Efficient Representation and Fast Look-up of Maximum Entropy Language Models

Jia Cui, Stanley Chen and Bowen Zhou

IBM T.J. Watson Research Center
Yorktown Heights, New York 10598, USA
jiacui,stanchen,zhou@us.ibm.com

***Abstract*—Word class information has long been proven useful in language modeling (LM). However, the improved performance of class-based LMs over word n-gram models generally comes at the cost of increased decoding complexity and model size. In this paper, we propose a modified version of the Maximum Entropy token-based language model of [1] that matches the performance of the best existing class-based models, but which is as fast for decoding as a word n-gram model. In addition, while it is easy to statically combine word n-gram models built on different corpora into a single word n-gram model for fast decoding, it is unknown how to statically combine class-based LMs effectively. Another contribution of this paper is to propose a novel combination method that retains the gain of class-based LMs over word n-gram models. Experimental results on several spoken language translation tasks show that our model performs significantly better than word n-gram models with comparable decoding speed and only a modest increase in model size.**

I. Introduction

Language modeling is a crucial component in automatic speech recognition and statistical machine translation. While word n-gram models are the dominant technology in practice, class-based models offer the potential for improved performance [2]. For example, Model M, a class-based Maximum Entropy (ME) model, has achieved some of the largest reported gains in speech recognition performance as compared to a word n-gram model [3]. However, this performance comes at some expense: models are typically twice as large or more as compared to a word n-gram model trained on the same data, and decoding speed is many times slower. In many real-life systems, these additional resource requirements are unacceptable.[1]

In this paper, we propose a modified version of the Maximum Entropy token-based language model (MET-LM) [1] that is about as fast as a word n-gram model for decoding, and which can be almost as compact. Unlike Model M and many other class-based models which separate the prediction of classes and words, an MET-LM predicts the future word and class as a single unit, or *token*. While Maximum Entropy models are often slow due to the computation of normalization constants, we modify MET-LMs by truncating histories when back-off occurs in such a way that all normalization constants can be pre-computed and cached. We provide a new way of explaining MET-LMs, and show how they can be efficiently implemented using a similar data structure as has been used for word n-gram models.

For some tasks, training data may be available from multiple sources, and better performance can be achieved by linearly interpolating separate LMs built on each sub-corpus as compared to building a single LM on the combined data. A general way to implement such a combined model is to compute the score of each component LM on the fly and then to linearly interpolate scores. Typically, this will be many times slower than just evaluating a single LM. Thus, there is a large speed benefit if one can *statically* combine multiple LMs into a single LM without loss in performance. While such an algorithm is known for word n-gram models [4], no comparable algorithm exists for class n-gram models. In this work, we show that by interpolating the *data* rather than the *models*, we can achieve comparable performance with no extra effort at training or decoding time. This is similar in spirit to *count merging* [5] where the component corpora are weighted unevenly, and the challenge lies in smoothing effectively as most smoothing methods are poorly suited to "weighted" counts. Here, we show how to choose weights so that existing Maximum Entropy regularization methods give good performance.

The rest of the paper is organized as follows: Sec. II introduces MET-LM and compares it with Model M. Sec. III describes the new fast decoding strategy and efficient implementation for MET-LM. Sec. IV discusses our method for data interpolation, as well as a method for pruning the resulting model. Experimental results on machine translation tasks are presented in Sec. V including comparisons with word n-gram models and Model M, followed by conclusions and future work in the last section.

