Adaptation of Maximum Entropy Capitalizer: Little Data Can Help a Lot

Ciprian Chelba and **Alex Acero**
Microsoft Research
One Microsoft Way
Redmond, WA 98052
{chelba,alexac}@microsoft.com

Abstract

A novel technique for maximum "a posteriori" (MAP) adaptation of maximum entropy (MaxEnt) and maximum entropy Markov models (MEMM) is presented.

The technique is applied to the problem of recovering the correct capitalization of uniformly cased text: a "background" capitalizer trained on 20Mwds of Wall Street Journal (WSJ) text from 1987 is adapted to two Broadcast News (BN) test sets — one containing ABC Primetime Live text and the other NPR Morning News/CNN Morning Edition text — from 1996.

The "in-domain" performance of the WSJ capitalizer is 45% better than that of the 1-gram baseline, when evaluated on a test set drawn from WSJ 1994. When evaluating on the mismatched "out-of-domain" test data, the 1-gram baseline is outperformed by 60%; the improvement brought by the adaptation technique using a very small amount of matched BN data — 25-70kwds — is about 20-25% relative. Overall, automatic capitalization error rate of 1.4% is achieved on BN data.

1 Introduction

Automatic capitalization is a practically relevant problem: speech recognition output needs to be capitalized; also, modern word processors perform capitalization among other text proofing algorithms such as spelling correction and grammar checking. Capitalization can be also used as a preprocessing step in named entity extraction or machine translation. We study the impact of using increasing amounts of training data as well as using a small amount of adaptation data on this simple problem that is well suited to data-driven approaches since vast amounts of "training" data are easily obtainable by simply wiping the case information in text.

As in previous approaches, the problem is framed as an instance of the class of sequence labeling problems. A case frequently encountered in practice is that of using mismatched — out-of-domain, in this particular case we used Broadcast News — test data. For example, one may wish to use a capitalization engine developed on newswire text for e-mail or office documents. This typically affects negatively the performance of a given model, and more sophisticated models tend to be more brittle. In the capitalization case we have studied, the relative performance improvement of the MEMM capitalizer over the 1-gram baseline drops from in-domain — WSJ — performance of 45% to 35-40% when used on the slightly mismatched BN data.

In order to take advantage of the adaptation data in our scenario, a maximum a-posteriori (MAP) adaptation technique for maximum entropy (MaxEnt) models is developed. The adaptation procedure proves to be quite effective in further reducing the capitalization error of the WSJ MEMM capitalizer on BN test data. It is also quite general and could improve performance of MaxEnt models in any scenario where model adaptation is desirable. A further relative improvement of about 20% is obtained by adapting the WSJ model to Broadcast News (BN) text. Overall, the MEMM capitalizer adapted to BN data achieves 60% relative improvement in accuracy over the 1-gram baseline.

The paper is organized as follows: the next section frames automatic capitalization as a sequence labeling problem, presents previous approaches as well as the widespread and highly sub-optimal 1-gram capitalization technique that is used as a baseline in most experiments in this work and others. The MEMM sequence labeling technique is briefly reviewed in Section 3. Section 4 describes the MAP adaptation technique used for the capitalization of out-of-domain text. The detailed mathematical derivation is presented in Appendix A. The experimental results are presented in Section 5, followed by conclusions and suggestions for future work.

2 Capitalization as Sequence Tagging

Automatic capitalization can be seen as a sequence tagging problem: each lower-case word receives a

tag that describes its capitalization form. Similar to the work in (Lita et al., 2003), we tag each word in a sentence with one of the tags:

- `LOC` lowercase
- `CAP` capitalized
- `MXC` mixed case; no further guess is made as to the capitalization of such words. A possibility is to use the most frequent one encountered in the training data.
- `AUC` all upper case
- `PNC` punctuation; we decided to have a separate tag for punctuation since it is quite frequent and models well the syntactic context in a parsimonious way

For training a given capitalizer one needs to convert running text into uniform case text accompanied by the above capitalization tags. For example,

```
PrimeTime continues on ABC .PERIOD
Now ,COMMA from Los Angeles ,COMMA
Diane Sawyer .PERIOD
```

becomes

```
primetime_MXC continues_LOC on_LOC
abc_AUC .period_PNC
now_CAP ,comma_PNC from_LOC los_CAP
angeles_CAP ,comma_PNC diane_CAP
sawyer_CAP .period_PNC
```

The text is assumed to be already segmented into sentences. Any sequence labeling algorithm can then be trained for tagging lowercase word sequences with capitalization tags.

