Factorizing Complex Models: A Case Study in Mention Detection

Radu Florian, Hongyan Jing, Nanda Kambhatla and **Imed Zitouni**
IBM TJ Watson Research Center
Yorktown Heights, NY 10598
{raduf,hjing,nanda,izitouni}@us.ibm.com

Abstract

As natural language understanding research advances towards deeper knowledge modeling, the tasks become more and more complex: we are interested in more nuanced word characteristics, more linguistic properties, deeper semantic and syntactic features. One such example, explored in this article, is the mention detection and recognition task in the Automatic Content Extraction project, with the goal of identifying named, nominal or pronominal references to real-world entities—mentions—and labeling them with three types of information: entity type, entity subtype and mention type. In this article, we investigate three methods of assigning these related tags and compare them on several data sets. A system based on the methods presented in this article participated and ranked very competitively in the ACE'04 evaluation.

1 Introduction

Information extraction is a crucial step toward understanding and processing natural language data, its goal being to identify and categorize important information conveyed in a discourse. Examples of information extraction tasks are identification of the actors and the objects in written text, the detection and classification of the relations among them, and the events they participate in. These tasks have applications in, among other fields, summarization, information retrieval, data mining, question answering, and language understanding.

One of the basic tasks of information extraction is the *mention detection* task. This task is very similar to *named entity recognition* (NER), as the objects of interest represent very similar concepts. The main difference is that the latter will identify, however, only *named* references, while mention detection seeks named, nominal and pronominal references. In this paper, we will call the identified references *mentions* – using the ACE (NIST, 2003) nomenclature – to differentiate them from *entities* which are the real-world objects (the actual person, location, etc) to which the mentions are referring to[1].

Historically, the goal of the NER task was to find named references to entities and quantity references – time, money (MUC-6, 1995; MUC-7, 1997). In recent years, Automatic Content Extraction evaluation (NIST, 2003; NIST, 2004) expanded the task to also identify nominal and pronominal references, and to group the mentions into sets referring to the same entity, making the task more complicated, as it requires a co-reference module. The set of identified properties has also been extended to include the *mention type* of a reference (whether it is named, nominal or pronominal), its *subtype* (a more specific type dependent on the main entity type), and its *genericity* (whether the entity points to a specific entity, or a generic one[2]), besides the customary main entity type. To our knowledge, little research has been done in the natural language processing context or otherwise on investigating the specific problem of how such multiple labels are best assigned. This article compares three methods for such an assignment.

The simplest model which can be considered for the task is to create an atomic tag by "gluing" together the sub-task labels and considering the new label atomic. This method transforms the problem into a regular sequence classification task, similar to part-of-speech tagging, text chunking, and named entity recognition tasks. We call this model the *all-in-one* model. The immediate drawback of this model is that it creates a large classification space (the cross-product of the sub-task classification spaces) and that, during decoding, partially similar classifications will compete instead of cooperate - more details are presented in Section 3.1. Despite (or maybe due to) its relative simplicity, this model obtained good results in several instances in the past, for POS tagging in morphologically rich languages (Hajic and Hladká, 1998)

[1] In a pragmatic sense, entities are sets of mentions which co-refer.

[2] This last attribute, genericity, depends only loosely on local context. As such, it should be assigned while examining all mentions in an entity, and for this reason is beyond the scope of this article.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 473–480,
Sydney, July 2006.

and mention detection (Jing et al., 2003; Florian et al., 2004).

At the opposite end of classification methodology space, one can use a *cascade* model, which performs the sub-tasks sequentially in a predefined order. Under such a model, described in Section 3.3, the user will build separate models for each subtask. For instance, it could first identify the mention boundaries, then assign the entity type, subtype, and mention level information. Such a model has the immediate advantage of having smaller classification spaces, with the drawback that it requires a specific model invocation path.

In between the two extremes, one can use a *joint* model, which models the classification space in the same way as the all-in-one model, but where the classifications are not atomic. This system incorporates information about sub-model parts, such as whether the current word starts an entity (of any type), or whether the word is part of a nominal mention.