II. Maximum Entropy Token-based Language Models

A. Model Introduction

Maximum Entropy Token-based Language Model (MET-LM) [1] is designed to serve as a framework to explore local

[1]We discuss only ME class-based models here, as they have been shown to significantly outperform non-ME class-based models, e.g., [3].

dependencies between words and word classes or labels. It is an n-gram ME LM, predicting the future word and class based on the preceding words and word classes using a unified exponential model:

$$p_{\text{ME}}(w_i, c_i | w_{i-n+1}^{i-1}, c_{i-n+1}^{i-1}) = \frac{e^{\sum_k \lambda_k f_k(w_{i-n+1}^{i}, c_{i-n+1}^{i})}}{Z(w_{i-n+1}^{i-1}, c_{i-n+1}^{i-1})} \quad (1)$$

where $Z(w_{i-n+1}^{i-1}, c_{i-n+1}^{i-1}) = \sum_{w^i, c^i} e^{\sum_k \lambda_k f_k(w_{i-n+1}^{i}, c_{i-n+1}^{i})}$ ensures p_{ME} is a probability distribution; c_i is the class of word w_i; (w_i, c_i) altogether is called a *token*; $f_k(\cdot)$ is the k-th binary feature function with a word/class n-gram as the argument; and λ_k is a real-valued weight for feature f_k which is optimized during the ME training process.

Unlike conventional class-based LMs where the future word is usually predicted after its class is determined from the history, MET-LM predicts the future word and word class simultaneously. In other words, MET-LM models the distribution of tokens yet the features are still based on words and classes. We will show later that this unified modeling philosophy supports new types of class-based features which have not been explored in conventional class-based LMs. Although using two separate distributions can speed up model training [6], a unified distribution is beneficial for many real-time applications where faster decoding is preferred over faster training.

MET-LM is a general framework which theoretically allows all types of word classes including context-dependent ones such as part-of-speech tags. With context-dependent classes, it is not straightforward to express a conditional probability $p(w_i | w_{i-n+1}^{i-1})$ without using history class information. However, there is a natural representation for the probability of a complete sentence: $p(w_1^m) = \sum_{c_1^m} p(w_1^m, c_1^m)$, which can be easily computed using the forward algorithm to sum over the probabilities of all possible token sequences.

To avoid introducing more sparsity, MET-LM only incorporates m-gram ($m = 1..n$) features which are more general than the corresponding word m-gram. That is, an m-gram MET-LM feature has m positions where at each position, either a word *or* a class has to be chosen. For example, $f(c_{i-1} = \text{VERB}, c_i = \text{NOUN})$ is a valid MET-LM feature, but $f(c_{i-1} = \text{VERB}, w_{i-1} = \text{eat}, w_i = \text{cheese})$ is not. Let's use W to denote a word and C to denote a class. CCW denotes trigram features of the form $f(c_{i-2}, c_{i-1}, w_i)$ while WC denotes bigram features of the form $f(w_{i-1}, c_i)$, etc. Features such as CW and CCW are unusual in non-token-based language models because they associate history classes and future words directly. Here are all possible feature types for a trigram MET-LM: W, WW, WWW, C, WC, CW, CC, WCW, WWC, CWW, CCW, CWC, WCC and CCC. In practice, a MET-LM includes only a subset of these feature types because including all of them makes training extremely slow.

A feature is *activated* for an event if and only if the event matches the argument pattern. For example, given WW, WC and CC features, the event "eat cheese" activates the feature $f(\text{eat, cheese})$ as well as $f(\text{eat, NOUN})$ and $f(\text{VERB, NOUN})$ where VERB is the class of "eat" and NOUN is the class of "cheese".[2] When a feature f_k is activated, its weight λ_k contributes to the calculation of p_{ME}. Later, we will show that feature activation can be further constrained as long as p_{ME} remains a valid distribution.

B. Model Training and Feature Selection

MET-LM training incorporates the unigram caching method proposed in [7]. This method speeds up the Z calculation in that for each history, only future words which can activate conditional features are examined instead of all words in the vocabulary. However, this algorithm is less effective with class-based features. For example, given the presence of the feature $f(\text{VERB, NOUN})$, all nouns must be enumerated with the history "eat" instead of just those following "eat" in the training data. Training speed is therefore an important concern when selecting feature types for a MET-LM.