At test time, the uniform case text to be capitalized is first segmented into sentences[1] after which each sentence is tagged.

2.1 1-gram capitalizer

A widespread algorithm used for capitalization is the 1-gram tagger: for every word in a given vocabulary (usually large, 100kwds or more) use the most frequent tag encountered in a large amount of training data. As a special case for automatic capitalization, the most frequent tag for the first word in a sentence is overridden by `CAP`, thus capitalizing on the fact that the first word in a sentence is most likely capitalized[2].

[1]Unlike the training phase, the sentence segmenter at test time is assumed to operate on uniform case text.

[2]As with everything in natural language, it is not hard to find exceptions to this "rule".

Due to its popularity, both our work and that of (Lita et al., 2003) uses the 1-gram capitalizer as a baseline. The work in (Kim and Woodland, 2004) indicates that the same 1-gram algorithm is used in Microsoft Word 2000 and is consequently used as a baseline for evaluating the performance of their algorithm as well.

2.2 Previous Work

We share the approach to capitalization as sequence tagging with that of (Lita et al., 2003). In their approach, a language model is built on pairs (word, tag) and then used to disambiguate over all possible tag assignments to a sentence using dynamic programming techniques.

The same idea is explored in (Kim and Woodland, 2004) in the larger context of automatic punctuation generation and capitalization from speech recognition output. A second approach they consider for capitalization is the use a rule-based tagger as described by (Brill, 1994), which they show to outperform the case sensitive language modeling approach and be quite robust to speech recognition errors and punctuation generation errors.

Departing from their work, our approach builds on a standard technique for sequence tagging, namely MEMMs, which has been successfully applied to part-of-speech tagging (Ratnaparkhi, 1996). The MEMM approach models the tag sequence T conditionally on the word sequence W, which has a few substantial advantages over the 1-gram tagging approach:

- discriminative training of probability model $P(T|W)$ using conditional maximum likelihood is well correlated with tagging accuracy
- ability to use a rich set of word-level features in a parsimonious way: sub-word features such as prefixes and suffixes, as well as future words[3] are easily incorporated in the probability model
- no concept of "out-of-vocabulary" word: sub-word features are very useful in dealing with words not seen in the training data
- ability to integrate rich contextual features into the model

More recently, certain drawbacks of MEMM models have been addressed by the conditional random field (CRF) approach (Lafferty et al., 2001) which slightly outperforms MEMMs on a standard part-of-speech tagging task. In a similar vein, the work

[3]Relative to the current word, whose tag is assigned a probability value by the MEMM.

of (Collins, 2002) explores the use of discriminatively trained HMMs for sequence labeling problems, a fair baseline for such cases that is often overlooked in favor of the inadequate maximum likelihood HMMs.

The work on adapting the MEMM model parameters using MAP smoothing builds on the Gaussian prior model used for smoothing MaxEnt models, as presented in (Chen and Rosenfeld, 2000). We are not aware of any previous work on MAP adaptation of MaxEnt models using a prior, be it Gaussian or a different one, such as the exponential prior of (Goodman, 2004). Although we do not have a formal derivation, the adaptation technique should easily extend to the CRF scenario.

A final remark contrasts rule-based approaches to sequence tagging such as (Brill, 1994) with the probabilistic approach taken in (Ratnaparkhi, 1996): having a weight on each feature in the MaxEnt model and a sound probabilistic model allows for a principled way of adapting the model to a new domain; performing such adaptation in a rule-based model is unclear, if at all possible.

3 MEMM for Sequence Labeling

A simple approach to sequence labeling is the maximum entropy Markov model. The model assigns a probability $P(T|W)$ to any possible tag sequence $T = t_1 \dots t_n = T_1^n$ for a given word sequence $W = w_1 \dots w_n$. The probability assignment is done according to:

$$P(T|W) = \prod_{i=1}^{n} P(t_i|\underline{x}_i(W, T_1^{i-1}))$$

where t_i is the tag corresponding to word i and $\underline{x}_i(W, T_1^{i-1})$ is the conditioning information at position i in the word sequence on which the probability model is built.