The paper presents a novel contrastive analysis of these three models, comparing them on several datasets in three languages selected from the ACE 2003 and 2004 evaluations. The methods described here are independent of the underlying classifiers, and can be used with any sequence classifiers. All experiments in this article use our in-house implementation of a maximum entropy classifier (Florian et al., 2004), which we selected because of its flexibility of integrating arbitrary types of features. While we agree that the particular choice of classifier will undoubtedly introduce some classifier bias, we want to point out that the described procedures have more to do with the organization of the search space, and will have an impact, one way or another, on most sequence classifiers, including conditional random field classifiers.[3]

The paper is organized as follows: Section 2 describes the multi-task classification problem and prior work, Section 3.3 presents and contrasts the three meta-classification models. Section 4 outlines the experimental setup and the obtained results, and Section 5 concludes the paper.

2 Multi-Task Classification

Many tasks in Natural Language Processing involve labeling a word or sequence of words with a specific property; classic examples are part-of-speech tagging, text chunking, word sense disambiguation and sentiment classification. Most of the time, the word labels are atomic labels, containing a very specific piece of information (e.g. the word is noun plural, or starts a noun phrase, etc). There are cases, though, where the labels consist of several related, but not entirely correlated, properties; examples include mention detection—the task we are interested in—, syntactic parsing with functional tag assignment (besides identifying the syntactic parse, also label the constituent nodes with their functional category, as defined in the Penn Treebank (Marcus et al., 1993)), and, to a lesser extent, part-of-speech tagging in highly inflected languages.[4]

The particular type of mention detection that we are examining in this paper follows the ACE general definition: each mention in the text (a reference to a real-world entity) is assigned three types of information:[5]

- An entity type, describing the type of the entity it points to (e.g. person, location, organization, etc)
- An entity subtype, further detailing the type (e.g. organizations can be commercial, governmental and non-profit, while locations can be a nation, population center, or an international region)
- A mention type, specifying the way the entity is realized – a mention can be named (e.g. *John Smith*), nominal (e.g. *professor*), or pronominal (e.g. *she*).

Such a problem – where the classification consists of several subtasks or attributes – presents additional challenges, when compared to a standard sequence classification task. Specifically, there are inter-dependencies between the subtasks that need to be modeled explicitly; predicting the tags independently of each other will likely result in inconsistent classifications. For instance, in our running example of mention detection, the subtype task is dependent on the entity type; one could not have a *person* with the subtype *non-profit*. On the other hand, the mention type is relatively independent of the entity type and/or subtype: each entity type could be realized under any mention type and vice-versa.

The multi-task classification problem has been subject to investigation in the past. Caruana et al. (1997) analyzed the multi-task learning

[3] While not wishing to delve too deep into the issue of label bias, we would also like to point out (as it was done, for instance, in (Klein, 2003)) that the label bias of MEMM classifiers can be significantly reduced by allowing them to examine the right context of the classification point - as we have done with our model.

[4] The goal there is to also identify word properties such as gender, number, and case (for nouns), mood and tense (for verbs), etc, besides the main POS tag. The task is slightly different, though, as these properties tend to have a stronger dependency on the lexical form of the classified word.

[5] There is a fourth assigned type – a flag specifying whether a mention is specific (i.e. it refers at a clear entity), generic (refers to a generic type, e.g. "the scientists believe .."), unspecified (cannot be determined from the text), or negative (e.g. "no person would do this"). The classification of this type is beyond the goal of this paper.

(MTL) paradigm, where individual related tasks are trained together by sharing a common representation of knowledge, and demonstrated that this strategy yields better results than one-task-at-a-time learning strategy. The authors used a back-propagation neural network, and the paradigm was tested on several machine learning tasks. It also contains an excellent discussion on how and why the MTL paradigm is superior to single-task learning. Florian and Ngai (2001) used the same multi-task learning strategy with a transformation-based learner to show that usually disjointly handled tasks perform slightly better under a joint model; the experiments there were run on POS tagging and text chunking, Chinese word segmentation and POS tagging. Sutton et al. (2004) investigated the multitask classification problem and used a dynamic conditional random fields method, a generalization of linear-chain conditional random fields, which can be viewed as a probabilistic generalization of cascaded, weighted finite-state transducers. The subtasks were represented in a single graphical model that explicitly modeled the sub-task dependence and the uncertainty between them. The system, evaluated on POS tagging and base-noun phrase segmentation, improved on the sequential learning strategy.