MET-LMs are smoothed using Gaussian priors [8] where different variances are used for each feature type. Prior variances are chosen according to model performance on held-out data.[3] Given the long training time of class-based models, we optimize variances only for word-based MET-LMs and use these values to set the priors for class-based features. We observe empirically that lower-level n-gram features tend to have larger prior variances than higher-level n-gram features. In addition, for the same value of n, class-based features tend to have smaller prior variances than purely word-based features. This is consistent with the intuition that class-based features, though more frequently observed, are less predictive than word-based features due to class qualities.

C. Comparing with Model M

Model M [8] is a recently developed ME language model which has shown significant improvement in real ASR applications. As in conventional class-based LMs, Model M is composed of two separate distributions for predicting future word classes and words, respectively:

$$p_{\text{model-M}}(w_i | w_{i-n+1}^{i-1}) = p(c_i | w_{i-n+1}^{i-1}, c_{i-n+1}^{i-1}) p(w_i | w_{i-n+1}^{i-1}, c_i)$$

This factorization can speed up training since one need only normalize over words from the predetermined future class.

Both MET-LM and Model M use the ME framework to combine word n-gram features (containing the same information as a regular word n-gram model) with class-based features. However, while their class-based features overlap, they are not identical. MET-LM can exploit features such as $f(c_{i-2}, c_{i-1}, w_i)$ that are absent from Model M, to directly model the relationship between history classes and future words. On the other hand, Model M can limit the future word choice of a given word history with a feature of the form $f(w_{i-1}, c_i, w_i)$, while MET-LM can only do this indirectly.

[2]For simplicity, we abbreviate a feature such as $f(w_{i-1} = \text{eat}, w_i = \text{cheese})$ as $f(\text{eat,cheese})$.

[3]Using the conventions from [8], we found σ^2 to be fairly stable (within the range 0.1–4) across training set sizes.

III. Back-off Lookup for MET-LM

One of the big obstacles in applying ME LMs to real applications is decoding speed. ME LMs are generally much slower than conventional n-gram LMs due to the time-consuming process of computing Z values. One way to alleviate this issue is to pre-calculate and save all Z values for histories seen in the training data. However, the Z calculation for unseen histories is still very expensive given that all words in the vocabulary must be enumerated. On the other hand, it is also impractical to save Z for all possible histories. In this section, we propose a back-off lookup strategy for ME LMs. Instead of computing the Z value on the fly for an unseen history, we back-off recursively to the longest seen n-gram history. This back-off method avoids any online Z computation and thus greatly speeds up lookups.

Note that for MET-LMs, probabilities are conditioned on token histories rather than word histories (Eq. 1). A word history may be mapped to multiple token histories if word classes are context-dependent. In this section, we focus on the simplified situation where each word belongs to only a single word class.[4] In this scenario, the Z value for a word history equals that of the corresponding token history: $Z(w_{i-n+1}^{i-1}) = Z(w_{i-n+1}^{i-1}, c_{i-n+1}^{i-1})$. Moreover, the conditional probability of the future word equals the conditional probability of the future token and the probability of a word sequence equals the probability of a token sequence.

In this simplified situation, MET-LM back-off lookup speed is comparable to that of conventional n-gram LMs while retaining the performance advantage of a class-based LM. The following is the back-off recipe for a trigram MET-LM, where H denotes all observed histories in the training data and $Z_0 = \sum_w \exp(\sum_k \lambda_k f_k(w))$:

$$p_b(w_3|w_1,w_2) = \begin{cases} \frac{\exp(\sum_k \lambda_k f_k(w_1^3, c_1^3))}{Z(w_1,w_2)} & \text{if } (w_1,w_2) \in H \\ p_b(w_3|w_2) & \text{otherwise} \end{cases}$$

$$p_b(w_3|w_2) = \begin{cases} \frac{\exp(\sum_k \lambda_k f_k(w_2^3, c_2^3))}{Z(w_2)} & \text{if } w_2 \in H \\ p_b(w_3) = \frac{\exp(\sum_k \lambda_k f_k(w_3, c_3))}{Z_0} & \text{otherwise} \end{cases}$$