The approach we took is the one in (Ratnaparkhi, 1996), which uses $\underline{x}_i(W, T_1^{i-1}) = \{w_i, w_{i-1}, w_{i+1}, t_{i-1}, t_{i-2}\}$. We note that the probability model is causal in the sequencing of tags (the probability assignment for t_i only depends on previous tags t_{i-1}, t_{i-2}) which allows for efficient algorithms that search for the most likely tag sequence $T^*(W) = \arg\max_T P(T|W)$ as well as ensures a properly normalized conditional probability model $P(T|W)$.

The probability $P(t_i|\underline{x}_i(W, T_1^{i-1}))$ is modeled using a maximum entropy model. The next section briefly describes the training procedure; for details the reader is referred to (Berger et al., 1996).

3.1 Maximum Entropy State Transition Model

The sufficient statistics that are extracted from the training data are tuples $(y, \#, \underline{x}) = (t_i, \#, \underline{x}_i(W, T_1^{i-1}))$ where t_i is the tag assigned in context $\underline{x}_i(W, T_1^{i-1}) = \{w_i, w_{i-1}, w_{i+1}, t_{i-1}, t_{i-2}\}$ and $\#$ denotes the count with which this event has been observed in the training data. By way of example, the event associated with the first word in the example in Section 2 is (`*bdw*` denotes a special boundary type):

```
MXC 1
currentword=primetime
previousword=*bdw*
nextword=continues
t1=*bdw* t1,2=*bdw*,*bdw*
prefix1=p prefix2=pr prefix3=pri
suffix1=e suffix2=me suffix3=ime
```

The maximum entropy probability model $P(y|\underline{x})$ uses features which are indicator functions of the type:

$$f(y, \underline{x}) = \begin{cases} 1, & if\ y = MXC\ and\ \underline{x}.w_i = primetime \\ 0, & o/w \end{cases}$$

Assuming a set of features $\mathcal{F}$ whose cardinality is F, the probability assignment is made according to:

$$p_\Lambda(y|\underline{x}) = Z^{-1}(\underline{x}, \Lambda) \cdot exp\left[\sum_{i=1}^{F} \lambda_i f_i(\underline{x}, y)\right]$$

$$Z(\underline{x}, \Lambda) = \sum_{y} exp\left[\sum_{i=1}^{F} \lambda_i f_i(\underline{x}, y)\right]$$

where $\Lambda = \{\lambda_1 \dots \lambda_F\} \in \mathcal{R}^F$ is the set of real-valued model parameters.

3.1.1 Feature Selection

We used a simple *count cut-off* feature selection algorithm which counts the number of occurrences of all features in a predefined set after which it discards the features whose count is less than a pre-specified threshold. The parameter of the feature selection algorithm is the threshold value; a value of 0 will keep all features encountered in the training data.

3.1.2 Parameter Estimation

The model parameters Λ are estimated such that the model assigns maximum log-likelihood to the training data subject to a Gaussian prior centered at $\underline{0}$, $\Lambda \sim \mathcal{N}(\underline{0}, diag(\sigma_i^2))$, that ensures smoothing (Chen and Rosenfeld, 2000):

$$L(\Lambda) = \sum_{\underline{x}, y} \tilde{p}(\underline{x}, y) \log p_\Lambda(y|\underline{x}) \qquad (1)$$

$$-\sum_{i=1}^{F}\frac{\lambda_i^2}{2\sigma_i^2}+const(\Lambda)$$

As shown in (Chen and Rosenfeld, 2000) — and rederived in Appendix A for the non-zero mean case — the update equations are:
$\lambda_i^{(t+1)}=\lambda_i^{(t)}+\delta_i$, where δ_i satisfies:

$$\sum_{\underline{x},y}\tilde{p}(\underline{x},y)f_i(\underline{x},y)-\frac{\lambda_i}{\sigma_i^2}=\frac{\delta_i}{\sigma_i^2}+\sum_{\underline{x},y}\tilde{p}(\underline{x})p_\Lambda(y|\underline{x})f_i(\underline{x},y)exp(\delta_i f^{\#}(\underline{x},y)) \quad (2)$$

In our experiments the variances are tied to $\sigma_i=\sigma$ whose value is determined by line search on development data such that it yields the best tagging accuracy.