In a similar spirit to the approach presented in this article, Florian (2002) considers the task of named entity recognition as a two-step process: the first is the identification of mention boundaries and the second is the classification of the identified chunks, therefore considering a label for each word being formed from two sub-labels: one that specifies the position of the current word relative in a mention (outside any mentions, starts a mention, is inside a mention) and a label specifying the mention type . Experiments on the CoNLL'02 data show that the two-process model yields considerably higher performance.

Hacioglu et al. (2005) explore the same task, investigating the performance of the AIO and the cascade model, and find that the two models have similar performance, with the AIO model having a slight advantage. We expand their study by adding the hybrid *joint* model to the mix, and further investigate different scenarios, showing that the cascade model leads to superior performance most of the time, with a few ties, and show that the cascade model is especially beneficial in cases where *partially-labeled* data (only some of the component labels are given) is available. It turns out though, (Hacioglu, 2005) that the cascade model in (Hacioglu et al., 2005) did not change to a "mention view" sequence classification[6] (as we did in Section 3.3) in the tasks following the entity detection, to allow the system to use longer range features.

[6]As opposed to a "word view".

3 Classification Models

This section presents the three multi-task classification models, which we will experimentally contrast in Section 4. We are interested in performing sequence classification (e.g. assigning a label to each word in a sentence, otherwise known as tagging). Let $\mathcal{X}$ denote the space of sequence elements (words) and $\mathcal{Y}$ denote the space of classifications (labels), both of them being finite spaces. Our goal is to build a classifier

$$h : \mathcal{X}^+ \rightarrow \mathcal{Y}^+$$

which has the property that $|h(\bar{x})| = |\bar{x}|, \forall \bar{x} \in \mathcal{X}^+$ (i.e. the size of the input sequence is preserved). This classifier will select the a posteriori most likely label sequence $\bar{y} = \arg\max_{\bar{y}'} p(\bar{y}'|\bar{x})$; in our case $p(\bar{y}|\bar{x})$ is computed through the standard Markov assumption:

$$p(y_{1,m}|\ \bar{x}) = \prod_i p(y_i|\bar{x}, y_{i-n+1,i-1}) \qquad (1)$$

where $y_{i,j}$ denotes the sequence of labels $y_i..y_j$. Furthermore, we will assume that each label y is composed of a number of sub-labels $y = \left(y^1 y^2 \ldots y^k\right)$[7]; in other words, we will assume the factorization of the label space into k subspaces $\mathcal{Y} = \mathcal{Y}^1 \times \mathcal{Y}^2 \times \ldots \times \mathcal{Y}^k$.

The classifier we used in the experimental section is a maximum entropy classifier (similar to (McCallum et al., 2000))—which can integrate several sources of information in a rigorous manner. It is our empirical observation that, from a performance point of view, being able to use a diverse and abundant feature set is more important than classifier choice, and the maximum entropy framework provides such a utility.

3.1 The All-In-One Model

As the simplest model among those presented here, the all-in-one model ignores the natural factorization of the output space and considers all labels as atomic, and then performs regular sequence classification. One way to look at this process is the following: the classification space $\mathcal{Y} = \mathcal{Y}^1 \times \mathcal{Y}^2 \times \ldots \times \mathcal{Y}^k$ is first mapped onto a same-dimensional space $\mathcal{Z}$ through a one-to-one mapping $o : \mathcal{Y} \rightarrow \mathcal{Z}$; then the features of the system are defined on the space $\mathcal{X}^+ \times \mathcal{Z}$, instead of $\mathcal{X}^+ \times \mathcal{Y}$.

While having the advantage of being simple, it suffers from some theoretical disadvantages:

- The classification space can be very large, being the product of the dimensions of sub-task spaces. In the case of the 2004 ACE data there are 7 entity types, 4 mention types and many subtypes; the observed number of actual

[7]We can assume, without any loss of generality, that all labels have the same number of sub-labels.

