CS 6740/INFO 6300 **Spring 2010**

Lecture 10 – March 2

Lecturer: Prof Lillian Lee *Scribes: Jerzy Hausknecht & Kent Sutherland*

More on Language Models

1 Review: Relevance Model

In the previous lecture, we discussed the idea of *relevance models*, as presented in [Lavrenko & Croft 01]. For each query, a language model for relevance is constructed. The final product is a language model based on a collection of documents. The final model estimation details were very similar to *query likelihood*, even though the *relevance model* was derived from the ideas in [Robertson & Spärck Jones 76]. The *relevance model* work reinforces the "importance of being reversed" (Lafferty and Zhai's phrasing), as assigning likelihood to a query from a document-based model yields better statistical estimates than assigning likelihood to a document based on a query-based model.

While such an approach to deriving language models "works" mathematically, the intuition for it severely stretches the original assumptions made in the Robertson & Spärck Jones approach. This discussion starts with a mostly clean slate and attempts to justify the language model approach to query likelihood in a more intuitive fashion.

2 Warmup Exercise

2.1 Motivation and Notation

We start with one attempt to derive query likelihood. Suppose we are given a finite set of "generating" language models, where each language model can be considered a "topic." Assume that different topics can overlap each other. Consider the following generative story: a user randomly selects a topic language model, and then generates the document d according to that language model. Let T_d be the topic the user selected. Similarly, T_q is the topic language model chosen by the user to generate the query q. In general, let T_x be a language model that was chosen to generate x. For any given query q, score documents by $P(T_q = T_d|d, q)$. Intuitively, this is the probability that for a given document and query, the language model that generated the query also generated the document. In this scenario, the representation of d and q as strings or word vectors isn't relevant.

2.2 Attempting to Derive Document Probability

Now we would like to find the probability that a particular topic that generated a query q also generated a document d. Unfortunately, we don't actually have these probabilities, so

instead we'll employ "wishful thinking" and pretend that we do. For now, assume that for a document d there exists a topic t^* such that $P(T_d = t^*|d) = 1$. In other words, assume that we know that topic t^* generated document d with absolute certainty. Note that this t^* may be different for different d.

Using this notation, we now score documents based on the following probability: $P(T_q = t^*|d, q)$. We consider $T_q = t^*$ to be independent of d, so we rewrite the previous probability as $P(T_q = t^*|q)$. Notice that is a probability conditioned on q, similar to earlier derivations. Using Bayes' theorem, we can instead condition on the document's topic (which we assume we know) as follows: $\frac{P(q|T_q=t^*)P(T_q=t^*)}{P(q)}$. The expression $P(q)$ in the denominator is document independent, and therefore has no impact when ranking the documents. Additionally, $P(T_q = t^*)$ can be considered a (perhaps uniform) prior that indicates the probability that a random query's language model will equal t^*; it is therefore document independent as well. Thus, the two document independent terms are dropped from the scoring function, leaving the final query likelihood expression: $P(q|T_q = t^*)$, similar to [Ponte and Croft 98].

3 Using Divergence

One issue with the simple model described above is that it doesn't effectively handle the case where one topic is a subset of another. For example, consider two document-derived language models, one for "cats" and the other for "Persian cats." In this instance, a query for "cats" would arguably be relevant to both topics. The problem is that the above model assumes there is a single generating topic t^* that has probability 1, while all other topics have probability 0. This would result in the "Persian cats" document not being retrieved. Clearly a more discriminative means of judging query likelihood is necessary.

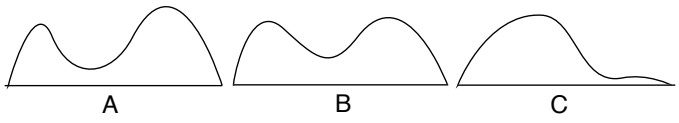

Figure 1: Example distributions. A and B could be considered similar, whereas C is not similar to A or B.