In the above formulation, the regular ME probability is computed only when the full history is seen in the training data. When the history is unseen, some long-range class-based features which are activated in a full ME lookup are ignored in both the numerator and denominator, so that p_b is still a valid distribution. For example, the feature $f(\text{VERB, NOUN, and})$ is not activated when computing $p_b(\text{and}|\text{eat cheese})$ if the history "eat cheese" does not occur in the training data. Because not all class-based features are always activated, this back-off strategy may lead to some performance loss as compared to the normal computation. The amount of loss depends mainly on the types of features in the model. For example, models with CWW features are affected more than models with WWC features. Finally, we note that this lookup strategy does not affect training, as training does not involve any unseen histories.

[4]MET-LM experiments with context-dependent word classes can be found in [1].

Next, we show that a back-off MET-LM, due to the types of features it allows, can be stored in a format similar to ARPA n-gram model format and can be queried by a quick table lookup. ARPA format is widely used to represent back-off n-gram language models. In this format, each word n-gram is associated with a conditional probability and a back-off parameter, if available. MET-LM also has either one or two values attached to each n-gram feature: the feature value and the Z value. In our implementation, Z values are attached to only word-based features, though their computation also involves class-based features.

All n-gram features are organized into a trie structure in reversed order. Each node in the trie corresponds to an m-gram from t_i to t_{i-m+1} where t is either a word or a word class. Given a query $p(w_i|w_{i-n+1}^{i-1})$, the lookup has two steps. In step one, we search the trie for the longest matching word history, starting from w_{i-1} and moving backward. Suppose we stop at w_{i-m}, in which case $Z(w_{i-m}^{i-1})$ will provide the normalization constant to use. In step two, we search the trie for all activated features from position i to position $i-m$. It is possible that the future word w_i can activate class-based features with words/classes at position before $i-m$. We ignore these features because they are not involved in computing $Z(w_{i-m}^{i-1})$.

IV. Data Interpolation and Model Pruning

A. Data Interpolation

The usual method for LM training with multiple data sources is to build separate LMs on each corpus and then to linearly interpolate all of the models using weights that optimize the likelihood of held-out data. The interpolation of conventional n-gram back-off LMs can be done off-line in the following manner: the combined LM contains the union of the n-grams found in the component LM's, and the conditional probability of each n-gram is set to the appropriate linearly interpolated value. Finally, back-off factors are set to make the model normalize correctly. However, this method is not applicable for general ME LM's, because parameters do not correspond directly to probabilities and neither feature weights nor normalization values can be computed directly through interpolation.

In the case where only an in-domain corpus and out-of-domain corpus are to be combined, various interpolation methods for ME LMs have been proposed [9], [10], [11], [12], [13]. In [12], an ME model built on the out-of-domain data is used as a prior model for an ME LM built on the in-domain data. In [11], a separate prior is built for each data set and a unified ME LM is trained on both data sets.

In this section, we address the issue by doing data interpolation rather than model interpolation. We interpolate n-gram counts from sub-corpora and build a single model on the combined counts. One challenge of this method is deciding how to select data interpolation weights without having to do an excessively expensive optimization. In our experiments, we

have found that using weights from the usual interpolation method (optimizing the perplexity of held-out data when doing conventional model interpolation) works well. In addition, scaling the highest LM interpolation weight to be 1 and the others proportionally makes regularization easier: We can then use similar σ values for our Gaussian prior as when building an LM on a single corpus (See Section II-B).