All-In-One Model	Joint Model
B-PER B-LOC B-ORG B-MISC	B-

Table 1: Features predicting start of an entity in the *all-in-one* and *joint* models

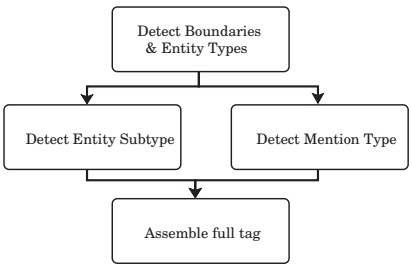

Figure 1: Cascade flow example for mention detection.

sub-label combinations on the training data is 401. Since the dynamic programing (Viterbi) search's runtime dependency on the classification space is $O\left(|\mathcal{Z}|^n\right)$ (n is the Markov dependency size), using larger spaces will negatively impact the decoding run time.[8]

- The probabilities $p\left(z_i|\bar{x}, z_{i-n,i-1}\right)$ require large data sets to be computed properly. If the training data is limited, the probabilities might be poorly estimated.
- The model is not friendly to partial evaluation or weighted sub-task evaluation: different, but partially similar, labels will compete against each other (because the system will return a probability distribution over the classification space), sometimes resulting in wrong partial classification.[9]
- The model cannot *directly* use data that is only partially labeled (i.e. not all sub-labels are specified).

Despite the above disadvantages, this model has performed well in practice: Hajic and Hladká (1998) applied it successfully to find POS sequences for Czech and Florian et al. (2004) reports good results on the 2003 ACE task. Most systems that participated in the CoNLL 2002 and 2003 shared tasks on named entity recognition (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003) applied this model, as they modeled the identification of mention boundaries and the assignment of mention type at the same time.

3.2 The Joint Model

The joint model differs from the all-in-one model in the fact that the labels are no longer atomic: the features of the system can inspect the constituent sub-labels. This change helps alleviate the data sparsity encountered by the previous model by allowing sub-label modeling. The joint model theoretically compares favorably with the all-in-one model:

- The probabilities $p\left(y_i|\bar{x}, y_{i-n,i-1}\right) = p\left(\left(y_i^1, \ldots, y_i^k\right)|\bar{x}, \left(y_{i-n,i-1}^j\right)_{j=1,k}\right)$ might require less training data to be properly estimated, as different sub-labels can be modeled separately.
- The joint model can use features that predict just one or a subset of the sub-labels. Table 1 presents the set of basic features that predict the start of a mention for the CoNLL shared tasks for the two models. While the joint model can encode the start of a mention in one feature, the all-in-one model needs to use four features, resulting in fewer counts per feature and, therefore, yielding less reliably estimated features (or, conversely, it needs more data for the same estimation confidence).
- The model can predict some of the sub-tags ahead of the others (i.e. create a dependency structure on the sub-labels). The model used in the experimental section predicts the sub-labels by using only sub-labels for the previous words, though.
- It is possible, though computationally expensive, for the model to use additional data that is only partially labeled, with the model change presented later in Section 3.4.

3.3 The Cascade Model

For some tasks, there might already exist a natural hierarchy among the sub-labels: some sub-labels could benefit from knowing the value of other, primitive, sub-labels. For example,

- For mention detection, identifying the mention boundaries can be considered as a primitive task. Then, knowing the mention boundaries, one can assign an entity type, subtype, and mention type to each mention.
- In the case of parsing with functional tags, one can perform syntactic parsing, then assign the functional tags to the internal constituents.

[8] From a practical point of view, it might not be very important, as the search is pruned in most cases to only a few hypotheses (beam-search); in our case, pruning the beam only introduced an insignificant model search error (0.1 F-measure).

[9] To exemplify, consider that the system outputs the following classifications and probabilities: O (0.2), B-PER-NAM (0.15), B-PER-NOM (0.15); even the latter 2 suggest that the word is the start of a person mention, the O label will win because the two labels competed against each other.

Words	Since	**Donna**	**Karan**	**International**	went	public	in	1996	...
Labels	O	B-ORG	I-ORG	I-ORG	O	O	O	O	...

Figure 2: Sequence tagging for mention detection: the case for a cascade model.

- For POS tagging, one can detect the main POS first, then detect the other specific properties, making use of the fact that one knows the main tag.

The cascade model is essentially a factorization of individual classifiers for the sub-tasks; in this framework, we will assume that there is a more or less natural dependency structure among subtasks, and that models for each of the subtasks will be built and applied in the order defined by the dependency structure. For example, as shown in Figure 1, one can detect mention boundaries and entity type (at the same time), then detect mention type and subtype in "parallel" (i.e. no dependency exists between these last 2 sub-tags).

