Feature Lattices for Maximum Entropy Modelling

Andrei Mikheev*
HCRC, Language Technology Group, University of Edinburgh,
2 Buccleuch Place, Edinburgh EH8 9LW, Scotland, UK.
e-mail: Andrei.Mikheev@ed.ac.uk

Abstract

Maximum entropy framework proved to be expressive and powerful for the statistical language modelling, but it suffers from the computational expensiveness of the model building. The iterative scaling algorithm that is used for the parameter estimation is computationally expensive while the feature selection process might require to estimate parameters for many candidate features many times. In this paper we present a novel approach for building maximum entropy models. Our approach uses the feature collocation lattice and builds complex candidate features without resorting to iterative scaling.

1 Introduction

Maximum entropy modelling has been recently introduced to the NLP community and proved to be an expressive and powerful framework. The maximum entropy model is a model which fits a set of pre-defined constraints and assumes maximum ignorance about everything which is not subject to its constraints thus assigning such cases with the most uniform distribution. The most uniform distribution will have the entropy on its maximum

Because of its ability to handle overlapping features the maximum entropy framework provides a principle way to incorporate information from multiple knowledge sources. It is superior to traditionally used for this purpose linear interpolation and Katz back-off method. (Rosenfeld, 1996) evaluates in detail a maximum entropy language model which combines unigrams, bigrams, trigrams and long-distance trigger words, and provides a thorough analysis of all the merits of the approach.

* Now at Harlequin Ltd.

The iterative scaling algorithm (Darroch&Ratcliff, 1972) applied for the parameter estimation of maximum entropy models computes a set of feature weights (λs) which ensure that the model fits the reference distribution and does not make spurious assumptions (as required by the maximum entropy principle) about events beyond the reference distribution. It, however, does not guarantee that the features employed by the model are good features and the model is useful. Thus the most important part of the model building is the feature selection procedure. The key idea of the feature selection is that if we notice an interaction between certain features we should build a more complex feature which will account for this interaction. The newly added feature should improve the model: its Kullback-Leibler divergence from the reference distribution should decrease and the conditional maximum entropy model will also have the greatest log-likelihood (L) value:

The basic feature induction algorithm presented in (Della Pietra et al., 1995) starts with an empty feature space and iteratively tries all possible feature candidates. These candidates are either atomic features or complex features produced as a combination of an atomic feature with the features already selected to the model's feature space. For every feature from the candidate feature set the algorithm prescribes to compute the maximum entropy model using the iterative scaling algorithm described above, and select the feature which in the largest way minimizes the Kullback-Leibler divergence or maximizes the log-likelihood of the model. This approach, however, is not computationally feasible since the iterative scaling is computationally expensive and to compute models for many candidate features many times is unreal. To

make feature ranking computationally tractable in (Della Pietra et al., 1995) and (Berger et al., 1996) a simplified process proposed: at the feature ranking stage when adding a new feature to the model, all previously computed parameters are kept fixed and, thus, we have to fit only one new constraint imposed by the candidate feature. Then after the best ranked feature has been established, it is added to the feature space and the weights for all the features are recomputed. This approach estimates good features relatively fast but it does not guarantee that at every single point we add the best feature because when we add a new feature to the model *all* its parameters can change.

In this paper we present a novel approach to feature selection for the maximum entropy models. Our approach uses a feature collocation lattice and selects candidate features without resorting to the iterative scaling.

2 Feature Collocation Lattice

We start the modelling process by building a sample space w to train our model on. The sample space consists of observed events of interest mapped to a set of atomic features Υ which we should define beforehand. Thus every observation from the sample space is a binary vector of atomic features: if an observation includes a certain feature, its corresponding bit in the vector is turned on (set to 1) otherwise it is 0.