Ultimately, we need a means of comparing the query topic to the document topic that is more expressive than a binary decision. [Lafferty & Zhai 01] used divergence to score distributions, and [Lavrenko & Croft 01] also mentioned using cross-entropy to achieve similar results. Divergence can be thought of as way to formally represent how "close" two distributions are. Instead of considering the probability that the query and document were generated by the same language model, documents are scored by the divergence between T_q and T_d. For example, consider the distributions in figure 1. Distributions A and B might represent two language models that would considered "close" to each other, whereas C would not be

"close" to A or B. This introduces significantly more expressiveness in scoring compared to a binary decision.

Consider these topics as distributions over single terms (as opposed to phrases, sentences, or documents; this way, we don't have to worry about length issues). In other words, each word in the vocabulary is assigned a probability of being chosen by the language model. Let $\vec{\Theta}_d$ be the distribution over individual words representing the topic that generated a document, and let $\vec{\Theta}_q$ be the distribution over individual words representing the topic that generated a query. Both $\vec{\Theta}_q$ and $\vec{\Theta}_d$ are $\in \mathbb{R}^m$ ($[0,1]^m$). Since they are probability distributions, we can be sure that $\Theta_d[\cdot] = \Theta_q[\cdot] = 1$ (recall that $x[\cdot]$ indicates the sum over all terms in x).

Now that we have our topics represented as distributions, how do we score these distributions against each other? Since the distributions are discrete, one basic approach would be to treat the distributions as vectors and score using Euclidean distance. This approach is overly simplistic; for the remainder of this section we will consider using Kullback-Leibler divergence as a scoring function.

3.1 Kullback-Leibler Divergence

The Kullback-Leibler divergence (KL divergence, or just KLD) between $\vec{\Theta}_q$ and $\vec{\Theta}_d$ is defined as follows:

$$KLD(\vec{\Theta}_q || \vec{\Theta}_d) := \sum_j \Theta_q[j] \log \frac{\Theta_q[j]}{\Theta_d[j]}$$

We define $0 \log 0 = 0$ by limiting arguments.

3.1.1 Properties

First, note that if the two distributions are equal, then $\log \frac{\Theta_q[j]}{\Theta_d[j]} = \log 1 = 0$ for all j. The result is a KL divergence of 0. Also important is that the definition of KL divergence is asymmetric, so the order of the distributions matters. The first distribution is considered the "true," or reference distribution. Wikipedia explains that KL divergence measures "the expected number of extra bits required to code samples from $\vec{\Theta}_q$ when using a code based on $\vec{\Theta}_d$, rather than using a code based on $\vec{\Theta}_q$."

The asymmetry of the KL divergence can be demonstrated quite easily. Suppose for some j, $\Theta_q[j] = 0$ and $\Theta_d[j] > 0$. Zero is contributed to the sum; there must exist some j' such that $\Theta_q[j'] > \Theta_d[j']$ (since both distributions sum to 1), and therefore $\log \frac{\Theta_q[j']}{\Theta_d[j']} > 0$. However, consider the case where there exists a j such that $\Theta_q[j] > 0$ and $\Theta_d[j] = 0$. This results in division by zero, and therefore KL divergence is undefined in this case. Continuing with Wikipedia's explanation, this means that it is impossible to express a signal or code generated from $\vec{\Theta}_q$ in terms of $\vec{\Theta}_d$, as $\vec{\Theta}_d$ simply does not have the necessary information to do so. In other terms, according to the reference distribution $\vec{\Theta}_q$, v_j is possible, but $\vec{\Theta}_d$ assigns this possible event zero probability, thus making it impossible to code for v_j. This

asymmetry means KL divergence is not a true metric, as a metric must satisfy the three properties of symmetry, non-negativity, and the triangle inequality. Despite this, it is still a useful scoring function, as we have shown that this asymmetry makes intuitive sense.