An ME LM built on interpolated data satisfies constraints of the following form (ignoring regularization):

$$\sum_{x,y} \hat{c}(x) p(y|x) f_k(x,y) = \sum_j \alpha_j \sum_{x_j,y_j} \hat{c}_j(x_j,y_j) f_k(x_j,y_j)$$

where x_j, y_j are histories and future words in the j-th sub-corpus; α_j is the real-valued interpolation weight for the j-th sub-corpus; $\hat{c}_j$ is the empirical count in the j-th sub-corpus; $\hat{c}(x) = \sum_j \alpha_j \hat{c}_j(x)$, which can be fractional, is the empirical history count from the weighted mixed data; and $p(y|x)$ is the ME LM to be trained. The right-hand side corresponds to feature targets, and our method can be regarded as interpolating feature targets.

Note that it is not clear how to apply conventional n-gram model smoothing techniques such as Katz [14], Witten-Bell [15] or Kneser-Ney smoothing [16] to data with fractional counts, as we have here. All of these smoothing techniques assume integer n-gram counts, utilizing *count of count* values (e.g., the number of unigrams) in determining discounts or back-off factors. Even "unnatural" integer counts can make these algorithms perform poorly. For example, consider a data set consisting of two copies of the same corpus, so that all counts are even. Good-Turing back-off will not work properly since many counts of counts are zero. In contrast, if we just cut prior variances in half, ME models with a Gaussian prior will produce the same exact model as with a single copy of the corpus. This suggests that ME smoothing is more robust with respect to "unnatural" texts as compared to conventional smoothing methods.

B. ME LM Pruning

In many real applications, LM pruning is a required step and thus has been studied extensively, e.g., [17], [18]. In this paper, pruning is based mainly on the values of features and partially on the nested structure of features. In ME modeling, the feature weight λ_k can be positive or negative. A feature with a small absolute weight value $|\lambda_k|$ has relatively little influence on a model, and thus we prune features with weight below a threshold. However, ME LM features are not independent of each other and pruning a feature with a small weight can have a significant impact on the weights of other features after retraining. In MET-LM, a feature $f(t_a^b)$ is called a *nested* feature of $f(t_c^b)$ if t_a^b is a word/class suffix sub-string of t_c^b. A feature value is strongly related to the feature values of its nested features because whenever this feature is activated, all of its nested features are activated too. Thus, for any feature passing the weight threshold, we keep all of its nested features as well.

After a MET-LM is built, features are pruned from higher level to lower level. A feature remains if either its value is above the threshold or it is a nested feature of a remaining higher-level feature. Empirically, we have found that it may be preferable to not prune any unigram or bigram features. After pruning, the remaining features are retrained.

V. Experimental Results

We investigate three translation tasks to demonstrate the effectiveness of our back-off strategy and our data interpolation and pruning techniques for MET-LMs. The first task is standard IWSLT 2006 Chinese to English translation. The second is English to Dari translation and the third is Chinese to English translation. The latter two tasks are part of real-time speech to speech translation systems, and thus have strict requirements for language model size and speed.

The baseline models are 4-gram modified KN (mKN) LMs which are stored in regular ARPA format. The MET-LMs we evaluate include all word-based feature types and some types of class-based features. For each task, 150 word classes are generated from the task training data using the algorithm in [2]; the value 150 was determined empirically in previous work [12].

For each task, we build a hierarchical phrase-based machine translation model [19]. The relative weights of the language model and translation model are retuned for each LM. The weights tuned on held-out data are used to report translation performance on the test data.

A. IWSLT 2006 and Single-Source MET-LM

The IWSLT 2006 Chinese to English task [20] contains 39,953 parallel sentences including 1.6M English words with 11,146 unique words. The held-out data has 489 sentences with 7 references and the test data contains 500 sentences with 7 references.

We build several MET-LMs by varying the types of features used. Specifically, MetA contains WC, CW, CC, WCC, WCCC features. MetB is a much smaller model including only CC and WCC features. MetB-tag employs the same types of features as MetB but the class of each word is taken to be its dominant Part-of-Speech tag (out of 36) obtained from the UPenn TreeBank data [21]. All language models are unpruned.

