JUST-IN-TIME LANGUAGE MODELLING

Adam Berger and Robert Miller

School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15213
`aberger,rcm@cs.cmu.edu`

ABSTRACT

Traditional approaches to language modelling have relied on a fixed corpus of text to inform the parameters of a probability distribution over word sequences. Increasing the corpus size often leads to better-performing language models, but no matter how large, the corpus is a static entity, unable to reflect information about events which postdate it. In these pages we introduce an online paradigm which interleaves the estimation and application of a language model. We present a Bayesian approach to online language modelling, in which the marginal probabilities of a static trigram model are dynamically updated to match the topic being dictated to the system. We also describe the architecture of a prototype we have implemented which uses the World Wide Web (WWW) as a source of information, and provide results from some initial proof of concept experiments.

1. BACKGROUND

Of pressing concern to language modelling researchers is how to detect and account for a "non-stationary" source; that is, a source of words whose distribution changes over time. To take a concrete example, suppose we put an ASR system to the task of transcribing the evening news (to automatically generate a transcript for the hearing-impaired, say). The anchor may begin with a segment on Bosnia, in which words such as `strife`, `famine`, `Serbs`, `Albright` and `U.N.` are more probable than in general. Then the anchor moves to a story on the Iditarod dog sled race in Alaska, during which time the words `snow`, `mush`, `cold` and `canine` are more likely than in general. Adaptive language modelling addresses the task of ensuring that a model keeps up with a changing source distribution. In short: as the topic changes, so should the model.

One approach has been to partition the training corpus into a number of topics—a coarse division might be sports, politics and useless banter—and train individual models on each topic. When applying this composite model, one needs somehow to detect the topic at hand and select the model appropriate to the topic. A somewhat more refined approach is to allow a few topics to be active at once, and apply a weighted average of the individual topic models [1].

These approaches rely on a training corpus fixed "offline," prior to applying the model. Such an approach works well when the topic at hand is to be found in the training corpus, but not when the topic is absent from the corpus. This is not a problem that can be fixed with Moore's law and patience: nearly any source distribution which an ASR system is going to encounter will change as events occur in the world. No speech recognition system trained on data prior to 1993, for example, could possibly recognize that `Marlins` and `baseball` have a strong lexical correlation.

Adam Berger is partially supported by an IBM Cooperative Fellowship. Robert Miller is partially supported by an NDSEG Fellowship.

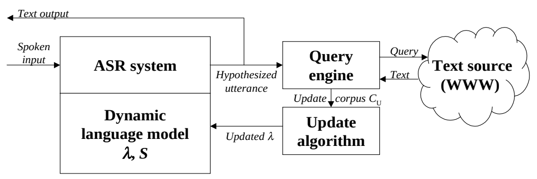

Figure 1: ASR system using just-in-time language modelling.

This document describes a language modelling system in which the estimation and application of the model are coupled; that is, a model which learns as it works. We envision the behavior of an ASR system incorporating this language model as follows (Figure 1). In processing a single utterance, the system uses the current language model to generate a hypothesis for the utterance. The hypothesis is then (while the system awaits the next utterance, say) sent to a query engine, which generates an update corpus based on using the hypothesis as a set of keywords in a search. The language model is reestimated to take into account the update corpus, and can be applied either to rescore the current utterance, or just to process the next utterance.

2. AN ONLINE MODEL OF LANGUAGE

Given is some stationary distribution (our *default model*) $\mathcal{S}$. We imagine $\mathcal{S}$ to be a trigram or similar conventional language model[1], whose parameters have been estimated offline on a large corpus of text $C_{\mathcal{S}}$. The system is occasionally provided with an "update corpus" of text $C_{\mathcal{U}}$ which (one hopes) has a high semantic correlation with the current topic of dictation, but which is likely to be much smaller—by several orders of magnitude—than $C_{\mathcal{S}}$.

How one generates such a corpus is taken up in the following section. Our concern here is to construct a language model λ which incorporates knowledge gleaned from $C_{\mathcal{U}}$ into $\mathcal{S}$. Some desired properties of λ:

[1]We make no explicit assumptions about the form of $\mathcal{S}$ in what follows, though the prototype described later uses the trigram model of [2]

(1) The influence which the update corpus $C_{\mathcal{U}}$ plays should increase with its size (number of words) $N_{\mathcal{U}}$. That is, as its size increases, we should consider $C_{\mathcal{U}}$ a more reliable source of information, and pay it greater heed.