3.1.2 Non-Negativity of KLD

Although KLD is asymmetric, it does satisfy the non-negativity requirement of formal metrics. This can be shown by first expanding the log: $KLD(\vec{\Theta}_q||\vec{\Theta}_d) = \sum_j \Theta_q[j] \log \Theta_q[j] - \sum_j \Theta_q[j] \log \Theta_d[j]$. We then rewrite as $-(-\sum_j \Theta_q[j] \log \Theta_q[j]) + (-\sum_j \Theta_q[j] \log \Theta_d[j])$. To simplify things, think of this expression as $-A + B$. The term A is the entropy of $\vec{\Theta}_q$, while B is the cross-entropy of $\vec{\Theta}_q$ and $\vec{\Theta}_d$. The cross-entropy B must be at least as large as the entropy A. The result is $KLD(\vec{\Theta}_q||\vec{\Theta}_d) \geq 0$.

3.2 KLD with Relative Frequency Estimates

Continue to consider KLD as a difference of log terms. Term A, the entropy of $\vec{\Theta}_q$, is document independent. Therefore, for relative ranking of documents, we only need consider the cross-entropy (term B). As a final detail, we flip the sign of term B so that lower cross-entropy results in a higher similarity score. We now need to incorporate this expression with our language model.

One means of estimating the language model distributions is to use the relative frequency estimate for the query language model, defined as: $\hat{\Theta}_q[j] = \frac{q[j]}{q[\cdot]}$. When combined with KL divergence, the scoring function becomes:

$$\frac{1}{q[\cdot]} \sum_j q[j] \log \Theta_d[j] = \frac{\sum_j \log \Theta_d[j]^{q[j]}}{q[\cdot]} \stackrel{\text{rank}}{=} \sum_j q[j] \log \Theta_d[j]$$

Notice that the numerator of this score is the standard query log likelihood that we have seen before.

4 Other Uses of Language Models

Language models are very useful in information retrieval, but also in other fields of language processing.

4.1 Speech Recognition

Using language models in speech recognition led to considerable improvements. [Ponte & Croft 98] actually credit the speech recognition community with the origin of the phrase "language model." Generally stated, the goal of speech recognition is to convert an acoustic signal a into a sequence of words w. Stated in probabilistic terms, the goal is to find:

$$\arg\max_{\text{sequences } w} P(w|a)$$

As we have done before, Bayes' theorem can be used to condition on w rather than a, where $P(w)$ is determined using a language model.

$$\arg\max_{\text{sequences } w} \frac{P(a|w)P(w)}{P(a)}$$

The utility of language models is evident given the sentence "It's hard to wreck a nice beach." Phonetically, this is very similar to "It's hard to recognize speech." Without a language model it would be hard to discriminate between these two sentences, even though one is much more probable in general speech. By using language models as a prior probability we can ensure that the second sentence will receive a much higher score.

4.2 Translation

Another field of language processing where language models are used is in translation. Consider translating a French statement into English.

$$\arg\max_{\text{sequences } e} P(e|f) = \arg\max_{\text{sequences } e} \frac{P(f|e)P(e)}{P(f)}.$$

$P(f|e)$ is a reverse translation probability, $P(e)$ is a measure of whether or not the translation is "good" English (using LMs), and $P(f)$ can be ignored since it is translation independent. Although this scoring function contains more terms, it is useful as it separates the fluency of translation (represented as $P(e)$) from meaning preservation (represented as $P(f|e)$). This insight was used to build the Candide system [Berger et al. 1994].

5 Language Models with Structure

Previously, we have not been specific about whether documents and queries are represented as word vectors or strings. We now consider language models over all word sequences, as documents have structure which a bag of words representation ignores. This raises a question: how do we estimate $P(w_1 \dots w_l)$ where $w_k \in \{v_1, \dots, v_m\}$? We assume fixed length ℓ for simplicity.

5.1 Counting and n-grams

The first idea is to just count. However, we want to count more than just individual word occurrences. The question then becomes what do we count? Do we count the appearance of individual sentences? This clearly would not work, as the sentence "The 24 Hours of Adrenalin Solo World Championship is often a grueling showcase for the world's toughest bikes and riders" may only appear once in the entire corpus. A compromise is to count n-grams, or consecutive sets of n words. For example, "grueling showcase" would be a 2-gram (bigram), or "bikes and riders" would be a 3-gram (trigram). This helps to alleviate the problem of data sparsity.