4 MAP Adaptation of Maximum Entropy Models

In the adaptation scenario we already have a MaxEnt model trained on the background data and we wish to make best use of the adaptation data by balancing the two. A simple way to accomplish this is to use MAP adaptation using a prior distribution on the model parameters.

A Gaussian prior for the model parameters Λ has been previously used in (Chen and Rosenfeld, 2000) for smoothing MaxEnt models. The prior has 0 mean and diagonal covariance: $\Lambda\sim\mathcal{N}(\underline{0},diag(\sigma_i^2))$. In the adaptation scenario, the prior distribution used is centered at the parameter values Λ^0 estimated from the background data instead of $\underline{0}$: $\Lambda\sim\mathcal{N}(\Lambda^0,diag(\sigma_i^2))$.

The regularized log-likelihood of the adaptation training data becomes:

$$L(\Lambda) = \sum_{\underline{x},y}\tilde{p}(\underline{x},y)\log p_\Lambda(y|\underline{x})-\sum_{i=1}^{F}\frac{(\lambda_i-\lambda_i^0)^2}{2\sigma_i^2}+const(\Lambda) \quad (3)$$

The adaptation is performed in stages:

- apply feature selection algorithm on adaptation data and determine set of features $\mathcal{F}_{adapt}$.
- build new model by taking the union of the background and the adaptation feature sets: $\mathcal{F}=\mathcal{F}_{background}\cup\mathcal{F}_{adapt}$; each of the background features receives the corresponding weight λ_i determined on the background training data; the new features $\mathcal{F}_{adapt}\setminus\mathcal{F}_{background}$[4] introduced in the model receive 0 weight. The resulting model is thus equivalent with the background model.
- train the model such that the regularized log-likelihood of the adaptation training data is maximized. The prior mean is set at $\Lambda^0=\Lambda_{background}\cdot\underline{0}$; $\cdot$ denotes concatenation between the parameter vector for the background model and a 0-valued vector of length $|\mathcal{F}_{adapt}\setminus\mathcal{F}_{background}|$ corresponding to the weights for the new features.

As shown in Appendix A, the update equations are very similar to the 0-mean case:

$$\sum_{\underline{x},y}\tilde{p}(\underline{x},y)f_i(\underline{x},y)-\frac{(\lambda_i-\lambda_i^0)}{\sigma_i^2}=\frac{\delta_i}{\sigma_i^2}+\sum_{\underline{x},y}\tilde{p}(\underline{x})p_\Lambda(y|\underline{x})f_i(\underline{x},y)exp(\delta_i f^{\#}(\underline{x},y)) \quad (4)$$

The effect of the prior is to keep the model parameters λ_i close to the background ones. The cost of moving away from the mean for each feature f_i is specified by the magnitude of the variance σ_i: a small variance σ_i will keep the weight λ_i close to its mean; a large variance σ_i will make the regularized log-likelihood (see Eq. 3) insensitive to the prior on λ_i, allowing the use of the best value λ_i for modeling the adaptation data.

Another observation is that not only the features observed in the adaptation data get updated: even if $E_{\tilde{p}(\underline{x},y)}[f_i]=0$, the weight λ_i for feature f_i will still get updated if the feature f_i triggers for a context $\underline{x}$ encountered in the adaptation data and *some* predicted value y — not necessarily present in the adaptation data in context $\underline{x}$.

In our experiments the variances were tied to $\sigma_i=\sigma$ whose value was determined by line search on development data drawn from the adaptation data. The common variance σ will thus balance optimally the log-likelihood of the adaptation data with the Λ^0 mean values obtained from the background data.

Other tying schemes are possible: separate values could be used for the $\mathcal{F}_{adapt}\setminus\mathcal{F}_{background}$ and $\mathcal{F}_{background}$ feature sets, respectively. We did not experiment with various tying schemes although this is a promising research direction.

4.1 Relationship with Minimum Divergence Training

Another possibility to adapt the background model is to do minimum KL divergence (MinDiv) train-

[4] We use $A\setminus B$ to denote set difference.

ing (Pietra et al., 1995) between the background exponential model $\mathcal{B}$ — assumed fixed — and an exponential model $\mathcal{A}$ built using the $\mathcal{F}_{background} \cup \mathcal{F}_{adapt}$ feature set. It can be shown that, if we smooth the $\mathcal{A}$ model with a Gaussian prior on the feature weights that is centered at $\underline{0}$ — following the approach in (Chen and Rosenfeld, 2000) for smoothing maximum entropy models — then the MinDiv update equations for estimating $\mathcal{A}$ on the adaptation data are identical to the MAP adaptation procedure we proposed[5].