(2) But how much greater heed? There should be a "knob" to adjust how much we adjust $\mathcal{S}$ as a function of the size of the update corpus.

We'll consider language models in the exponential family

$$\mathcal{F} \equiv \{\lambda : p(y|\mathbf{x},\lambda) = \frac{1}{Z_\lambda(x)} e^{\lambda_y} s(y|\mathbf{x})\} \tag{1}$$

where

- $s(y|\mathbf{x})$ is the probability (value) which the static model $\mathcal{S}$ assigns to the event that word y follows the sequence of words $\mathbf{x}$;
- $\lambda = \{\lambda_1, \lambda_2, \ldots\}$ is a set of real-valued parameters, one for each word. Roughly speaking, λ_y represents an adjustment to the marginal probability of word y in $\mathcal{S}$. In other words, word y is on average about e^{λ_y} more probable according to λ than according to $\mathcal{S}$.
- $Z_\lambda(\mathbf{x})$ is a normalization term, ensuring $\sum_y p(y|\mathbf{x},\lambda) = 1$.

One member of $\mathcal{F}$ is of particular interest, namely the model with $\lambda = \{0, 0, 0 \ldots\}$, which we'll write as λ^0. This is just another way of writing $\mathcal{S}$, the static distribution.

If the event $(\mathbf{x}, y)$ (the word y following the words $\mathbf{x}$) occurs $c_u(\mathbf{x}, y)$ times in a corpus of text which we denote by $C_{\mathcal{U}}$, then the probability which the model λ assigns to $C_{\mathcal{U}}$ (the "likelihood" of $C_{\mathcal{U}}$) is

$$p(C_{\mathcal{U}}|\lambda) \quad = \quad \prod_{\mathbf{x},y} \left[\frac{e^{\lambda_y} s(y|\mathbf{x})}{\sum_y e^{\lambda_y} s(y|\mathbf{x})}\right]^{c_u(\mathbf{x},y)} \tag{2}$$

We will also require a probability distribution over models $\lambda \in \mathcal{F}$, for which we adopt a Gaussian:

$$p(\lambda) = \frac{1}{\sqrt{2\pi\sigma^2}} \prod_y \exp\left(-\frac{{\lambda_y}^2}{2\sigma^2}\right) \tag{3}$$

A Gaussian prior of the form (3) imposes a "smoothness constraint," penalizing models by the amount they diverge from the default distribution $\mathcal{S}$. A justification for this approach comes from Occam's Razor, which, in this context, prefers the smoothest among competing models of the data $C_{\mathcal{U}}$.

One can think of σ^2, the real-valued parameter in (3), as a degree of trust in the update corpus. As σ^2 decreases, the *a priori* probability of a model with large parameters λ_y decreases. That is, the prior distribution of models becomes peaked around λ^0, the model with *no* adjustments in marginal probabilities. Thus σ^2 seems to satisfy the second of our desiderata.

The MAP-optimal model

Given a sampling distribution (2) and a prior distribution (3), we are now in a position to introduce the posterior distribution. For a particular update corpus $C_{\mathcal{U}}$, define $Q(\lambda)$, the *quality* (or posterior probability) of model λ, as

$$Q(\lambda) \equiv p(\lambda|C_{\mathcal{U}}) \tag{4}$$

$Q(\lambda)$ ranks candidate models by how likely they are, in light of the update corpus $C_{\mathcal{U}}$. The optimal model $\lambda^\star$—the one with the highest score—is written $\lambda^\star \equiv \text{argmax}_\lambda p(\lambda|C_{\mathcal{U}})$. Applying Bayes' law,

$$\lambda^\star = \text{argmax}_\lambda \frac{p(C_{\mathcal{U}}|\lambda)p(\lambda)}{p(C_{\mathcal{U}})} = \text{argmax}_\lambda p(C_{\mathcal{U}}|\lambda)p(\lambda) \tag{5}$$

We can drop off the denominator in the last equality since $C_{\mathcal{U}}$ is independent of λ.

Notice that in the case of a uniform prior $p(\lambda)$, the optimization problem posed by (5) reduces to *maximum-likelihood*: find that λ which assigns maximal probability to the data $C_{\mathcal{U}}$.

