In many applications of natural language processing (NLP) it is necessary to determine the likelihood of a given word combination. We discuss *class based methods* that attempt to abstract over the simple count, in order to tackle problems of data sparsity and support generalization.).

1 Introduction

- Many word combinations are infrequent or do not occur in any given corpus.
- How can we make decisions that depend on these?
- The natural idea is to move to class based estimation. POS tags would be one example of a class based abstraction.

  Note: what about pos of unknown words?
- In this lecture we will discuss *word association models* that will allow for the estimation of the probability of previously unseen word combination using available information on "most similar" words.
- One of the reading/presentation will offer some criticism by showing that the definition of similarity ought to depend on the tasks.

Some Data

- (Essen and Steinbiss, 1992): In a 75%-25% split of the million-word corpus, 12% of the bigrams in the test partition did not occur in the training portion.
- (Brown, DellaPietra, deSouza, Lai, & Mercer, 1992): A corpus of 360 millions words, $250,000$ different words.
  - Out of 6.75×10^{10} possible bigrams, only 14.5×10^6 actually occur. Of these, 8×10^6 only once.
  - 1.7×10^{16} trigrams are possible; 75×10^6 actually occur. Of these, 53×10^6 only once.
  - Using Good-Turing estimates, we can estimate that, in a new sample, the ML estimates 14.7% or the trigram and 2.2% of the trigrams will be 0.
- The *aggregate probability* of these unseen events can be significant.
- This gives an idea both on the size of the model, and of the data sparsity.

What do the *smoothing* and *interpolation methods* offer in the case of unknown words?

When estimating $P(w_2|w_1)$, when w_2 never happened following w_1, we basically estimate the bigram distribution based on the distribution of w_2; the more frequent it is, the higher the estimation of the bigram probability.

Class-based and **similarity-based** models provide an alternative to this independence assumption. In these models, the relationship between given words is modeled by analogy with other words that are in some sense similar to the given ones.

Brown et al. (1992): suggest a *class-based* n-gram model in which words with similar cooccurrence distributions are clustered into word classes.

Suppose that the vocabulary V is partitioned into a disjoint partition $C_1, C_2, \ldots C_k$

We say that a language model is an **n-gram class model** if it is an n-gram language model and if, in addition,

$$P(w_i|w_1^{i-1}) = P(w_i|c_i)P(c_i|c_1^{i-1}), \forall i = 1, \ldots n$$

The number of parameters of a model like that would thus be $O(C^n)$, which can be much fewer than a word based model.

The question is how to split the words into classes; it can be shown that the optimal thing to do is to split it in such a way the the *average mutual information* of adjacent classes is maximized, but there is not clear way of doing it.

Pereira, Tishby, and Lee (1993) propose a *soft distributional clustering* scheme for certain grammatical cooccurrences in which membership of a word in a class is probabilistic. Cooccurrence probabilities of words are then modeled by averaged cooccurrence probabilities of word clusters.

The key issue about these and the methods we will discuss is that the grouping is **distributional:**

We measure word similarity by the predictions words make about what words they cooccur with; this is different from general purpose (non-distributional) clustering methods, where (word) similarity is defined from intrinsic features independently of the cooccurrence.

In this rest of this lecture we will describe a general scheme (following Dagan, Lee, Pereira) for using word similarity, with the goal of improving the probability estimates of language models. In doing that we will also discuss several similarity measures in the context of both language modeling and a prediction task.

1.1 Distributional Similarity Models

We wish to model conditional probability distributions arising from the cooccurrence of linguistic objects.

We will consider here only bigram models. Specifically, consider pairs (w_1, w_2) with $w_1 \in V_1, w_2 \in V_2$.

- $P(w_2|w_1)$
  The conditional probability that a pair has second element w_2 given that its first element is w_1.

  Of course, we do not not the true language model and will therefore resort to different approximations of this quantity.

- $P(w)$
  denotes the unigram probability of the word w.