However, we wish to point out that the equivalence holds *only if the feature set for the new model* $\mathcal{A}$ *is* $\mathcal{F}_{background} \cup \mathcal{F}_{adapt}$. The straightforward application of MinDiv training — by using only the $\mathcal{F}_{adapt}$ feature set for $\mathcal{A}$ — will not result in an equivalent procedure to ours. In fact, the difference in performance between this latter approach and ours could be quite large since the cardinality of $\mathcal{F}_{background}$ is typically several orders of magnitude larger than that of $\mathcal{F}_{adapt}$ and our approach also updates the weights corresponding to features in $\mathcal{F}_{background} \setminus \mathcal{F}_{adapt}$. Further experiments are needed to compare the performance of the two approaches.

5 Experiments

The baseline 1-gram and the background MEMM capitalizer were trained on various amounts of WSJ (Paul and Baker, 1992) data from 1987 — files `WS87_{001-126}`. The in-domain test data used was file `WS94_000` (8.7kwds).

As for the adaptation experiments, two different sets of BN data were used, whose sizes are summarized in Table 1:

1. BN CNN/NPR data. The training/development/test partition consisted of a 3-way random split of file `BN624BTS`. The resulting sets are denoted CNN-trn/dev/tst, respectively
2. BN ABC Primetime data. The training set consisted of file `BN623ATS` whereas the development/test set consisted of a 2-way random split of file `BN624ATS`

5.1 In-Domain Experiments

We have proceeded building both 1-gram and MEMM capitalizers using various amounts of background training data. The model sizes for the 1-gram and MEMM capitalizer are presented in Table 2. Count cut-off feature selection has been used for the MEMM capitalizer with the threshold set at 5, so the MEMM model size is a function of the training data. The 1-gram capitalizer used a vocabulary of the most likely 100k wds derived from the training data.

[5]Thanks to one of the anonymous reviewers for pointing out this possible connection.

Data set	Partition		
	train	devel	test
WSJ	2-20M	—	8.7k
CNN	73k	73k	73k
ABC	25k	8k	8k

Table 1: Background and adaptation training, development, and test data partition sizes

Model	No. Param. (10^3)		
Training Data Size (10^6)	2.0	3.5	20.0
1-gram	100	100	100
MEMM	76	102	238

Table 2: Background models size (number of parameters) for various amounts of training data

We first evaluated the in-domain and out-of-domain relative performance of the 1-gram and the MEMM capitalizers as a function of the amount of training data. The results are presented in Table 3. The MEMM capitalizer performs about 45% better than the 1-gram one when trained and evaluated on Wall Street Journal text. The relative performance improvement of the MEMM capitalizer over the 1-gram baseline drops to 35-40% when using out-of-domain Broadcast News data. Both models benefit from using more training data.

Model	Test Data	Cap ERR (%)		
Training Data Size (10^6)		2.0	3.5	20.0
1-gram	WSJ-tst	5.4	5.2	4.4
MEMM	WSJ-tst	2.9	2.5	2.3
1-gram	ABC-dev	3.1	2.9	2.6
MEMM	ABC-dev	2.2	2.0	1.6
1-gram	CNN-dev	4.4	4.2	3.5
MEMM	CNN-dev	2.7	2.5	2.1

Table 3: Background models performance on in-domain (WSJ-test) and out-of-domain (BN-dev) data for various amounts of training data

5.2 Adaptation Experiments

We have then adapted the best MEMM model built on 20Mwds on the two BN data sets (CNN/ABC) and compared performance against the 1-gram and the unadapted MEMM models.

There are a number of parameters to be tuned on development data. Table 4 presents the varia-

tion in model size with different count cut-off values for the feature selection procedure on the adaptation data. As can be seen, very few features are added to the background model. Table 5 presents the variation in log-likelihood and capitalization accuracy on the CNN adaptation training and development data, respectively. The adaptation procedure was found

Cut-off	0	5	10^6
No. features	243,262	237,745	237,586

Table 4: Adapted model size as a function of count cut-off threshold used for feature selection on CNN-trn adaptation data; *the entry corresponding to the cut-off threshold of* 10^6 *represents the number of features in the background model*

to be insensitive to the number of reestimation iterations, and, more surprisingly, to the number of features added to the background model from the adaptation data, as shown in 5. The most sensitive parameter is the prior variance σ^2, as shown in Figure 1; its value is chosen to maximize classification accuracy on development data. As expected, low values of σ^2 result in no adaptation at all, whereas high values of σ^2 fit the training data very well, and result in a dramatic increase of training data log-likelihood and accuracies approaching 100%.