When we have a set of atomic features Υ and a training sample of configurations w, we can build the feature collocation lattice. Such collocation lattice will represent, in fact, the factorial constraint space (χ) for the maximum entropy model and at the same time will contain all seen and logically implied configurations (w^+). Formally, the feature collocation lattice is a 3-ple: $\langle \theta, \subseteq, \xi^w \rangle$ where

- θ is a set of nodes of the lattice which corresponds to the union of the feature space of the maximum entropy model and the configuration space: $\theta = \chi \cup \Phi(w)$. In fact, the nodes in the lattice (θ) can have dual interpretation – on one hand they can act as mapped configurations from the extended configuration space (w^+) and on the other hand they can act as features from the constraint space (χ);
- $\subseteq$ is a transitive, antisymmetric relation over $\theta \times \theta$ – a partial ordering. We also will need the indicator function to flag whether the relation $\subseteq$ holds from node i to node k:

$$f_{\theta_i}(\theta_k) = \begin{cases} 1 & if\ \ \theta_i \subseteq \theta_k \\ 0 & otherwise \end{cases}$$

- ξ^w is a set of *configuration frequency counts* of the nodes (θ) of the lattice. This represents how many times we saw a particular configuration in our training samples. Because of the dual interpretation of the nodes, a node can also be associated with its *feature frequency count* i.e. the number of times we see this feature combination anywhere in the lattice. The *feature frequency* of a node will then be $\xi^\chi(\theta_k) = \sum_{\theta_i \in \theta} f_{\theta_k}(\theta_i) * \xi^w_{\theta_i}$ which is the sum of all the configuration frequency counts (ξ^w) of the descendant nodes.

  Suppose we have a lattice of nodes $A, B, [AB]$ with obvious relations: $A \subseteq [AB]; B \subseteq [AB]$:

  

  The configuration frequency ξ^w_A will be the number of times we saw A but not $[AB]$ and then the feature frequency of A will be: $\xi^\chi_A = \xi^w_A + \xi^w_{AB}$ i.e. the number of times we saw A in all the nodes.

When we construct the feature collocation lattice from a set of samples, each sample represents a feature configuration which we must add to the lattice as its node (θ_k). To support generalizations over the domain we also want to add to the lattice the nodes which are shared parts with other nodes in the lattice. Thus we add to the lattice all sub-configurations of a newly added configuration which are the intersections with the other nodes. We increment the configuration frequency ($\xi^w_{\theta_k}$) of a node each time we see in the training samples this particular configuration in full. For example, if a configuration $[ABCD]$ comes from a training sample and it is still not in the lattice, we create a node $[ABCD]$ and set its configuration frequency $\xi^w_{[ABCD]}$ to 1. If by that time there is a node $[ABDE]$ in the lattice, we then also create

the node $[ABD]$, relate it to the nodes $[ABCD]$ and $[ABDE]$ and set its configuration frequency to 0. If $[ABCD]$ had already existed in the lattice, we would simply incremented its configuration frequency: $\xi^w_{[ABCD]} = \xi^w_{[ABCD]} + 1$.

Thus in the feature lattice we have nodes with non-zero configuration frequencies, which we call *reference nodes* and nodes with zero configuration frequencies which we call latent or *hidden nodes*. Reference nodes actually represent the observed configuration space (w). Hidden nodes are never observed on their own but only as parts of the reference nodes and represent possible generalizations about domain: low-complexity constraints (χ) and logically possible configurations (w^+).

This method of building the feature collocation lattice ensures that along with true observations it contains hidden nodes which can provide generalizations about the domain. At the same time there is no over-generation of the hidden nodes: no logically impossible feature combinations and no hidden nodes without generalization power are included.

3 Feature Selection

After we constructed from a set of samples the feature collocation lattice $\langle \theta, \subseteq, \xi^w \rangle$, which we will call the empirical lattice, we try to estimate which features contribute and which do not to the frequency distribution on the reference nodes. Thus only the predictive features will be retained in the lattice. The optimized feature space can be seen as a feature lattice defined over the empirical feature lattice: $\theta' \subseteq \theta$ and initially it is empty: $\theta' = \emptyset$. We build the optimized lattice by incrementally adding a feature (atomic or complex) from the empirical lattice, together with the nodes which are the minimal collocations of this feature with the nodes already included into the optimized lattice. The necessity to add the collocations comes from the fact that the features (or nodes) can overlap with each other and we want to have a unique node for such overlap. So if in the optimized feature lattice there is just one feature A, then when we add the feature B we also have to add the collocation $[AB]$ if it exists in the empirical lattice. The configuration frequency of a node in the optimized lattice (ξ'^w) then can be computed as:

$$\xi'^w_{\theta_k} = \sum_{[\theta_i \in \theta \ \&\ \theta_i \notin \theta' \ \&\ \nexists \theta_j : \theta_j \in \theta' \ \&\ \theta_k \subseteq \theta_j \ \&\ \theta_j \subseteq \theta_i]} f_{\theta_k}(\theta_i) * \xi^w_{\theta_i} \quad (1)$$

Thus a node in the optimized lattice takes all configuration frequencies (ξ^w) of itself and the above related nodes if these nodes do not belong to the optimized lattice themselves and there is no higher node in the optimized lattice related to them.