  (or rather, some empirical estimate drawn from a base language model, the true probability being unknown)

A similarity-based language model consists of three parts:

- A scheme for deciding which word pairs require a similarity-based estimate.
- A method for combining information from similar words.
- A a function measuring the similarity between words.

Notice that the models are typically constructed such that they consider only similarity between words in V_1, which are the conditioning events for the probabilities $P(w_2|w_1)$ that we want to estimate.

Keep this in mind. Although these models were originally proposed in the context of langauge models, the same ideas apply for feature abstractions. (We started to mention in passing the relations between elements we estimate probability for in language models and features for classifiers).

1.2 MLE; discounting and Redistribution

Recall that we discussed the *maximum likelihood estimate* of bigrams (that is of the pair (w_1, w_2) conditional on the appearance of w_1:

$$P_{ML}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})}.$$

where $C(\cdot)$ represents the observed frequency of the event.

Due the unreliability of this estimate, especially the presence of zero-count events, we adjust these estimate via *smoothing techniques*, and then *redistribute* the probability mass via *backoff* or *interpolation*.

Here we will make use of Katz's backoff, which we mentioned last time, mostly since it is considered more robust than existing interpolation techniques. (See: Goodman and Chen).

For the case of bigrams, we write the *modified backoff* as follows:

$$P_{bo}(w_2|w_1) = \begin{cases} \hat{P}(w_2|w_1), & \text{if } C(w_1w_2) > 0 \\ \alpha(w_1)\tilde{P}(w_2|w_1), & \text{if } C(w_1w_2) = 0 \end{cases}.$$

where $\hat{P}$ is the *smoothed* estimate (e.g., Good-Turing) and $\tilde{P}$ is the redistributed estimation, with $\alpha(w_1)$ being the normalization factor.

Comment: In the original back-off model, we backed off to $P(w_2)$ as the model for predicting $P(w_2|w_1)$ for unseen word pairs.

The key issue here is that we want to backoff to a more detailed model than unigrams. Therefore, we generalize Katz's method by writing $\tilde{P}(w_2|w_1)$ enabling us to use similarity-based estimates for unseen word pairs instead of unigram frequency.

That is, similarity based estimates will be used for unseen word pairs only.

Again, you can question that; when you think about these as features, you may want to generalize in other situations.

1.3 Combining Evidence

The goal is **to estimate $\tilde{P}(w_2|w_1)$ by taking into account information from words that are distributionally similar to w_1.**

The Similarity-based Assumption: If word w_1' is *similar* to word w_1, then w_1' can yield information about the probability of unseen word pairs involving w_1.

How to do it?

Most methods use a *weighted average* of the evidence provided by similar words, where the weight given to a particular word $w_1^{'}$ depends on its similarity to w_1.

Define:

$W(w_1, w_1')$**:** An increasing function of the similarity between w_1 and w_1' ,

$S(w_1)$**:** The set of words most similar to w_1.

Then the general form of the similarity model is a W-weighted linear combination of predictions of similar words:

$$P_{SIM}(w_2|w_1) = \sum_{w_1' \in S(w_1)} \frac{W(w_1, w_1')}{norm(w_1)} P(w_2|w_1'),$$

where

$$norm(w_1) = \sum_{w_1' \in S(w_1)} W(w_1, w_1')$$

is a normalization factor.

Interpretations: w_2 is more likely to occur with w_1 if it tends to occur with the words that are most similar to w_1.

What should $S(w_1)$ be?

We could have $S(w_1) = V_1$, or could threshold it to contain the k closest words to w_1, or those the are closer than some threshold. (See experiments in Dagan etc al.)

How to use it?

We can either have: $\tilde{P} = P_{SIM}$ in the backoff equation before, or use interpolation, e.g.:

$$\tilde{P}(w_2|w_1) = \lambda P(w_2) + (1 - \lambda) P_{SIM}(w_2|w_1)$$

which is a linear combination of the original backoff method and the similarity estimate. $\lambda = 0$ gives the similarity, $\lambda = 1$ is the backoff.