We first compare the perplexities (PPL) on the test data with regular ME lookup and with back-off ME lookup. The test data is the union of all test references and contains 2.5% out-of-vocabulary words. This corresponds to a total of 47579 word predictions, of which 46.2% have trigram histories observed in the training data, 29.4% have only observed bigram histories, 21.9% have only observed unigram histories, and the remaining 2.5% are predicted with unknown histories.

Table I shows the perplexity results as well as the running time in seconds. All of the MET-LMs improve in perplexity over the baseline by over 15%. The back-off strategy incurs little performance loss as compared to the regular ME lookup, yet it is much faster. For example, regular ME lookup takes

TABLE I
COMPARING BACK-OFF STRATEGIES IN PPL AND SPEED FOR VARIOUS LMS

	Regular		Back-off	
LM	PPL	Time (s)	PPL	Time (s)
mKN	143			
MetA	119	260	121	1
MetB	120	147	122	1
MetB-tag	117	235	118	1

147 seconds with MetB while the back-off lookup takes no more than one second.

Table II presents the Chinese to English translation results. The size of each model is measured by the number of features in millions (M). The decoding time is measured on a single machine in seconds (s). From the table, we can see that all MET-LMs have achieved significant improvement in BLEU score as compared to the baseline with only a moderate model size increase. Particularly, MetA improves by 1.4 BLEU points with a model twice the size. MetB and MetB-tag achieve 0.6 and 1.0 BLEU improvements, respectively, with similar model sizes yet even faster decoding speeds. As a comparison, we have also implemented Model M without our back-off strategy. It provides a significant BLEU improvement of 1.3 yet has a much slower speed.

TABLE II
COMPARING LMS BY BLEU SCORE, MODEL SIZE AND RUNNING TIME ON IWSLT06 CHINESE TO ENGLISH TRANSLATION TASK

LM	Size (M)	Held BLEU	Test BLEU	Time (s)
mKN	0.44	21.61	21.74	174
MetA	0.87	22.86	23.21	172
MetB	0.53	22.83	22.42	148
MetB-tag	0.48	22.49	22.80	143
model-M	1.38	22.46	23.04	965

B. English to Dari Translation

We carried out training data interpolation experiments on an English(SVO)-to-Dari(SOV) task. The Dari LM training data is categorized into 11 groups [22], including a total of about 30M tokens with a vocabulary of 45K words. Both held-out data and test data have 1569 sentences with one reference.

Given the mixed nature of the Dari LM training data, Table III shows that better performance can be achieved with an interpolated LM than with a single LM constructed on the union of the training data. Even though a "unified" Model M and MET-LM can both reduce the perplexity from 155 to 137 and 138, respectively, the interpolated mKN LM can achieve a significantly better PPL (122) than all unified LMs. A MET-LM built with interpolated data can further bring down the perplexity to 112.[5] The above models are not pruned.

Four interpolated MET-LMs are built with interpolated Dari data using baseline interpolation weights. MetB1 and MetB2 both include CC and WCC features. They are pruned with threshold 0.1 and 0.2, respectively, and all unigram features remain. MetC1 and MetC2 both include CC, CCW, and CCCW features. MetC2 has the same pruning method and threshold as MetB2. MetC1 keeps all unigram and bigram features but uses a higher threshold of 0.5.

[5]The interpolation weights used here are a little different from the baseline interpolation weights suggested in Sec. IV-A. In particular, we discount the weights of large sub-corpora.