Figure 1 shows how the configuration frequencies in the optimized lattice are redistributed when adding a new feature. First the lattice is empty. When we add the feature A to the optimized lattice (Figure 1.a), because no other features are present in the optimized lattice, it takes all the configuration frequencies of the nodes where we see the feature A: $\xi'^w_A = \xi^w_A + \xi^w_{AB} + \xi^w_{AC} + \xi^w_{ABC}$. Case b) of Figure 1 represents the situation when we add the feature B to the optimized lattice which already includes the feature A. Apart from the node B we also add the collocation of the nodes A and B to the optimized lattice. Now we have to redistribute the configuration frequencies in the optimized lattice. The configuration frequency of the node A now will become the number of times of seeing the feature A but not the feature combination AB: $\xi'^w_A = \xi^w_A + \xi^w_{AC}$. The configuration frequency of the node B will be the number of times of seeing the node B but not the node AB: $\xi'^w_B = \xi^w_B + \xi^w_{BC}$. The configuration frequency of the node AB will be: $\xi'^w_{AB} = \xi^w_{AB} + \xi^w_{ABC}$. When we add the feature C to the optimized lattice (Figure 1.c) we produce a fully saturated lattice identical to the empirical lattice, since the node C will collocate with the node A producing AC and will collocate with the node B producing BC. These nodes in their turn will collocate with each other and with the node AB producing the node ABC.

During the optimized lattice construction all the features (atomic and complex) from the empirical lattice compete, and we include the one which results in a optimized lattice with the smallest divergence $D(p \,||\, p')$ and equation ??) and therefore with the greatest log-likelihood $L_p(p')$, where:

- $p(\theta_i)$ is the probability for the i-th node in

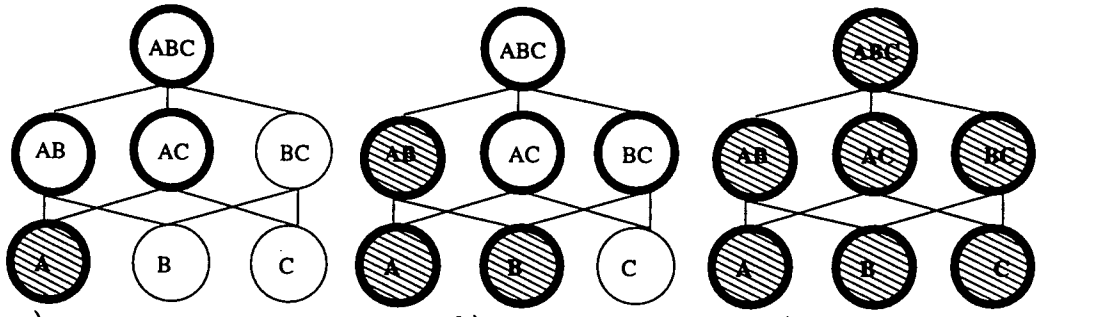

Figure 1: This figure shows the redistribution of the configuration frequencies in the optimized feature lattice when adding new nodes. Case a) stands for adding the feature A to the empty lattice, case b) stands for adding the feature B to the lattice with the feature A and case c) stands for adding the feature C to the lattice with the atomic features A and B and their collocations. The unfilled nodes stand for the nodes in the empirical lattice which don't have reference in the optimized lattice. The nodes in bold stand for the nodes decided by the optimized lattice (i.e. they can be assigned with non-default probabilities).

the empirical lattice:

$$p(\theta_i) = \frac{\xi^w_{\theta_i}}{N} \qquad where \quad N = \sum_{\theta_j \in \theta} \xi^w_{\theta_j} \quad (2)$$

- $p'(\theta_i)$ is the probability assigned to the i-th node using only the nodes included into the optimized lattice.

$$p'(\theta_i) = \begin{cases} \frac{\xi'^w_{\theta_i}}{N} & if \ \theta_i \in \theta' \\ \frac{\xi'^w_{\theta_j}}{N} & if \ \theta_i \notin \theta' \ \& \\ & [\exists \theta_j : \theta_j \in \theta' \ \& \ \theta_j \subseteq \theta_i] \ \& \\ & \quad [\nexists \theta_k : \theta_k \in \theta' \ \& \ \theta_k \subseteq \theta_i \ \& \ \theta_j \subseteq \theta_k] \\ 1/|Y| & otherwise \end{cases} \quad (3)$$

The optimized lattice assigns the probability to a node in the empirical lattice equal to that of its most specific sub-node from the optimized lattice. For reference nodes which do not have sub-nodes in the optimized lattice at all (undecided nodes) according to the maximum entropy principle we assign the uniform probability of making an arbitrary prediction.