It is also possible to make the λ depend on w_1, as we suggested in the other interpolation algorithms, but this require more complex training.

1.4 Distributional Similarity Measures

We now have a general scheme, and all we need is to plug in a specific measure for the similarity of two words (tokens).

There are many possible functions, and not a clear or principled way to define them.

Notice that we are talking about *distances between probability distributions*. That is, we are not measuring the similarity between w and w', but rather the **distributional similarity**.

The distributions in play are:

$$P(w_2|w) \text{ and } P(w_2|w').$$

where w_2 varies and w_1, w' are fixed.

More generally, we can say that we measure the similarity of w and w' in some context c, which may not be the word before it, but the **subject** of it, if w, w' are verbs, or its **object**, or any other chosen context. So we are looking to define

$$Sim(w, w') \quad \text{or} \quad Dist(w, w')$$

as a function of two probability distributions

$$p = p(c|w) \quad \text{and} \quad q = q(c|w').$$

This immediately gives rise to several distance measures based on different ways for measuring **distances between distributions**.

A natural way to measure distance between two discrete distributions $p, q : X \to [0, 1]$ is the L_1 **distance**:

$$dist_{L1}(p, q) = \sum_{x \in X} |p(x) - q(x)|.$$

Other popular measures include the **KL divergence**:

$$dist_{KL}(p, q) \equiv D(p||q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}.$$

However,

$$D(p||q) = \infty \qquad \text{when } p(x) > 0 \textit{ and } q(c) = 0.$$

Thus, it cannot be used directly on maximum likelihood estimate (MLE) probabilities.

One possible solution is the **Jensen-Shannon measure** which first defines new probability distribution, the average

$$\frac{1}{2}(p+q)$$

and then considers the two distances (of p and q from this average, and averages it:

$$dist_{JS}(p,q) \equiv \frac{D(p||\frac{p+q}{2}) + D(q||\frac{p+q}{2})}{2}.$$

Another solution is the α -skew divergence measure, which uses the p distribution to smooth the q distribution. The value of the parameter α controls the extent to which the KL divergence is approximated.

$$dist_{\alpha}(p,q) \equiv D(\alpha p||(1-\alpha)q).$$

Using $\alpha = 0.99$ provides a close approximation to the KL divergence and has been shown to provide good results in previous research (Lee, 2001).

1.4.1 Mutual Information Based measures

Several distances are based on the notion of the **mutual information**.

Definition 1.1 *Given two random variables X, Y, the mutual information between them is the KL divergence between the joint probability distribution on X, Y and the* product *distribution defined on X, Y.*

$$I(X;Y) = D(p(x,y)||p(x)p(y)).$$

Phrased otherwise, it is the measure of how far the true joint is from independence.

We have defined mutual information between random variables. In many applications this is being abused and people use it to measure *point-wise mutual information.* Let x, y be two points in a probability space. The point-wise mutual information between these two points can be defined to be:

$$I(x,y) = log\frac{p(x,y)}{p(x)p(y)}.$$

This can be thought of as some statistical measure relating the events x and y.

For example, consider looking at pair of consecutive words w, w'. We can define:

- $p(w)$ - the frequency of the word w in a corpus,
- $p(w')$ - the frequency of the word w' in a corpus
- $p(w, w')$ - frequency of the pair w, w' in the corpus.

and then, consider $I(w, w')$ as *the amount of information* about the occurrence of w in location i given that we know that w' occurs at $i + 1$.

It turns out the this is not a good measure, and we will see examples for it.

As an example, consider two extreme cases. Assume the w, w' occurs only together. Then:

$$I(w,w') = log\frac{p(w,w')}{p(w)p(w')} = log\frac{p(w)}{p(w)p(w')} = log\frac{1}{p(w')}$$

So, the mutual information *increases* among perfectly dependent bigrams, when they become rarer.