A very important advantage of the cascade model is apparent in classification cases where identifying chunks is involved (as is the case with mention detection), similar to advantages that rescoring hypotheses models have: in the second stage, the chunk classification stage, it can switch to a *mention view*, where the classification units are entire mentions and words outside of mentions. This allows the system to make use of aggregate features over the mention words (e.g. all the words are capitalized), and to also effectively use a larger Markov window (instead of 2-3 words, it will use 2-3 chunks/words around the word of interest). Figure 2 contains an example of such a case: the cascade model will have to predict the type of the entire phrase *Donna Karan International*, in the context '*Since <chunk> went public in ..*', which will give it a better opportunity to classify it as an organization. In contrast, because the joint model and AIO have a *word view* of the sentence, will lack the benefit of examining the larger region, and will not have access at features that involve partial future classifications (such as the fact that another mention of a particular type follows).

Compared with the other two models, this classification method has the following advantages:

- The classification spaces for each subtask are considerably smaller; this fact enables the creation of better estimated models
- The problem of partially-agreeing competing labels is completely eliminated
- One can easily use different/additional data to train any of the sub-task models.

3.4 Adding Partially Labeled Data

Annotated data can be sometimes expensive to come by, especially if the label set is complex. But not all sub-tasks were created equal: some of them might be easier to predict than others and, therefore, require less data to train effectively in a cascade setup. Additionally, in realistic situations, some sub-tasks might be considered to have more informational content than others, and have precedence in evaluation. In such a scenario, one might decide to invest resources in annotating additional data only for the particularly interesting sub-task, which could reduce this effort significantly.

To test this hypothesis, we annotated additional data with the entity type only. The cascade model can incorporate this data easily: it just adds it to the training data for the entity type classifier model. While it is not immediately apparent how to incorporate this new data into the all-in-one and joint models, in order to maintain fairness in comparing the models, we modified the procedures to allow for the inclusion. Let $\mathcal{T}$ denote the original training data, and $\mathcal{T}'$ denote the additional training data.

For the all-in-one model, the additional training data cannot be incorporated directly; this is an inherent deficiency of the AIO model. To facilitate a fair comparison, we will incorporate it in an indirect way: we train a classifier C on the additional training data $\mathcal{T}'$, which we then use to classify the original training data $\mathcal{T}$. Then we train the all-in-one classifier on the original training data $\mathcal{T}$, adding the features defined on the output of applying the classifier C on $\mathcal{T}$.

The situation is better for the joint model: the new training data $\mathcal{T}'$ *can* be incorporated directly into the training data $\mathcal{T}$.[10] The maximum entropy model estimates the model parameters by maximizing the data log-likelihood

$$L = \sum_{(x,y)} \hat{p}\,(x,y) \log q_\lambda\,(y|x)$$

where $\hat{p}\,(x,y)$ is the observed probability distribution of the pair (x,y) and $q_\lambda\,(y|x) = \frac{1}{Z}\prod_j \exp\left(\lambda_j \cdot f_j\,(x,y)\right)$ is the conditional ME probability distribution as computed by the model. In the case where some of the data is partially annotated, the log-likelihood becomes

$$L = \sum_{(x,y)\in\mathcal{T}\cup\mathcal{T}'} \hat{p}\,(x,y) \log q_\lambda\,(y|x)$$

[10]The solution we present here is particular for MEMM models (though similar solutions may exist for other models as well). We also assume the reader is familiar with the normal MaxEnt training procedure; we present here only the differences to the standard algorithm. See (Manning and Schütze, 1999) for a good description.

$$= \sum_{(x,y)\in\mathcal{T}} \hat{p}(x,y) \log q_\lambda(y|x) + \sum_{(x,y)\in\mathcal{T}'} \hat{p}(x,y) \log q_\lambda(y|x) \quad (2)$$

The only technical problem that we are faced with here is that we cannot directly estimate the observed probability $\hat{p}(x,y)$ for examples in $\mathcal{T}'$, since they are only partially labeled. Borrowing the idea from the expectation-maximization algorithm (Dempster et al., 1977), we can replace this probability by the re-normalized system proposed probability: for $(x, y_x) \in \mathcal{T}'$, we define

$$\hat{q}(x,y) = \hat{p}(x)\underbrace{\delta(y \in y_x)\frac{q_\lambda(y|x)}{\sum_{y'\in y_x} q_\lambda(y'|x)}}_{=\hat{q}_\lambda(y|x)}$$

where y_x is the subset of labels from $\mathcal{Y}$ which are consistent with the partial classification of x in $\mathcal{T}'$. $\delta(y \in y_x)$ is 1 if and only if y is consistent with the partial classification y_x.[11] The log-likelihood computation in Equation (2) becomes

$$L = \sum_{(x,y)\in\mathcal{T}} \hat{p}(x,y) \log q_\lambda(y|x) + \sum_{(x,y)\in\mathcal{T}'} \hat{q}(x,y) \log q_\lambda(y|x)$$