TABLE III
COMPARING LM INTERPOLATION WITH BUILDING A SINGLE LM ON THE UNION OF THE DATA FOR DARI

LM	PPL
mKN	155
Model M	137
MET-LM	138
interp mKN	122
interp MET-LM	112

Table IV shows the perplexities of MET-LMs before pruning (PPL), after pruning (Pr PPL), and using our back-off strategy on the pruned model (Pr-Bo PPL). First, we see that the pruned models yield almost the same perplexities as the unpruned models. Second, the back-off strategy yields almost the same results as regular ME lookups for MetB1 and MetB2. However, a larger perplexity increase is observed for MetC1 and MetC2. That is because MetC1 and MetC2 include lots of history-class-based features, such as CCW and CCCW. These features are more likely to be ignored during back-off.

TABLE IV
COMPARING UNPRUNED AND PRUNED MODELS WITH BACK-OFF ME LOOKUP FOR DARI

LM	PPL	Pr PPL	Pr-Bo PPL
MetB1	115	114	116
MetC1	115	115	119
MetB2	115	114	115
MetC2	115	114	119

Table V presents English-Dari translation performance with different LMs. All MET-LMs have better BLEU scores than the baseline model. Notably, MetC1 improves by 0.7 BLEU points over the baseline without increasing model size and decoding is only marginally slower. MetC2 is faster than MetC1 with a bigger model size because it has less bigram features remaining after pruning.

TABLE V
INTERPOLATED LMS ON ENGLISH/DARI TRANSLATION

LM	Size (M)	Held BLEU	Test BLEU	Time (s)
Interp-mKN	1.4	15.64	14.71	1351
MetB1	2.6	16.63	15.59	1509
MetC1	**1.4**	**16.47**	**15.43**	**1475**
MetB2	1.7	16.46	15.29	1480
MetC2	1.8	16.45	15.53	1374

C. Chinese to English Translation

Finally, we build and test MET-LMs on a Chinese-to-English translation task focusing on the travel domain. The English LMs are built on 900K sentences from 11 sources

containing 32M words using a 47kw vocabulary. There are 851 held-out sentences and 1000 test sentences, all with a single reference translation. All MET-LMs are interpolated models with CW, WC, CCW, and CCCW features. They are pruned with threshold 1, 1.5, and 2, respectively, without removing any unigram features. Table VI shows that all MET-LMs are able to improve the translation accuracy with virtually no overhead in decoding time. For example, MetD2 improves by 0.7 BLEU points (25.60 vs. 24.94) with only a 24% increase in model size and a 5% increase in decoding time.

TABLE VI
INTERPOLATED LMS ON CHINESE/ENGLISH TRANSLATION

LM	Held BLEU	Test BLEU	Size (M)	Time (s)
interp-mKN	27.01	24.94	2.9	352
MetD1	27.55	25.25	4.4	377
MetD2	**27.28**	**25.60**	**3.6**	**370**
MetD3	27.17	25.26	2.9	365

VI. CONCLUSIONS AND FUTURE WORK

Maximum Entropy language models have had limited use in real-time applications due to their computationally expensive training and lookup. In this paper, we have proposed a novel back-off lookup strategy for ME LMs, which avoids online Z computation by backing off to pre-computed Zs of seen histories. It loses little performance as compared to the regular ME lookup strategy yet is significantly faster.

When this strategy is applied to MET-LM with deterministic word classes, the model can be stored in an ARPA-like format and lookup speed is comparable to that of conventional n-gram LMs. Yet, it still offers performance comparable to the state-of-art class-based Model M. In the case where word classes are context-dependent, the Z value of a word history will depend on its class labels. The back-off idea is still applicable, but the algorithm will be more complex.

We have also proposed a new viewpoint for building ME LMs with multiple data components. Compared to online model interpolation, our method of data interpolation is faster and saves space. Though there is no fast way to compute optimal data interpolation weights, we have shown empirically that using the baseline LM interpolation weights to interpolate the data can lead to class-based MET-LMs with good performance.

We have noticed that the perplexity improvements of MET-LM in the interpolated case are not as big as those with a single data source. Sub-optimal interpolation weights may be a reason, and further investigation of this issue should be conducted.