On the other hand, if they are completely independent,

$$I(w,w') = log\frac{p(w,w')}{p(w)p(w')} = log1 = 0.$$

So, it can be thought of as a good measure for independence, but not a good measure for dependence, since it depends on the score of individual words.

Similarity measures based on this notions, therefore, make some corrections to account for this. Notice that

$$I(c,w) = log\frac{p(c,w)}{p(c)p(w)} = log\frac{p(c|w)p(w)}{p(c)p(w)} = log\frac{p(c|w)}{p(c)}.$$

A popular similarity measure based on mutual information is the one due to Dekang Lin:

$$dist_{Lin}(w,w') \equiv \frac{\sum_{F(w)\cap F(w')} I(c,w) + I(c,w')}{\sum_{F(w)} I(c,w) + \sum_{F(w')} I(c,w')},$$

where

$$F(w) \equiv \{c|I(c,w) > 0\}.$$

Look for definitions in (Dagan, Lee Pereira, 1999) and in (Julie Weeds, David Weir and Diana McCarthy, 2004)

This is also a source for confusion in this area since, sometimes, people did not realized that, and defined measures that are identical to existing ones.

Notice that for each similarity measure, a corresponding weight function $W(w_1; w_1')$ needs to be given. Again, this is relatively arbitrary; the only requirement is that it is increasing in the similarity between w_1 and w_1'.

1.5 Evaluation

How can we evaluate what we do?

1.5.1 Evaluation via a Language Model

Use the new bigram estimate vs. the original Katz backoff method for the bigram estimate, and see which one gives a better langauge model.

Consider N pairs (w_1, w_2).

Compute *test set perplexity:*

$$\sqrt[N]{\Pi_{i=1}^{N} P(w_i|w_{i-1})^{-1}}$$

This represents the average number of alternatives presented by the bigram model after each test word.

Thus, a better model will have a *lower* perplexity. The hope is that lower perplexity will indicate better prediction of unseen bigrams.

- Requires tuning of parameters. E.g. $k = 60$, the number of similar words used. The metric used here in the KL divergence.
- 40 million WSJ corpus. Evaluation on $18,000$ held-out sample; 10.6% of it is unseen.
- Improvement of 2.4% (from 237.4 to 231.7)

1.5.2 Evaluation via a Prediction Task

Input: A noun and two verbs.

Choose: Which verb was more likely to have the noun as a direct object.

```
make plans     -->  {make, take} plans

take actions   -->  {make, take} actions
```

This can be thought of as a variation of the Word Sense Disambiguation Problem ("bank" of a river or a "bank" as a financial institution).

The method is that of a *pseudo-word test*.

- A list of *pseudo-words* is constructed. Each element is a pair of words in V_2. Each word in V_2 contributes to exactly one pseudo-word.
- Each occurrence of w_2 in the test set is replaced with its pseudo word.
- Advantage: the "confusion set" (alternative senses) are under control; they can be chosen to have the same frequency, same pos etc.
- No manual annotation is required
- Not clear that this is a good reflection of what will the models do in a *real* prediction task.

Methodology:

- 44 million words;
- identify transitive main verbs (V_2) and head nouns (V_1).(POS tagger + rule based tools; some noise)
- Choose 1000 most frequent nouns.
- 80% of the pairs – $587,833$ are using for training the models. 20% left for testing. (In one test, word pairs that occur only once were deleted.) Only $17,152$ pairs that were not seen in the training were left in the test corpus. 5 fold cross validation. was done. 4 folds were used to tune the parameters of the models.
- Error measure:

$$\frac{1}{N}(\#\text{of incorrect choices} + (\#\text{of ties})/2)$$

where N is the number of test corpus.

Results:

- MLE performs at 0.5 (since all the test elements are unknown).
- Backoff is doing worse than 0.5 – average of 0.515.
- Similarity based methods are doing much better 0.27
- Notice the RAND measure, that shows that not only the similarity is important, but also the weight function, $W(w_1 w_1')$.