Cut-off	σ^2	LogL (nats)	Cap ACC (%) CNN-trn	CNN-dev
0	0.01	-4258.58	98.00	97.98
0	3.0	-1194.45	99.63	98.62
5	0.01	-4269.72	98.00	97.98
5	3.0	-1369.26	99.55	98.60
10^6	0.01	-4424.58	98.00	97.96
10^6	3.0	-1467.46	99.52	98.57

Table 5: Adapted model performance for various count cut-off and σ^2 variance values; log-likelihood and accuracy on adaptation data CNN-trn as well as accuracy on held-out data CNN-dev; *the background model results (no new features added) are the entries corresponding to the cut-off threshold of* 10^6

Finally, Table 6 presents the results on test data for 1-gram, background and adapted MEMM. As can be seen, the background MEMM outperforms the 1-gram model on both BN test sets by about 35-40% relative. Adaptation improves performance even further by another 20-25% relative. Overall, the adapted models achieve 60% relative reduction in capitalization error over the 1-gram baseline on both BN test sets. An intuitively satisfying result is the fact that the cross-test set performance (CNN

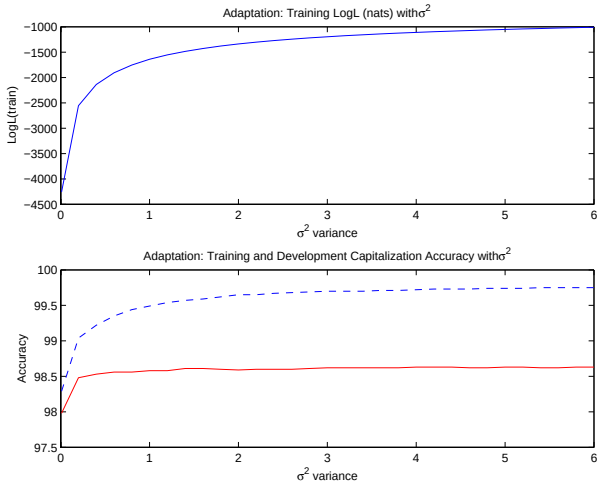

Figure 1: Variation of training data log-likelihood, and training/development data (- -/– line) capitalization accuracy as a function of the prior variance σ^2

Model	Adapt Data	Cap ERR (%) ABC-tst	CNN-tst
1-gram	—	2.7	3.7
MEMM	—	**1.8**	**2.2**
MEMM	ABC-trn	1.4	1.7
MEMM	CNN-trn	2.4	1.4

Table 6: Background and adapted models performance on BN test data; two adaptation/test sets are used: ABC and CNN

adapted model evaluated on ABC data and the other way around) is worse than the adapted one.

6 Conclusions and Future Work

The MEMM tagger is very effective in reducing both in-domain and out-of-domain capitalization error by 35%-45% relative over a 1-gram capitalization model.

We have also presented a general technique for adapting MaxEnt probability models. It was shown to be very effective in adapting a background MEMM capitalization model, improving the accuracy by 20-25% relative. An overall 50-60% reduction in capitalization error over the standard 1-gram baseline is achieved. A surprising result is that the adaptation performance gain is not due to adding more, domain-specific features but rather making better use of the background features for modeling the in-domain data.

As expected, adding more background training data improves performance but a very small amount of domain specific data also helps significantly if one can make use of it in an effective way. The

"There's no data like more data" rule-of-thumb could be amended by *"..., especially if it's the right data!"*.

As future work we plan to investigate the best way to blend increasing amounts of less-specific background training data with specific, in-domain data for this and other problems.

Another interesting research direction is to explore the usefulness of the MAP adaptation of MaxEnt models for other problems among which we wish to include language modeling, part-of-speech tagging, parsing, machine translation, information extraction, text routing.