For instance, for the example on Figure 1.b the optimized lattice includes only three nodes but there is just one undecided node (C) which is not shown in bold. So the probabilities for the nodes will be:

$$p'(A) = \frac{\xi'^w_A}{N} \quad p'(B) = \frac{\xi'^w_B}{N} \quad p'(AB) = \frac{\xi'^w_{AB}}{N}$$
$$p'(AC) = p'(A) \quad p'(BC) = p'(B)$$
$$p'(ABC) = p'(AB) \quad p'(C) = \frac{1}{|Y|}$$

N is the total count on the empirical lattice and is calculated as shown in equation 2:

$$N = \xi^w_A + \xi^w_B + \xi^w_C + \xi^w_{AB} + \xi^w_{AC} + \xi^w_{ABC}.$$

The presented above method provides us with an efficient way of selecting only important features from the initial set of candidate features without resorting to iterative scaling. When this way we add the features to the optimized lattice some candidate features might not sufficiently contribute to the probability distribution on the lattice. For instance, in the example presented on Figure 1, after we added the feature $[B]$ (case b) the only remaining undecided node was $[C]$. If the node $[C]$ is truly hidden (i.e. it does not have its own observation frequency) and all other nodes are optimally decided, there is no point to add the node $[C]$ into the lattice and instead of having 9 nodes we will have only

3. Another consideration which we apply during the lattice building is to penalize the development of low frequency (but not zero frequency) nodes i.e. the nodes with no reliable statistics on them. Thus we smooth the estimates on such nodes with the uniform distribution (which has the entropy on its maximum):

$$p''(\theta_i) = L * \frac{1}{|Y|} + (1 - L) * p'(\theta_i) \quad where \quad L = \frac{THRESHOLD}{THRESHOLD + \xi'^{w}_{\theta_i}}$$

So for high frequency nodes this smoothing is very minor but for nodes with frequencies less than two thresholds the penalty will be considerable. This will favor nodes which do not create sparce collocations with other nodes.

The described method is similar in spirit to the method of word trigger incorporation to a trigram model suggested in (Rosenfeld, 1996): if a trigram predicts well enough there is no need for an additional trigger. The main difference is that we do not recompute the maximum entropy model every time but use our own frequency redistribution method over the collocation lattice. This is the crucial difference which makes a tremendous saving in time. We also do not require a newly added feature to be either atomic or a collocation of an atomic feature with a feature already included into the model as it was proposed in (Della Pietra et al., 1995) (Berger et al., 1996). All the features are created equal and the model should decide on the level of granularity by itself.

4 Model Generalization

After we have chosen a subset of features for our model, we restrict our feature lattice to the optimized lattice. Now we can compute the maximum entropy model taking the reference probabilities (which are configuration probabilities) as in equation 3.

The nodes from the optimized lattice serve both as possible domain configurations and as potential constraint features to our model. We, however, want to constrain only the nodes with the reliable statistics on them in order not to overfit the model. This in its turn will take off certain computational load, since we expect a considerable number of fragmented (simply infrequent) nodes in the optimized lattice. This comes from the requirement to build all the collocations when we add a new node. Although many top-level nodes will not be constrained, the information from such infrequent nodes will not be lost completely – it will contribute to more general nodes since for every constrained node we marginalize over all its unconstrained descendants (more specific nodes). Thus as possible constraints for the model we will consider only those nodes from the optimized lattice, whose *marginalized over responses* feature frequency counts[1] are greater than a certain threshold, e.g.: $\xi'^{\chi}_{\theta=(x,Y)} > 5$. This consideration is slightly different from the one suggested in (Ristad, 1996) where it was proposed to unconstrain nodes with infrequent *joint* feature frequency counts. Thus if we saw a certain feature configuration say 5,000 times and it always gave a single response we suggest to constrain as well the observation that we never saw this configuration with the other responses. If we applied the suggestion of (Ristad, 1996) and cut out on the basis of the joint frequency we would lose the negative evidence, which is quite reliable judging by the total frequency of the observation.