To further simplify the evaluation, the quantities $\hat{q}(x,y)$ are recomputed every few steps, and are considered constant as far as finding the optimum λ values is concerned (the partial derivative computations and numerical updates otherwise become quite complicated, and the solution is no longer unique). Given this new evaluation function, the training algorithm will proceed exactly the same way as in the normal case where all the data is fully labeled.

4 Experiments

All the experiments in this section are run on the ACE 2003 and 2004 data sets, in all the three languages covered: Arabic, Chinese, and English. Since the evaluation test set is not publicly available, we have split the publicly available data into a 80%/20% data split. To facilitate future comparisons with work presented here, and to simulate a realistic scenario, the splits are created based on article dates: the test data is selected as the last 20% of the data in chronological order. This way, the documents in the training and test data sets do not overlap in time, and the ones in the test data are posterior to the ones in the training data. Table 2 presents the number of documents in the training/test datasets for the three languages.

[11] For instance, the full label *B-PER* is consistent with the partial label *B*, but not with *O* or *I*.

Language	Training	Test
Arabic	511	178
Chinese	480	166
English 2003	658	139
English 2004	337	114

Table 2: Datasets size (number of documents)

Each word in the training data is labeled with one of the following properties:[12]

- if it is not part of any entity, it's labeled as *O*
- if it is part of an entity, it contains a tag specifying whether it starts a mention (*B*-) or is inside a mention (*I*-). It is also labeled with the entity type of the mention (seven possible types: person, organization, location, facility, geo-political entity, weapon, and vehicle), the mention type (named, nominal, pronominal, or premodifier[13]), and the entity subtype (depends on the main entity type).

The underlying classifier used to run the experiments in this article is a maximum entropy model with a Gaussian prior (Chen and Rosenfeld, 1999), making use of a large range of features, including lexical (words and morphs in a 3-word window, prefixes and suffixes of length up to 4, WordNet (Miller, 1995) for English), syntactic (POS tags, text chunks), gazetteers, and the output of other information extraction models. These features were described in (Florian et al., 2004), and are not discussed here. All three methods (AIO, joint, and cascade) instantiate classifiers based on the same feature types whenever possible. In terms of language-specific processing, the Arabic system uses as input morphological segments, while the Chinese system is a character-based model (the input elements $x \in \mathcal{X}$ are characters), but it has access to word segments as features.

Performance in the ACE task is officially evaluated using a special-purpose measure, the ACE value metric (NIST, 2003; NIST, 2004). This metric assigns a score based on the similarity between the system's output and the gold-standard at both mention and entity level, and assigns different weights to different entity types (e.g. the person entity weights considerably more than a facility entity, at least in the 2003 and 2004 evaluations). Since this article focuses on the mention detection task, we decided to use the more intuitive (unweighted) F-measure: the harmonic mean of precision and recall.

[12] The mention encoding is the IOB2 encoding presented in (Tjong Kim Sang and Veenstra, 1999) and introduced by (Ramshaw and Marcus, 1994) for the task of base noun phrase chunking.

[13] This is a special class, used for mentions that modify other labeled mentions; e.g. French in *"French* wine". This tag is specific only to ACE'04.

For the cascade model, the sub-task flow is presented in Figure 1. In the first step, we identify the mention boundaries together with their entity type (e.g. person, organization, etc). In preliminary experiments, we tried to "cascade" this task. The performance was similar on both strategies; the separated model would yield higher recall at the expense of precision, while the combined model would have higher precision, but lower recall. We decided to use in the system with higher precision. Once the mentions are identified and classified with the entity type property, the data is passed, in parallel, to the mention type detector and the subtype detector.

