be more than a fraction of 3 × 10^9 / 10^15 = 3 × 10^–6 nonzero entries: about 3 in a million entries at most are nonzero. (b) D contains about 3 × 10^9 bigrams, p now has 10^10 entries, so the ratio of zeros over nonzeros is now at most 3 out of 10. (c) Assuming that V only contains tokens that occur in D, there will not be a single zero in p. (d) i takes values between 1 and 10^15, thus needing log2(10^15) ≈ 50 bits ≈ 6 bytes, so we get about 3 × (6 + 4) × 10^9 bytes, that is 30 GB.

The message from your answers to Task 1 is that (a) there is a serious curse of dimensionality problem, and (b) that there is a serious model size problem – just storing p in a naïve, direct way would be very expensive in terms of memory capacity. Solving these problems is not trivial and we defer a discussion to a later point (bonus exam tasks at end of this task sheet). For the time being assume that a clever model $\mathcal{M}$ is available which allows us to compute $p(i) = \mathcal{M}(t_i)$, such that always $p(i) > 0$.

Solution to task 2.
(a) $P(a\ b\ c\ d\ e) = P(e \mid a\ b\ c\ d)\ P(d \mid a\ b\ c)\ P(a\ b\ c) =$
$P(e \mid c\ d)\ P(d \mid b\ c)\ P(a\ b\ c) = P(c\ d\ e)\ P(b\ c\ d)\ P(a\ b\ c)\ /\ P(c\ d)\ P(b\ c) =$
$P(c\ d\ e)\ P(b\ c\ d)\ P(a\ b\ c)\ /\ \sum_{w\in V} P(wcd) \sum_{w\in V} P(wbc)$.

(b) The probability $P(c \mid a\ b\ d\ e)$ is equal to $P(a\ b\ c\ d\ e) / \sum_{w\in V} P(abwde)$. The enumerator and each of the summands in the denominator can be computed as in (a).

Solution to task 3.



$$\begin{tikzcd}
X_1 \arrow[r] \arrow[rr, bend left] & X_2 \arrow[r] \arrow[rr, bend left] & X_3 \arrow[r] \arrow[rr, bend left] & X_4 \arrow[r] & X_5
\end{tikzcd}$$

Solution to task 4. Example of a solution: Whenever a practical application requires to compute (log) probabilities of an observed word sequence $a_1\ a_2\ ...\ a_K$, and this sequence contains a trigram abc with $p(abc) = 0$, then the model $\mathcal{M}$ will falsely assign a zero probability to $a_1\ a_2\ ...\ a_K$. Since the training data D, despite its large size, only covers a small part of potential test texts, test sequences $a_1\ a_2\ ...\ a_K$ of some length will almost always contain trigrams not in D.

Solution to task 5. (One possibility). Define a unigram-based trigram model $\mathcal{U}(u\ v\ w) = \#u\ \#v\ \#w / N^3$ and a bigram-based trigram model $\mathcal{B}(u\ v\ w) = N\ \#uv\ \#vw / (\#v\ (N-1)^2)$. Linearly mix this with the frequency count model $\mathcal{M}$ to obtain a final trigram model $\mathcal{F}(t) = (\alpha\ \mathcal{U}(t) + \beta\ \mathcal{B}(t) + \gamma\ \mathcal{M}(t)) / (\alpha + \beta + \gamma)$, where the mixing coefficients are all greater than 0. Optimizing the mixing coefficients could be done by a cross-validation.

Solution to task 6. (Example) Let $d(p_1, p_2)$ be a distance function between probability vectors (or a quasi-distance function like the KL divergence). Let p_0 be the probability vector obtained from the simple frequency counting. Define
$L(D, \theta) = d(p_\theta, p_0) + \alpha \sum_i a_i^2$. The first term pulls $p_\theta(i)$ toward the frequency-counted distribution ("empirical distribution") p_0, the second term has two functions: it acts as

a regularizer, and it prevents the parameters a_i to move toward infinity when a gradient descent method is used to minimize the loss. The coefficient α balances between the two components.

Solution to task 7. For instance: "I do not badly ill functioning device costs"

Solution to task 8. (a) A HMM with m hidden states and 100,000 observable symbols has about m^2 + 100,000 * m parameters. The rule of thumb that a model should not have more parameters than about a tenth of the training data points leads to an upper bound defined by m^2 + 100,000 * m = 3 * 10^8. This implies that m << 100,000 and the term m^2 + 100,000 * m is dominated by the second component, so we may ignore the m^2 part, ending up with $m \approx$ 3 * 10^3 = 3,000, i.e. an order of magnitude between 10^3 and 10^4. – Other lines of reasoning may be based on semantical considerations: the hidden state belief vectors can be considered to code for the "meaning" of a word and its grammatical role, and one might want to argue from there (difficult but interesting).