Acknowledgments

Special thanks to Adwait Ratnaparkhi for making available the code for his MEMM tagger and MaxEnt trainer.

References

A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–72, March.

Eric Brill. 1994. Some Advances in Transformation-Based Part of Speech Tagging. In *National Conference on Artificial Intelligence*, pages 722–727.

Stanley F. Chen and Ronald Rosenfeld. 2000. A Survey of Smoothing Techniques for Maximum Entropy Models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50.

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1–8, University of Pennsylvania, Philadelphia, PA, July. ACL.

Joshua Goodman. 2004. Exponential Priors for Maximum Entropy Models. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 305–312, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Ji-Hwan Kim and Philip C. Woodland. 2004. Automatic Capitalization Generation for Speech Input. *Computer Speech and Language*, 18(1):67–90, January.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA.

L. Lita, A. Ittycheriah, S. Roukos, and N. Kambhatla. 2003. tRuEcasIng. In *Proccedings of ACL*, pages 152–159, Sapporo, Japan.

Doug B. Paul and Janet M. Baker. 1992. The design for the Wall Street Journal-based CSR corpus. In *Proceedings of the DARPA SLS Workshop*. February.

S. Della Pietra, V. Della Pietra, and J. Lafferty. 1995. Inducing features of random fields. Technical Report CMU-CS-95-144, School of Computer Science, Carnegie Mellon University, Pittsburg, PA.

Adwait Ratnaparkhi. 1996. A Maximum Entropy Model for Part-of-Speech Tagging. In Eric Brill and Kenneth Church, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142. Association for Computational Linguistics, Somerset, New Jersey.

Appendix

A Modified IIS for MaxEnt MAP Adaptation Using a Gaussian Prior

The regularized log-likelihood of the training data — to be maximized by the MAP adaptation training algorithm — is:

$$
\begin{aligned}
L(\Lambda) &= \sum_{\underline{x},y} \tilde{p}(\underline{x},y) \log p_\Lambda(y|\underline{x}) \\
&\quad - \sum_{i=1}^{F} \frac{(\lambda_i - \lambda_i^0)^2}{2\sigma_i^2} + const(\Lambda) \\
&= \sum_{\underline{x},y} \tilde{p}(\underline{x},y) \sum_{i=1}^{F} \lambda_i f_i(\underline{x},y) - \\
&\quad \sum_{\underline{x},y} \tilde{p}(\underline{x},y) \log \sum_{y'} \exp\left[\sum_{i=1}^{F} \lambda_i f_i(\underline{x},y')\right] \\
&\quad - \sum_{i=1}^{F} \frac{(\lambda_i - \lambda_i^0)^2}{2\sigma_i^2} + const(\Lambda) \\
&= \sum_{\underline{x},y} \tilde{p}(\underline{x},y) \sum_{i=1}^{F} \lambda_i f_i(\underline{x},y) - \\
&\quad \sum_{\underline{x}} \tilde{p}(\underline{x}) \log \sum_{y'} \exp\left[\sum_{i=1}^{F} \lambda_i f_i(\underline{x},y')\right] \\
&\quad - \sum_{i=1}^{F} \frac{(\lambda_i - \lambda_i^0)^2}{2\sigma_i^2} + const(\Lambda)
\end{aligned}
$$

where the last equality holds because the argument of the $\log$ is independent of y.

The derivation of the updates follows very closely the one in (Chen and Rosenfeld, 2000) for smoothing a MaxEnt model by using a Gaussian prior with $\underline{0}$ mean and diagonal covariance matrix. At each iteration we seek to find a set of updates for Λ, $\Delta = \{\delta_i\}$, that increase the regularized log-likelihood $L(\Lambda)$ by the largest amount possible.

After a few basic algebraic manipulations, the difference in log-likelihood caused by a Δ change in the Λ values becomes:

$$\begin{aligned} & L(\Lambda+\Delta) - L(\Lambda) \\ = & \sum_{\underline{x},y} \tilde{p}(\underline{x},y) \sum_{i=1}^{F} \delta_i f_i(\underline{x},y) - \\ & \sum_{\underline{x}} \tilde{p}(\underline{x}) \log \sum_{y} p_\Lambda(y|\underline{x}) \exp\left[\sum_{i=1}^{F} \delta_i f_i(\underline{x},y)\right] \\ & - \sum_{i=1}^{F} \frac{2(\lambda_i - \lambda_i^0)\delta_i + {\delta_i}^2}{2\sigma_i^2} \end{aligned}$$

