

Maximum Entropy modeling for MWE identification. Comparison with decision trees.

Begoña Villada Moirón
Alfa-Informatica
University of Groningen
m.b.villada.moiron@rug.nl

February 16, 2007

1 Introduction

This report begins with a description of the maximum entropy framework commonly used in modeling natural language processing tasks. As far as our knowledge goes, this represents a first attempt to model multiword expression identification using a maximum entropy model. A related sort of models, loglinear models, have been previously used for automated identification of phrasal verbs (eg. *look something up*) in English corpora [Blaheta and Johnson, 2001] and for identification of support verb constructions (e.g. *iem. op de hoogte houden*) in Dutch corpora [Villada Moirón, 2005], however, the modeling framework is entirely different since it is a fully unsupervised method.

I continue by describing an attempt to use explicit linguistic information as features. As stated below, despite the effort put into it, this attempt has not yet been successful. Next, I report on the settings and experiments done by using the same numeric features employed in identification of MWEs using decision trees. This is followed by a report on the results achieved and the evaluation.

2 Maximum Entropy Model

2.1 Modeling a random process

(The following description summarizes and adapts parts of the CL paper by Berger et al. [1996].)

Modeling an NLP process (e.g. POS-tagging, parse selection, text categorization) using Maximum Entropy requires two tasks:

- extracting a set of facts about the process that generated our training data and,

- constructing a model of this process.

We assume that the training data was generated by a random process. This random process produces an output value y , a member of a finite set Y . If we are modeling a translation system, the random process produces a translation in the target language for a word in the source language. E.g. the English verb *to be* can be translated into Spanish by one of two words $\{estar, ser\}$. The translation system may produce one of two outcomes *ser* or *estar*. If we are modeling MWE identification, the random process will return the outcome that has the highest probability, i.e. the outcome will be MWE or NON-MWE depending on which label has been assigned a higher probability.

The maximum entropy method follows a simple principle: “model all that is known about the data and assume nothing about what is unknown (Berger, tutorial)” .

The task is to construct a stochastic model that accurately represents the behavior of the random process. Such a model is a method of estimating the conditional probability that, given a context x , the random process will output y $p(y|x)$.

On training data We compute the empirical probability distribution \tilde{p} defined by

$$\tilde{p}(x, y) \equiv \frac{1}{N} x \mid \text{number of times } (x, y) \text{ occurs in sample} \quad (1)$$

This is the empirical distribution of seeing an outcome y given a context x . Here, the distribution is over the set of all possible outcomes Y and the set of all possible contexts X .

In the MWE identification task, the context can be a set of features that represent different types of linguistic information. E.g. the context can be a vector that describes whether a candidate expression allows passive, the position of the constituent phrase, verb finiteness, etc.

Features and constraints Constraints on the context can be added. In the translation example, we may want to express that En. *be* translates as Sp. *estar* when the adjective *contenta* ‘glad’ follows *estar*. To express this, a **feature function** (or **feature**) is used:

$$f(x, y) = \begin{cases} 1 & \text{if } y=\text{estar and 'contenta' follows 'estar'} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The expected value of f with respect to the empirical distribution $\tilde{p}(x, y)$ is the statistic we want. This expected value is denoted by,

$$\tilde{p}(f) \equiv \sum_{x,y} \tilde{p}(x,y) f(x,y) \quad (3)$$

For a given feature pair (context,outcome), the above statistic checks the feature's estimated value in the whole empirical probability distribution. Berger mentions that when we encounter a statistic that we believe is useful, we acknowledge its importance by requiring that our model accord with it. This is done by constraining the expected value that the model assigns to the corresponding feature function f . The expected value of f with respect to the model $p(y|x)$ (the model we seek) is

$$p(f) \equiv \sum_{x,y} \tilde{p}(x)p(y|x)f(x,y) \quad (4)$$

Then, this expected value is constrained to be the same as the expected value of f in the training data:

$$p(f) = \tilde{p}(f) \quad (5)$$

This is a requirement, so-called a *constraint*.
Combining the last three equations yields:

$$\sum_{x,y} \tilde{p}(x)p(y|x)f(x,y) = \sum_{x,y} \tilde{p}(x,y)f(x,y) \quad (6)$$

The importance of constraints: if we only pay attention to those models $p(y|x)$ that satisfy these constraints, we eliminate those models which do not agree with the training sample.