VII. ACKNOWLEDGMENT

This work is partially supported by the DARPA Transformative Apps program under the contract number of N10PC20037A through the DARPA Information Processing Techniques Office (IPTO).Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

Many thanks to Martin CMEJREK, Ahmad EMAMI and Songfang HUANG for their valuable comments and suggestions on earlier drafts of this paper.

REFERENCES

[1] J. Cui, Y. Su, K. Hall, and F. Jelinek, "Investigating linguistic knowledge in a maximum entropy token-based language model," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2007.

[2] P. Brown, S. D. Pietra, V. D. Pietra, and R. Mercer, "The mathematics of machine translation: parameter estimation," *Computational Linguistics*, vol. 19, pp. 263–312, 1993.

[3] S. F. Chen, "Performance prediction for exponential language models," in *Proc. of HLT-NAACL*, 2009.

[4] A. Stolcke, "SRILM — an extensible language modeling toolkit," *In Proceedings of ICSLP*, vol. 2, pp. 901–904, 2002.

[5] M. Federico, "Bayesian estimation methods for n-gram language model adaptation," in *Proceedings of ICSLP*, 1996, pp. 240–243.

[6] J. Goodman, "Classes for fast maximum entropy training," in *Proceedings of ICASSP*, 2001.

[7] J. Wu and S. Khudanpur, "Efficient training methods for maximum entropy language modeling," in *Proceedings of ICSLP*, 2000, pp. 114–117.

[8] S. F. Chen, "Performance prediction for exponential language models," Technical Report RC 24671, IBM Research Division, Tech. Rep., 2008.

[9] C. Chelba and A. Acero, "Adaptation of maximum entropy capitalizer: Little data can help a lot," in *Proceedings of EMNLP*, 2004, pp. 285–292.

[10] H. Daume III and D. Marcu, "Domain adaptation for statistical classifiers," *Journal of Artificial Intelligence Research*, vol. 26, pp. 101–126, 2006.

[11] T. Alumäe and M. Kurimo, "Domain adaptation of maximum entropy language models," in *Proceedings of ACL*, Stroudsburg, PA, USA, 2010, pp. 301–306.

[12] S. F. Chen, "Shrinking exponential language models," in *Proceedings of HLT-NAACL*, 2009, pp. 468–476.

[13] J. Finkel and C. Manning, "Hierarchical Bayesian domain adaptation," in *Proceedings of HLT-NAACL*, 2009, pp. 602–610.

[14] S. M. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," in *IEEE Transactions on Acoustics, Speech and Signal Processing, ASSP-35(3)*, 1987, pp. 400–401.

[15] I. H. Witten and T. C. Bell, "The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression," *IEEE Transactions on Information Theory*, vol. 37(4), pp. 1085–1094, 1991.

[16] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 1, 1995, pp. 181–184.

[17] J. Goodman and J. Gao, "Language model size reduction by pruning and clustering," in *Proceedings of ICSLP*, 2000, pp. 110–113.

[18] A. Stolcke, "Entropy-based pruning of backoff language models," in *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, 1998, pp. 270–274.

[19] B. Zhou, X. Zhu, B. Xiang, and Y. Gao, "Prior derivation models for formally syntax-based translation using linguistically syntactic parsing and tree kernels," in *Proceedings of the Second Workshop on Syntax and Structure in Statistical Translation*, 2008, pp. 19–27.

[20] M. Paul, "Overview of the IWSLT 2006 evaluation campaign," in *Proc. of the International Workshop of Spoken Language Translation*, 2006, pp. 1–15.

[21] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of English: The Penn treebank," *Computational Linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[22] B. Xiang, B. Zhou, and M. Cmejrek, "Advances in syntax-based Malay-English speech translation," in *Proceedings of ICASSP*. Los Alamitos, CA, USA: IEEE Computer Society, 2009, pp. 4801–4804.