Following the same lower bounding technique as in (Chen and Rosenfeld, 2000) by using $\log\ x \leq x-1$ and Jensen's inequality for the U-convexity of the exponential we obtain:

$$\begin{aligned} & L(\Lambda+\Delta) - L(\Lambda) \\ \geq & \sum_{\underline{x},y} \tilde{p}(\underline{x},y) \sum_{i=1}^{F} \delta_i f_i(\underline{x},y) + 1 - \\ & \sum_{\underline{x},y} \tilde{p}(\underline{x}) p_\Lambda(y|\underline{x}) \sum_{i=1}^{F} \frac{f_i(\underline{x},y)}{f^{\#}(\underline{x},y)} exp(\delta_i f^{\#}(\underline{x},y)) \\ & - \sum_{i=1}^{F} \frac{2(\lambda_i - \lambda_i^0)\delta_i + {\delta_i}^2}{2\sigma_i^2} = A(\Delta,\Lambda) \end{aligned}$$

where $f^{\#}(\underline{x},y) = \sum_{i=1}^{F} f_i(\underline{x},y)$. Taking the first and second partial derivative of $A(\Delta,\Lambda)$ with respect to δ_i we obtain, respectively:

$$\begin{aligned} \frac{\partial A(\Delta,\Lambda)}{\partial \delta_i} = & E_{\tilde{p}(\underline{x},y)}[f_i] - \frac{(\lambda_i - \lambda_i^0)}{\sigma_i^2} + \\ & \frac{\delta_i}{\sigma_i^2} - E_{\tilde{p}(\underline{x})p_\Lambda(y|\underline{x})}\left[f_i \cdot exp(\delta_i f^{\#})\right] \end{aligned}$$

and

$$\begin{aligned} \frac{\partial^2 A(\Delta,\Lambda)}{\partial {\delta_i}^2} = & -\frac{1}{\sigma_i^2} \\ & -E_{\tilde{p}(\underline{x})p_\Lambda(y|\underline{x})}\left[f_i \cdot f^{\#} \cdot exp(\delta_i f^{\#})\right] < 0 \end{aligned}$$

Since $A(\underline{0},\Lambda) = 0$ and $\frac{\partial^2 A(\Delta,\Lambda)}{\partial {\delta_i}^2} < 0$, by solving for the unique root Δ^* of $\frac{\partial A(\Delta,\Lambda)}{\partial \delta_i} = 0$ we obtain the maximum value of $A(\Delta,\Lambda)$ – which is non-negative and thus guarantees that the regularized log-likelihood does not decrease at each iteration: $L(\Lambda+\Delta^*) - L(\Lambda) \geq 0$.

Solving for the root of $\frac{\partial A(\Delta,\Lambda)}{\partial \delta_i} = 0$ results in the update Eq. 4 and is equivalent to finding the solution to:

$$\begin{aligned} & E_{\tilde{p}(\underline{x},y)}[f_i] - \frac{(\lambda_i - \lambda_i^0)}{\sigma_i^2} = \frac{\delta_i}{\sigma_i^2} + \\ & \sum_{\underline{x},y} \tilde{p}(\underline{x}) p_\Lambda(y|\underline{x}) f_i(\underline{x},y) exp(\delta_i f^{\#}(\underline{x},y)) \end{aligned}$$

A convenient way to solve this equation is to substitute $\alpha_i = exp(\delta_i)$ and $a_i = E_{\tilde{p}(\underline{x},y)}[f_i] - \frac{(\lambda_i - \lambda_i^0)}{\sigma_i^2}$ and then use Newton's method for finding the solution to $a_i = f(\alpha_i)$, where $f(\alpha)$ is:

$$f(\alpha) = \frac{\log \alpha}{\sigma_i^2} + \sum_{\underline{x},y} \tilde{p}(\underline{x}) p_\Lambda(y|\underline{x}) f_i(\underline{x},y) \alpha^{f^{\#}(\underline{x},y)}$$

The summation on the right hand side reduces to accumulating the coefficients of a polynomial in α whose maximum degree is the highest possible value of $f^{\#}(\underline{x},y)$ on any context $\underline{x}$ encountered in the training data and any allowed predicted value $y \in \mathcal{Y}$.