It has been recognized for some time in the computer vision community [3] that this type of Bayesian approach can be viewed as an instance of *regularization*, a popular method for solving ill-posed optimization problems. Given a set of data and a family of candidate models which account for the data, regularization prescribes that the optimal model $\lambda^\star$ maximize

$$F(\lambda) = D(\lambda) + \alpha T(\lambda)$$

where $D(\lambda)$ measures how well λ accounts for the data, $T(\lambda)$ measures the smoothness of the model λ, and α is a real-valued parameter which governs the tradeoff between these two objectives.

$F(\lambda)$ may reasonably be viewed as a fitness functional over conditional probability distributions. In this setting, it can be written as[2]:

$$\begin{aligned} F(\lambda) \quad &\equiv \quad \log p(C_{\mathcal{U}}|\lambda) + \log p(\lambda) \\ &= \quad \underbrace{\sum_{\mathbf{x},y} c_u(\mathbf{x},y) \log p(y|\mathbf{x},\lambda)}_{D(\lambda)} - \underbrace{\frac{1}{2\sigma^2}}_{\alpha} \underbrace{\sum_y {\lambda_y}^2}_{T(\lambda)} + \text{K} \end{aligned}$$

The constant $\alpha = \frac{1}{2\sigma^2}$ is a hyperparameter, trading off between well-fitted (σ^2 large) and smooth (σ^2 small) distributions. Substituting (2) and (3) into (5),

$$Q(\lambda) = \frac{1}{\sqrt{2\pi\sigma^2}} \prod_y \exp\left(-\frac{{\lambda_y}^2}{2\sigma^2}\right) \prod_{\mathbf{x},y} \left[\frac{e^{\lambda_y} s(y|\mathbf{x})}{\sum_y e^{\lambda_y} s(y|\mathbf{x})}\right]^{c_u(\mathbf{x},y)}$$

It is a straightforward exercise in calculus to derive the condition for $\lambda_y^\star$, the optimal value of λ_y (Space precludes a more thorough treatment here, but details are available in [4]):

$$\frac{\lambda_y^\star}{N_{\mathcal{U}}\sigma^2} + p_u(y) - \sum_{\mathbf{x}} p_u(\mathbf{x}) p(y|\mathbf{x},\lambda) = 0 \tag{6}$$

where $p_u(\mathbf{x}, y) \equiv c_u(\mathbf{x}, y)/N_{\mathcal{U}}$.

Some observations about (6):

- As $N_{\mathcal{U}}$ gets small, $\lambda_y^\star$ tends to zero. This is just what one would hope and expect: as the size of the update corpus shrinks, the optimal update to the marginal probability of y should vanish. The same occurs in the case of small σ^2, which is like saying that we don't trust the update corpus.
- Conversely, as $N_{\mathcal{U}}$ or σ^2 gets large, (6) becomes $p_u(y) = \sum_{\mathbf{x}} p_u(\mathbf{x}) p(y|\mathbf{x},\lambda)$. In other words, adjust $\lambda_y^\star$ so that the marginal probability of y in the model λ exactly matches its marginal probability in the update corpus.

[2]Here and elsewhere, K means "constant with respect to the variable(s) of interest," in this case λ_y.

Iterative scaling

We now discuss how to compute, given an update corpus $C_{\mathcal{U}}$ and a value for σ^2, the optimal model $\lambda^\star$. This approach derives from the *iterative scaling* algorithm, a method for finding the maximum-likelihood $\lambda^\star$ (i.e. in the case of a uniform prior). There are some technical issues to contend with when the optimal setting for λ_y is non-finite; these and other aspects of the iterative scaling are addressed in [5].

In seeking $\lambda^\star$, it helps to recast the problem to a slightly different form. Starting from some model λ, we seek the optimal change Δ_y to each parameter λ_y. We denote the vector of (additive) changes by Δ. Applying an auxiliary function argument common in such settings, we instead maximize a function $A(\Delta) \leq Q(\lambda + \Delta) - Q(\lambda)$, whose derivative is given by

$$\frac{\partial A(\Delta)}{\partial \Delta_y} = \frac{\lambda_y + \Delta_y}{N_{\mathcal{U}}\sigma^2} - p_u(y) + \sum_{\mathbf{x}} p_u(\mathbf{x}) p(y|\mathbf{x},\lambda) e^{\Delta_y} \qquad (7)$$

Finding the optimal Δ_y is a matter of setting this expression to zero for each y and solving for Δ_y. As a sanity check, notice that