So far we have:

- a means of representing statistical phenomena inherent in a sample of data (namely $\tilde{p}(f)$)
- a means of requiring that our model (desired) exhibit these phenomena (namely, $p(f) = \tilde{p}(f)$)

Among all the models that satisfy the required constraints $p \in C$, the maximum entropy philosophy says that we select the model with the most uniform distribution. How do we measure uniformity? Measuring the conditional entropy of the conditional distribution $p(y|x)$:

$$H(p) \equiv - \sum_{x,y} \tilde{p}(x)p(y|x) \log p(y|x) \quad (7)$$

Having a way to measure uniformity, the principle of maximum entropy states that: "To select a model from a set C of allowed probability distributions, choose the model $p \in C$ with maximum entropy $H(p)$ ":

$$p^* = \arg \max_{p \in C} H(p) \quad (8)$$

This is the model we seek and the model most likely to have generated our training data. This model is shown in equation 9, where Z is a normalizing constant and the λ term gives the weight associated with each feature.

$$p(y|x) = \frac{1}{Z_x} \exp\left[\sum_{i=0}^n \lambda_i f_i(y, x)\right] \quad (9)$$

Various software implementations exist of the maximum entropy framework. Although, they vary in their functionality, some of them will find the model showing the maximum entropy given as large a set as possible of training data. Next, the found model is evaluated on test data, to estimate its performance on predicting the probability of having an outcome assigned to unseen events.

2.2 Software for MaxEnt modeling

Several software implementations exist that facilitate the MaxEnt modeling. I chose two, TADM¹ originally developed by Rob Malouf and MAXENT² developed by Zhang Le.

3 Modeling multiword expression identification

Given a list of candidate expressions, we want to model the process that assigns a certain class to each expression. Candidate expressions may fall in class 'MWE' or class 'non-MWE'. Thus, the model should assign to an expression the class with the highest probability. This model is trained on a set of training data (already labeled data) and later tested on unseen data.

Next, I describe two attempts to establish a well performing identification model.

3.1 Using explicit linguistically-informed features

Any information that can be counted and may play a role in distinguishing MWES from non-MWES is potentially useful. The experiments described here concern identification of VERB NP (`verb obj1`) and VERB (NP) PP (`verb obj1 pc`) patterns.

Table 1 summarizes the information selected for each pattern.³ In modelling VERB NP identification, information about the PREP, `dependency`

¹<http://tadm.sourceforge.net/>.

²http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html.

³These raw features extracted from a parsed corpus are exactly the same as those used in MWE identification using decision trees.

(1)	verb	verb tense	lexeme finite/infinitive
(2)	PP	head obj NP location dependency relation	preposition head noun wrt to verb
(3)	NP	determiner adjective post-nominal mod number	head POS, lexeme singular/plural
(4)		pronoun	yes/no
(5)	SUBJ	head noun	
(6)	OBJ1	head noun	
(7)	passive	yes/no passive auxiliary	

Table 1: Raw features extracted from parsed data.

relation and OBJ1 head noun is not applicable. (The dependency relation assigned to the NP is always `obj1`).

From the raw features, I select a set of explicit information to extract potentially interesting characteristics of the data. This set is shown in figure 1. To give a concrete example: a candidate expression is represented as a **context** vector that consists of features describing the expression. In principle, no restrictions apply on the length of the vector, however, rare features (or rare expressions) may lead to data sparseness.

From the features enumerated in table 1, we construct feature functions. Thus, in the identification task a **context** vector consists of a set of feature functions f_i . Before experimenting with this framework, I ignore how complex the feature functions can be and also, how informative (or uninformative) these may be for identifying MWES. This is the reason to try out whether the feature functions shown in figure 2 can be of any use.

Synthesizing, candidate expressions in training data are represented as a context vector consisting of feature functions. Figure 3 shows an example. Each feature function is cross-classified with a class in training data. Class 0 represents a non-MWE.

The TADM maximum entropy package expects an event file in which all events are encoded as context vectors. Then, the parameter weights for the selected feature functions can be estimated. These weights correspond to the $\lambda_i f_i(y, x)$ in the final model shown in equation 9.

To be able to compare the performance of the decision trees to that of maximum entropy we used the same corpus (`clef`) to extract candidate

- verb lex and pos
- prep lex and pos
- noun (obj of prep) lex and pos (noun)
 - begins with capital letter => name entity (assumption)
 - contains a digit
- P_N tuple
- PN position in sentence (i_{pr},p_{r3},f_{o3},...)
- finiteness of the verb (1|0)
- frame assigned by the parser
- dependency relation between verb and PN tuple
- subject lexeme
- obj1 lexeme | or' _' if not available
- determiner POS preceding Noun
- determiner lexeme preceding Noun
- prenominal modification: [mod,nul] or [POS,lexeme]

- postnominal modification: [POS, head_lexeme]
- Noun number: sg|pl
- diminutive|nodim
- pronoun: y|n

Figure 1: Feature set to extract potentially interesting facts.