Initially we constrain all the nodes which satisfy the above requirement. In order to generalize and simplify our maximum entropy model, we unconstrain the most specific features, compute a new simplified maximum entropy model, and if it still predicts well, we repeat the process. So our aim is to remove from the constraints as many top level nodes as possible without losing the model fitness to the reference distribution ($\tilde{p}$) of the optimized feature lattice. The necessary condition for a node to be taken as a candidate to unconstrain, is that this node shouldn't have any constrained nodes above it. There is also a natural ranking for the candidate nodes: the closer to 1 the weight (λ) of a such a node is, the less it is important for the model. We can set a certain threshold on the weights, so all the candidate nodes whose λs differ from 1 less than this threshold will be unconstrained in one go. Therefore we don't have to use the iterative scaling for feature ranking and apply it only for model recomputation, possibly un-constraining several feature configurations (nodes) at once. This method, in fact, resembles the Backward Sequential Search

[1] $\xi'^{\chi}(\theta_k) = \sum_{\theta_i \in \theta'} f_{\theta_k}(\theta_i) * \xi'^{w}_{\theta_i}$

(BSS) proposed in (Pedersen&Bruce, 1997) for decomposable models. There is also a significant reduction in computational load since the generalized smaller model deviates from the previous larger model only in a small number of constraints. So we use the parameters of that larger model[2] as the initial values for the iterative scaling algorithm. This proved to decrease the number of required iterations by about ten-fold, which makes a tremendous saving in time.

There can be many possible criteria when to stop the generalization algorithm. The simplest one is just to set a predefined threshold on the deviation $D(\tilde{p} \mid\mid p)$ of the generalized model from the reference distribution. (Pedersen&Bruce, 1997) suggest to use Akaike's Information Criteria (AIC) to judge the acceptability of a new model. AIC rewards good model fit and penalizes models with high complexity measured in the number of features. We adopted the stop condition suggested in (Berger et al., 1996) – the maximization of the likelihood on a cross-validation set of samples which is unseen at the parameter estimation.

5 Application: Fullstop Problem

Sentence boundary disambiguation has recently gained certain attention of the language engineering community. It is required for most text processing tasks such as, tagging, parsing, parallel corpora alignment etc., and, as it turned out to be, this is a non-trivial task itself. A period can act as the end of a sentence or be a part of an abbreviation, but when an abbreviation is the last word in a sentence, the period denotes the end of a sentence as well. The simplest "period-space-capital_letter" approach works well for simple texts but is rather unreliable for texts with many proper names and abbreviations at the end of sentence as, for instance, the Wall Street Journal (WSJ) corpus ((Marcus et al., 1993)).

One well-known trainable systems – SATZ – is described in (Palmer&Hearst, 1997). It uses a neural network with two layers of hidden units. It was trained on the most probable parts-of-speech of three words before and three words after the period using 573 samples from the WSJ corpus. It was then tested on unseen 27,294 sentences from the same corpus and achieved 1.5% error rate. Another automatically trainable system described in (Reynar&Ratnaparkhi, 1997). This system is similar to ours in the model choice – it uses the maximum entropy framework. It was trained on two different feature sets and scored 1.2% error rate on the corpus tuned feature set and 2% error rate on a more portable feature set. The features themselves were words and their classes in the immediate context of the period mark. (Reynar&Ratnaparkhi, 1997) don't report on the number of features utilized by their model and don't describe their approach to feature selection but judging by the time their system was trained (18 minutes[3]) it did not aim to produce the best performing feature-set but estimated a given one.