- As $N_{\mathcal{U}}$ gets large, (7) turns into a linear version (linear since the constraints are non-overlapping) of the standard iterative scaling equation: the prior term drops out and $C_{\mathcal{U}}$ fully dictates the update Δ_y. The same thing happens as σ^2 gets large, meaning that we place no weight on the prior model $\mathcal{S}$.
- As $N_{\mathcal{U}}$ or σ^2 gets small, (7) turns into $\dfrac{\lambda_y + \Delta_y}{N_{\mathcal{U}}\sigma^2} = 0$; that is, $\Delta_y = -\lambda_y$. So as the update corpus shrinks, the optimal choice of Δ_y is the one which exactly cancels out any change we've made to the static model $\mathcal{S}$.

Efficient scoring

Members of the exponential family $\mathcal{F}$, as specified in (1), have a practical deficiency: calculating the probability which a particular $\lambda \in \mathcal{F}$ assigns to the event $(y, \mathbf{x})$ involves a sum, in the partition function $Z_\lambda(x)$, over all possible words y:

$$Z_\lambda(x) = \sum_y e^{\lambda_y} s(y|\mathbf{x}) \qquad (8)$$

A considerable speedup in evaluating (8) can be realized by the following heuristic. If the word y does not appear (or appears very rarely) in the update corpus $C_{\mathcal{U}}$, then fix $\lambda_y = 0$, on the grounds that we have insufficient evidence to alter the static marginal probability of y. This assumption is clearly false, since the absence of y is informative in itself. The assumption also violates (7), which dictates that, if $\lambda_y = 0$ and $p_u(y) = 0$, then

$$\frac{\Delta_y}{N_{\mathcal{U}}\sigma^2} + \sum_{\mathbf{x}} p_u(\mathbf{x}) p(y|\mathbf{x},\lambda) e^{\Delta_y} = 0$$

So $\Delta_y \neq 0$ in general, even if its empirical count is zero. In fact, the appropriate setting of Δ_y is negative in this case. However, by applying this heuristic, we can realize a savings as follows. Denoting by $\mathcal{Y}$ the set $\{y | y \in C_{\mathcal{U}}\}$ and by $\overline{\mathcal{Y}}$ the complement of this set, we can rewrite (8) as

$$\begin{aligned} Z_\lambda(x) &= \sum_{y \in \mathcal{Y}} e^{\lambda_y} s(y|\mathbf{x}) + \sum_{y \in \overline{\mathcal{Y}}} s(y|\mathbf{x}) \\ &= 1 + \sum_{y \in \mathcal{Y}} (e^{\lambda_y} - 1) s(y|\mathbf{x}) \end{aligned}$$

So we've reduced the sum over all words to a sum over just those words which appeared in the corpus $C_{\mathcal{U}}$, which can amount to a significant computational savings.

3. SYSTEM ARCHITECTURE

We have left much unsaid regarding the implementation of a dynamic language model. Here we describe some of the details of one particular implementation; the following section summarizes performance results of the system.

Our system operates in both 'sequential' and 'rescoring' modes. Sequential mode, designed for measuring the perplexity of text, assigns a probability (a score) to the input sentence by relying solely on the current language model. The input sentence is also used to generate an update corpus $C_{\mathcal{U}}$, from which the current language model is updated via the methods described in the previous section. The updated model is then ready to be applied to the next sentence. The sequential method benefits from and in fact relies upon some topical coherence from one sentence to the next.

'Rescoring' mode is designed for measuring word error rate of a speech (audio) signal. From an input utterance, an ASR system, equipped with a trigram language model, produces a hypothesized utterance, which is used to generate an update corpus $C_{\mathcal{U}}$. From $C_{\mathcal{U}}$ we can construct an updated language model, which is applied to generate a refined hypothesis of the input utterance. Unlike the 'sequential' approach, rescoring is an iterative procedure, though in practice we iterate only once in generating a final hypothesis.

Descending one level of abstraction, we now describe the behavior of the query engine, whose task is to generate an update corpus from a query. As depicted in figure 2, the query engine first filters noise words from the query, and then passes the query to a Web search engine. We experimented with two search engines: AltaVista[6], which indexes over 100 million Web pages of all kinds; and News Index[7], which indexes about 200 online news sources but revisits them frequently. Web pages found by the search engine are fetched in order of relevance in a multithreaded manner, stripped of HTML formatting tags, and added to the update corpus until it reaches the desired size. An update corpus of $10,000$ words typically contains about twenty Web pages and takes about three minutes to generate. These web pages comprise the update corpus $C_{\mathcal{U}}$, from which the techniques described in the previous section generate a new model λ', ready for use in decoding the next utterance.