Currently, the largest publicly-available n-gram corpus is the Google n-gram corpus, which was constructed from 1 trillion words of English text. This corpus contains over 13 million unique words occurring at least 200 times and has all n-grams up to $n = 5$ counted. As a frame of reference, the New York Times has a vocabulary size of approximately 75 thousand words. Given 13 million words, there are 169 trillion possible bigrams. However, there are only 315 million bigrams in the Google corpus, a tiny percentage of the bigrams that are theoretically possible. Moreover, there is a very basic limitation of the bigram model. Given a 1 trillon word corpus, there is a maximum of 1 trillion bigrams that can ~~theoretically possible~~, since bigrams are created by taking directly neighboring terms. This means that at least 168 trillion of the possible bigrams must be missing.

One attempt to make progress in spite of these difficulties is to simplify the model, conditioning only on the two words preceding a given word. This is the trigram assumption, and is stated as $P(w_1 \dots w_l) = \prod_{k=1}^{l} P(w_k|w_1 \dots w_{k-1}) \approx \prod_{k=1}^{l} P(w_k|w_{k-2}w_{k-1})$. It is also possible to condition only on the word immediately preceding a given word (bigram assumption). Context is dropped to make the problem tractable, but the problem of data sparsity remains. Approaches to dealing with data sparsity will be discussed in future lectures.

6 Questions

6.1 Kullback-Leibler Divergence

a) As mentioned in section 3, one simple way of scoring two distributions would be to treat the two distributions as vectors and take the dot product. Assuming that both distributions are discrete with a large number of terms, what problem arises?

b) Each column in the table below is a set of term counts representing three different topics.

	cat	**dog**	**Persian cat**
cat	25	4	5
fluffy	10	7	8
dog	2	20	1
fur	6	7	10
hairball	15	2	5
bone	2	10	1
bark	3	10	1
meow	10	2	3
Persian	2	1	30

1) Build a unigram language model for each topic based on these term counts. Use relative-frequency estimates, with no smoothing.

2) Pretend we are searching for documents about cats. Let $\vec{\Theta}_q$ be the "cats" language model you have just built. Calculate the Euclidean distance and negative cross-entropy

$(\sum_j \Theta_q[j] \log_2 \Theta_d[j])$ between the "cats" language model and the other two language models.

c) Kullback and Leibler originally defined their divergence slightly differently from the currently accepted definition. Kullback-Leibler divergence is defined as

$$D_{KL}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

They originally proposed a symmetric version of this: $D_{KL}(P\|Q) + D_{KL}(Q\|P)$. Note that this is similar to Jensen-Shannon divergence, a symmetric and smoothed measure of divergence.

[Lafferty & Zhai 01] begin with the standard (non-symmetric) version of Kullback-Leibler divergence for their loss function. Intuitively, what could be some potential problems with using the symmetric version to measure loss instead?

6.2 n-gram Smoothing

a) Bigrams and trigrams incorporate language structure while keeping inference tractable. What problems are there with increasing this to larger n-grams, such as 5-grams, or even 10-grams?

b) **(Mental exercise, no answer provided)** Due to the sparse data problem some measure of smoothing is necessary. One solution is to simply increment all n-gram counts by 1 and renormalize the distributions. This will ensure that the KL divergence is always defined. However, this fails to take into account the probability of individual words occurring in a given corpus. For example, if the bigrams "defenestration pants" and "smelly pants" both didn't occur in the corpus, they would be assigned the same likelihood even though the word "defenestration" is much less common than "smelly." One obvious improvement is to take individual word probabilities into account. However, this approach has a weakness in that it fails to account for the parts of speech of various words. For example, the bigram "the is" would be given a relatively high estimated probability as both "the" and "is" are very common words, even though structurally the bigram is gibberish. Therefore, the approach can be further refined by taking into account the likelihood of parts of speech that would form a bigram. Is such an approach reasonable from a theoretical standpoint, and can it be made tractable? Are there any major deficiencies to this approach?