To tackle this problem we applied our method to a maximum entropy model which used a lexicon of words associated with one or more categories from the set: abbreviation, proper noun, content word, closed-class word. This model employed atomic features such as the lexicon information for the words before and after the period, their capitalization and spellings. For training we collected from the WSJ corpus 51,000 samples of the form $\langle Y, F..F \rangle$ and $\langle N, F..F \rangle$, where Y stands for the end of sentence, N stands for otherwise and Fs stand for the atomic features of the model. We started to built the model with 238 most frequent atomic features which gave us the collocation lattice of 8,245 nodes in 8 minutes of processor time on five SUN Ultra-1 workstations working in parallel by means of multi-threading and Remote Process Communication. When we applied the feature selection algorithm (section 3), we in 53 minutes boiled the lattice down to 769 nodes. Then constraining all the nodes, we compiled a maximum entropy model in about 15 minutes and then using the constraint removal process in two hours we boiled the constraint space down to 283. In this set only 31 atomic features remained. This model was detected to achieve the best performance on a specified cross-validation set. For the evaluation we used the same 27,294 sentences as in (Palmer&Hearst, 1997)[4] which

[2] instead of the uniform distribution as prescribed in the step 1 of the Improved Iterative Scaling algorithm.

[3] Personal communication

[4] We would like to thank David Palmer for making his test data available to us.

were also used by (Reynar&Ratnaparkhi, 1997) in the evaluation of their system. These sentences, of course, were not seen at the training phase of our model. Our model achieved 99,2477% accuracy which is the highest quoted score on this test-set known to the authors. We attribute this to the fact that although we started with roughly the same atomic features as (Reynar&Ratnaparkhi, 1997) our system created complex features with higher prediction power.

6 Conclusion

In this paper we presented a novel approach for building maximum entropy models. Our approach uses a feature collocation lattice and selects the candidate features without resorting to iterative scaling. Instead we use our own frequency redistribution algorithm. After the candidate features have been selected we, using the iterative scaling, compute a fully saturated model for the maximal constraint space and then apply relaxation to the most specific constraints.

We applied the described method to several language modelling tasks such as sentence boundary disambiguation, part-of-speech tagging, stress prediction in continues speech generation, etc., and proved its feasibility for selecting and building the models with the complexity of tens of thousands constraints. We see the major achievement of our method in building compact models with only a fraction of possible features (usually there is a few hundred features) and at the same time performing at least as good as state-of-the-art: in fact, our sentence boundary disambiguater scored the highest known to the author accuracy (99.2477%) and our part-of-speech tagging model generalized for a new domain with only a tiny degradation in performance.

A potential drawback of our approach is that we require to build the feature collocation lattice for the whole observed feature-space which might not be feasible for applications with hundreds of thousands of features. So one of the directions in our future work is to find efficient ways for a decomposition of the feature lattice into non-overlapping sub-lattices which then can be handled by our method. Another avenue for further improvement is to introduce the "or" operation on the nodes of the lattice. This can provide a further generalization over the features employed by the model.

7 Acknowledgements

The work reported in this paper was supported in part by grant GR/L21952 (Text Tokenisation Tool) from the Engineering and Physical Sciences Research Council, UK. We would also like to acknowledge that this work was based on a long-standing collaborative relationship with Steve Finch.

References

A. Berger, S. Della Pietra, V. Della Pietra, 1996. A Maximum Entropy Approach to Natural Language Processing In *Computational Linguistics* vol.22(1)

J.N. Darroch and D. Ratcliff 1972. Generalized Iterative Scaling for Log-Linear Models. *The Annals of Mathematical Statistics*, 43(5).

S. Della Pietra, V.. Della Pietra, and J. Lafferty 1995. Inducing Features of Random Fields Technical report CMU-CS-95-144

M. Marcus, M.A. Marcinkiewicz, and B. Santorini 1993. Building a Large Annotated Corpus of English: The Penn Treebank. In *Computational Linguistics*, vol 19(2), ACL.

D. D. Palmer and M. A. Hearst 1997. Adaptive Multilingual Sentence Boundary Disambiguation. In *Computational Linguistics*, vol 23(2), ACL. pp. 241-269

T. Pedersen and R. Bruce 1997. A New Supervised Learning Algorithm for Word Sense Disambiguation. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, Providence, RI.

J. C. Reynar and A. Ratnaparkhi 1997. A Maximum Entropy Approach to Identifying Sentence Boundaries. In *Proceedings of the Fifth ACL Conference on Applied Natural Language Processing (ANLP'97)*, Washington D.C., ACL.

E. S. Ristad 1996. Maximum Entropy Modelling Toolkit. *Documentation for Version 1.3 Beta, Draft*,

R. Rosenfeld 1996. A Maximum Entropy Approach to Adaptive Statistical Language Learning. In *Computer Speech and Language*, vol.10(3), Academic Press Limited, pp. 197-228