4. EVALUATION

This section describes a set of experiments designed to gauge the ability of a dynamic language modelling system to reduce perplexity.

Table 1 summarizes the perplexity of three different newswire texts using a sequential approach. The table also demonstrates that

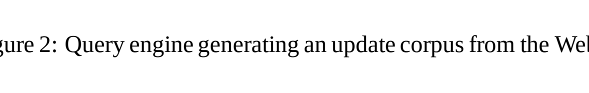

Figure 2: Query engine generating an update corpus from the Web.

	IRS	China	Marathon
Static trigram model	313	781	1060
Dynamic (using AltaVista)	315	781	1038
Dynamic (using News Index)	298	704	1016

Table 1: Perplexity of three AP newswire stories of 11/1/97—on IRS reform, US-Chinese relations, and the New York City Marathon—using a static and dynamic model.

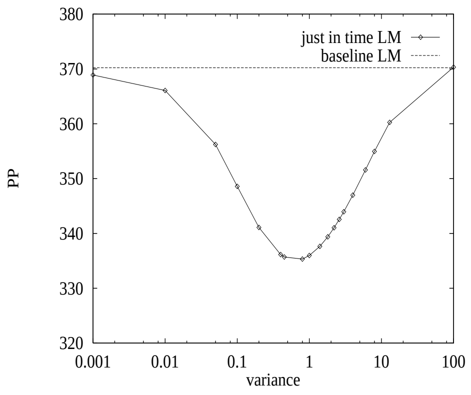

Figure 3: Observed dependence of perplexity of a segment of brodcast news data on the value of σ^2.

News Index—which, being a news-only index has a higher "signal to noise" ratio than AltaVista—is the better source, and thus we employ it exclusively in subsequent experiments.

Prior to generating any results, we applied binary search to discover the maximum-likelihood value of σ^2 for a heldout text corpus. Figure 3 depicts the observed dependence of σ^2 on the perplexity of an excerpt of broadcast news data taken from a the Hub4 evaluation set. Here we used the News Index source and the rescoring decoding approach. (It makes little sense to report perplexities in rescoring mode, but figure 3's interest lies only in the quality of the resulting system as a function of σ^2). Based on figure 3, the value of σ^2 was fixed at 0.8 for the experiments reported in table 1.

5. CONCLUSION

We have presented a Bayesian framework for dynamic language modelling, and discussed one particular implementation of a dynamic language modelling system. Though the described system is not today suitable for a real-time ASR system given the latency of the web, a nominally altered architecture—fetching an update corpus in the background after every few sentences, say, or even relying on a local information source (presumably of narrower extent than the WWW)—could make use of the approach described herein.

Considering that no effort was expended on trying to coax "relevant" queries from the search engine by judicious selection of keywords from the query, we consider the performance of our prototype encouraging. We are pursuing more refined approaches to generating an update corpus, and plan to incorporate the system into the CMU Sphinx [8] speech recognition system to determine what effect this approach can have on word error rate.

Acknowledgments

The authors thank John Lafferty and Stanley Chen for technical guidance.

6. REFERENCES

[1] K. Seymore and R. Rosenfeld (1997). Using story topics for language model adaptation. *Eurospeech '97*

[2] P. Clarkson and R. Rosenfeld (1997) Statistical language modeling using the CMU-Cambridge toolkit. *Eurospeech '97*

[3] R. Szeliski (1989) *Bayesian modelling of uncertainty in low level vision*. Boston: Kluwer Academic Publishers.

[4] A. Berger and R. Miller (1998). A real-time system for language modelling. CMU CS Department Technical Report. (Forthcoming).

[5] S. Della Pietra, V. Della Pietra and J. Lafferty (1997) Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*:19(4), 380–393.

[6] DEC, AltaVista. `http://altavista.digital.com`.

[7] News Index. `http://www.newsindex.com`.

[8] K.F. Lee, H.W. Hon and R. Reddy (1990) An overview of the SPHINX speech recognition system. *Journal of Acoustics, Speech, and Signal Processing*: 38:1.