- Lexical affinity related (4)
 - la_vp(VERB_lexeme,verb,PREP_lexeme)
 - la_pn(PREP_lexeme,NOUN_lexeme)
 - la_vn(VERB_lexeme,NOUN_lexeme)
 - la_vpn(VERB_lexeme,verb,PREP_lexeme,NOUN_lexeme)

- Local context of the verb (7)
 - lc_vdr(VERB_lexeme, dependency_relation)
 - lc_vdrp(VERB_lexeme, dependency_relation,Prep_lexeme)
 - lc_drpn(dependency_relation,Prep_lexeme,Noun_lexeme)

 - fr(VERB_lexeme,frame)
 - fr_dr(frame,dependency_relation)
 - fr_dr_p(frame,dependency_relation,Prep_lexeme)
 - fr_v_obj1(frame,Verb_lexeme,Obj1_lexeme)

- Morphological productivity or lack of it (2)
 - m_pn_n(Prep_lexeme,Noun_lexeme,Number_value) #number
 - m_pn_d(Prep_lexeme,Noun_lexeme,Diminutive_value) #diminutive

- ++ Determiner, adjectives (5)
 - det(Prep_lexeme,Noun_lexeme,DET_pos,DET_lexeme)
 - det(Verb_lexeme,Prep_lexeme,Noun_lexeme,DET_pos,DET_lexeme)
 - det(Verb_lexeme,Noun_lexeme,DET_pos,DET_lexeme)

 - adj(Prep_lexeme,Noun_lexeme,ADJ_pos,ADJ_lexeme)
 - adj(Verb_lexeme,Prep_lexeme,Noun_lexeme,ADJ_pos,ADJ_lexeme)

- Syntactic flexibility or not (8)
 - + Passive
 - p0(Verb_lexeme,Prep_lex,Noun_lex) --> if No passive seen
 - p1(Verb_lexeme,Prep_lex,Noun_lex,Auxiliary)

 - + Pronominalization
 - pr0(Verb_lexeme,Prep_lex,Noun_lex) --> if not pronominalized
 - pr1(Verb_lexeme,Prep_lex,Noun_lex) --> if pronominalized

Figure 2: Feature functions.

```

< la_vp(behooor,verb,bij,0),la_pn(bij,Avenue,0),
  la_vn(behooor,Avenue,0), la_vpn(behooor,bij,Avenue,0),
  lc_vdr(behooor,mod,0), lc_vdrp(behooor,mod,bij,0),
  lc_drpn(mod,bij,Avenue,0),m_pn_n(bij,Avenue,schapenscheerders,0),
  det0(bij,Avenue,det,_,0), det1(behooor,bij,Avenue,det,_,0),
  det2(behooor,Avenue,det,_,0),adj0(bij,Avenue,adj,weinige,0),
  adj1(behooor,bij,Avenue,adj,weinige,0), p0(behooor,bij,Avenue,0),
  pr0(behooor,bij,Avenue,0),loc1(behooor,bij,Avenue,ipr,0) >

```

Figure 3: Context vector representing an instance of the expression *bij Avenue behoor*.

expressions and their context vectors.

The proposed feature functions posed a problem. The explicit information included in the features leads to data sparseness. Consequently, the maximum entropy package cannot correctly estimate the $\lambda_i f_i(y, x)$ weights. As result, all feature functions are assigned a weight of '0'. Therefore, we cannot estimate $p(y|x)$ (probability of a class given a context vector) since it would yield a 0 probability value. This problem renders the current method unfeasible. Further research is needed in encoding the feature functions so that they capture generalizations.

3.2 Using numeric abstractions of linguistically-informed features

Instead of using explicit features like the ones encoded in the feature functions described above, one could measure certain linguistic characteristics in the set of candidates and use such quantitative information (or 'numeric abstractions') as features. Feature coding uses exactly the same features and the same measurements used in applying decision trees. Thus, for a description of feature coding and the quantitative measurements the reader is referred to Deliverable 1.3 (dated October 2006).

4 Experiments and Results

No actual results have been reached with the first approach, therefore, the results and evaluation refer to the maximum entropy method that uses numeric features. Maximum entropy is applied using `maxent`, a free toolkit developed by Zhang Le.

The maximum entropy method is applied in order to identify VERB (NP) PP and VERB NP MWES in the CLEF corpus. The datasets include almost 8,500 and 24,017 candidates, respectively. These numbers result after applying a frequency cutoff of 10.

	Number of iterations						Gaussian penalty
	30	60	80	90	110	130	
Accuracy	80.86	81.02	81.49	81.67	81.79	79.64	g=0
	80.87	81.27	81.21	81.07	81.35	81.65	g=0.2
	80.75	81.21	81.33	81.28	79.31	81.52	g=0.5
	80.80	80.70	80.50	81.29	81.29	81.58	g=0.7
Baseline	77.49						

Table 2: v (NP) PP experiment results (frequency ≥ 10).