7 Answers

7.1 Kullback-Leibler Divergence

a) This approach treats the distributions as if they were points in Euclidean space, which leads to the problem of measuring distance in high dimensions (the "curse of dimensionality"). The discriminative power of scoring this way is drastically reduced as the

number of dimensions goes up. This is because high-dimension Euclidean space causes most of the volume of the space to lie in the "corners" of each dimension. This means most points in the space are located in the "corners" also. In these high dimensions, each of the corners are all almost equally far away from each other, making distance a very unreliable measurement.

b) The first table represents the language models for each topic. The language models are simply the term frequencies divided by the total number of terms seen.

	cat	**dog**	**Persian cat**
cat	0.333	0.063	0.078
fluffy	0.133	0.111	0.125
dog	0.027	0.317	0.016
fur	0.08	0.111	0.156
hairball	0.2	0.032	0.078
bone	0.027	0.159	0.016
bark	0.04	0.159	0.016
meow	0.133	0.032	0.047
Persian	0.027	0.016	0.469

Once we have the language models, we can find the Euclidean distance and cross-entropy between "cat" and "dog", and "cat" and "Persian cat." Notice that when using Euclidean distance, the language models for "cat" and "dog" are scored as being more similar than "cat" and "Persian cat," a result we probably were not expecting. Compare this to cross-entropy, where "cat" and "Persian cat" are more similar than "cat" and "dog." Recall from section 3.2 that we score using negative cross-entropy, and that larger numbers indicate greater similarity.

Intuitively this makes sense, as cross-entropy treats one of the distributions as a sort of weighted average against the other distribution. Euclidean treats each term equally, so the importance of each term is lost.

	cat vs. dog	**cat vs. Persian**
euclidean distance	0.48	0.54
Negative cross-entropy	-4.04	-3.75

c) Intuitively, it seems that measuring KL divergence or cross-entropy should consider the language model of the query q as the "true" distribution. There would not be a reference distribution to score against if a symmetric measurement were used. Another problem goes back to the original "importance of being reversed." In general, the number of words available in a query is much smaller than the number of words in any given document. This makes it must more difficult to approximate a language model for a query, which adds additional uncertainty. Symmetrizing the divergence would simply add this additional uncertainty to the overall score.

7.2 n-gram Smoothing

a) Computationally, working with n-grams is an NP-complete problem. The number of n-grams increases exponentially with n, which makes extracting, storing, and iterating over a set of n-grams intractable for anything but very small values of n. Another problem goes back to data sparsity. Taken to an extreme, considering n-grams with a very large n is no better than considering entire sentences, which was one of the motivations for the n-gram model in the first place.

8 References

1. A.L. Berger, P.F. Brown, S.A. Della Pietra, V.J. Della Pietra, J.R. Gillett, J.D. Lafferty, R.L. Mercer, H. Printz, L. Ureš. The Candide system for machine translation. *Proceedings of the workshop on Human Language Technology, March* (1994) pp. 8-11.

2. A. Franz, T. Brants. "All Our N-gram are Belong to You" Google Research Blog. 03 Aug 2006. Google, Web. 7 Sun 2010. <http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html>.

3. S. Kullback and RA. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, vol. 22, issue 1 (1951) pp. 79-86

4. Lafferty and Zhai. Document Language Models, Query Models, and Risk Minimization for Information Retrieval. *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval* (2001) pp. 111-119

5. Lavrenko and Croft. Relevance based language models. *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval* (2001) pp. 120-127

6. Ponte and Croft. A language modeling approach to information retrieval. *Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval* (1998) pp. 275-281

7. S. E. Robertson and K. S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science and Technology*, vol. 27, no. 3 (1976) pp. 129-146

8. Wikipedia contributors. "Curse of dimensionality." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 30 Mar. 2010. Web. 3 Apr. 2010

9. Wikipedia contributors. "Kullback-Leibler divergence." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 28 Feb. 2010. Web. 8 Mar. 2010