	Number of iterations						Gaussian penalty
	30	60	80	90	110	130	
Accuracy	82.55	82.78	79.53	82.77	82.79	82.54	g=0
	82.56	82.69	82.74	82.74	82.75	82.76	g=0.2
	82.56	82.85	82.78	82.76	82.78	82.82	g=0.5
	82.56	82.72	82.74	79.9	82.76	82.82	g=0.7
Baseline	82.00						

Table 3: v NP experiment results (frequency ≥ 10).

For each syntactic pattern, I tried running the model several iterations to establish when the best accuracy is reached. The model allows another parameter, a Gaussian penalty. Tables 2 and 3 show the results for each settings, when varying the number of iterations and the Gaussian penalty parameters.

Some differences are observed while increasing the number of runs, showing that the performance of the model improves while raising the number of running cycles. The model estimates on the training data seem to become more reliable. Increasing the Gaussing penalty does not bring about much improvement.

Regarding VERB (NP) PP identification, the model performs better than baseline. The baseline is the accuracy reached by a model that always chooses the most frequent class, in both experiments, the non-MWE. Regarding VERB NP identification, the performance hardly improves over the baseline.

5 Comparison with decision trees

When evaluating decision trees, the WEKA software provides 'accuracy', precision, recall and f-measure given per class. Accuracy shows the proportion of candidate expressions that have been correctly classified. (For the definition of the other standard measures refer to Deliverable 1.3.).

Classifier	Accuracy	class 'y'			class 'n'		
		P	R	F	P	R	F
Decision trees	83.4	0.66	0.53	0.59	0.87	0.92	0.89
Maxent	81.79	0.62	0.39	0.47	0.84	0.93	0.88
Baseline	77.49	0	0	0	1.0	1.0	1.0

Table 4: Model comparison, v (NP) PP experiment.

Classifier	Accuracy	class 'y'			class 'n'		
		P	R	F	P	R	F
Decision trees	84.10	0.59	0.27	0.37	0.86	0.96	0.90
Maxent	82.85	0.52	0.17	0.25	0.84	0.96	0.896
Baseline	82.00	0	0	0	1.0	1.0	1.0

Table 5: Model comparison, v NP experiment.

5.1 Verb NP PP

Table 4 shows the performance reached by decision trees and maxent on the same v (NP) PP dataset. Decision trees reaches a better performance than maxent.

5.2 Verb NP

Table 5 shows the performance reached by decision trees and maxent on the same VERB NP dataset. Once again, decision trees reaches a higher accuracy than maxent. On this dataset, both methods perform worse than on the other dataset, with maxent hardly showing any improvement over the baseline.

5.3 Discussion

Precision (P), Recall (R) and F-measure (F) given in tables 4 and 5 are split per class. The measures related to each class show a huge difference, showing that the maxent classifier performs much better on the most frequent class (non-MWEs). For us, it is interesting to evaluate how well the classifier does in the identification of MWEs. Precision and recall are still very low, for this method to be used by e.g. computational lexicographers. In this respect, decision trees perform better on our data.

The training data we used in the VERB (NP) PP experiment is exactly the same as that used in the experiments performed with decision trees. The data used in the VERB NP includes a bigger dataset than the one used in earlier experiments; however, to compare decision trees to maxent on

the VERB NP data, we re-run the decision trees method on the new dataset again. Thus, the results given in Table 5 are directly comparable.

One possible explanation of the rather disappointing results might be that the maxent model is yet more sensitive than the decision trees to noise in training data. There are annotation errors in the training data and these may be the cause of a mediocre maxent performance.

After assessing that the quantitative results obtained with maxent are worse than those obtained with decision trees, I refrain from pursuing a qualitative evaluation of the maxent method. It is possible that the maxent method gives better results but more research needs to be devoted at finding better learning features.

6 Conclusions

In this report I presented the maximum entropy framework applied to the task of identifying MWES in large corpora. The comparison of decision trees and maxent reveals that the first framework performs better than the second one. These are somehow disappointing results, given that in other NLP tasks maxent typically reaches better performance than other inductive learning classifiers.

In my opinion, maxent may give better results if (i) training data contains none or very few errors and (ii) if more suitable features are fed as learning features. This issue is open for future investigation.

References

- A.L. Berger, V. Della Pietra, and S. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22 (1):39–71, 1996.
- Don Blaheta and Mark Johnson. Unsupervised learning of multi-word verbs. In *39th Annual Meeting and 10th Conference of the European chapter of the Association for Computational Linguistics (ACL39)*, pages 54–60, Toulouse, France, 2001. CNRS.
- Begoña Villada Moirón. *Data-driven Identification of fixed expressions and their modifiability*. PhD thesis, University of Groningen, 2005.