For English and Arabic, we spent three person-weeks to annotate additional data labeled with only the entity type information: 550k words for English and 200k words for Arabic. As mentioned earlier, adding this data to the cascade model is a trivial task: the data just gets added to the training data, and the model is retrained. For the AIO model, we have build another mention classifier on the additional training data, and labeled the original ACE training data with it. It is important to note here that the ACE training data (called $\mathcal{T}$ in Section 3.4) is consistent with the additional training data $\mathcal{T}'$: the annotation guidelines for $\mathcal{T}'$ are the same as for the original ACE data, but we only labeled entity type information. The resulting classifications are then used as features in the final AIO classifier. The joint model uses the additional partially-labeled data in the way described in Section 3.4; the probabilities $\hat{q}(x, y)$ are updated every 5 iterations.

Table 3 presents the results: overall, the cascade model performs significantly better than the all-in-one model in four out the six tested cases - the numbers presented in bold reflect that the difference in performance to the AIO model is statistically significant.[14] The joint model, while managing to recover some ground, falls in between the AIO and the cascade models.

When additional partially-labeled data was available, the cascade and joint models receive a statistically significant boost in performance, while the all-in-one model's performance barely changes. This fact can be explained by the fact that the entity type-only model is in itself errorful; measuring the performance of the model on the training data yields a performance of 82 F-measure;[15] therefore the AIO model will only access partially-correct data, and is unable to make effective use of it. In contrast, the training data for the entity type in the cascade model effectively triples, and this change is reflected positively in the 1.5 increase in F-measure.

Language	Data$^+$	A-I-O	Joint	Cascade
Arabic'04	no	59.2	59.1	**59.7**
	yes	59.4	**60.0**	**60.7**
English'04	no	72.1	72.3	**73.7**
	yes	72.5	**74.1**	**75.2**
Chinese'04	no	71.2	71.7	71.7
English '03	no	79.5	79.5	79.7

Table 3: Experimental results: F-measure on the full label

Language	Data$^+$	A-I-O	Joint	Cascade
Arabic'04	no	66.3	66.5	**67.5**
	yes	66.4	**67.9**	**68.9**
English'04	no	77.9	78.1	**79.2**
	yes	78.3	**80.5**	**82.6**
Chinese'04	no	75.4	76.1	**76.8**
English '03	no	80.4	80.4	81.1

Table 4: F-measure results on entity type only

Not all properties are equally valuable: the entity type is arguably more interesting than the other properties. If we restrict ourselves to evaluating the entity type output only (by projecting the output label to the entity type only), the difference in performance between the all-in-one model and cascade is even more pronounced, as shown in Table 4. The cascade model outperforms here both the all-in-one and joint models in all cases except English'03, where the difference is not statistically significant.

As far as run-time speed is concerned, the AIO and cascade models behave similarly: our implementation tags approximately 500 tokens per second (averaged over the three languages, on a Pentium 3, 1.2Ghz, 2Gb of memory). Since a MaxEnt implementation is mostly dependent on the number of features that fire on average on a example, and not on the total number of features, the joint model runs twice as slow: the average number of features firing on a particular example is considerably higher. On average, the joint system can tag approximately 240 words per second. The train time is also considerably longer; it takes 15 times as long to train the joint model as it takes to train the all-in-one model (60 mins/iteration compared to 4 mins/iteration); the cascade model trains faster than the AIO model.

One last important fact that is worth mentioning is that a system based on the cascade model participated in the ACE'04 competition, yielding very competitive results in all three languages.

[14] To assert the statistical significance of the results, we ran a paired Wilcoxon test over the series obtained by computing F-measure on each document in the test set. The results are significant at a level of at least 0.009.

[15] Since the additional training data is consistent in the labeling of the entity type, such a comparison is indeed possible. The above mentioned score is on entity types only.

5 Conclusion

As natural language processing becomes more sophisticated and powerful, we start focus our attention on more and more properties associated with the objects we are seeking, as they allow for a deeper and more complex representation of the real world. With this focus comes the question of how this goal should be accomplished – either detect all properties at once, one at a time through a pipeline, or a hybrid model. This paper presents three methods through which multi-label sequence classification can be achieved, and evaluates and contrasts them on the Automatic Content Extraction task. On the ACE mention detection task, the cascade model which predicts first the mention boundaries and entity types, followed by mention type and entity subtype outperforms the simple all-in-one model in most cases, and the joint model in a few cases.

Among the proposed models, the cascade approach has the definite advantage that it can easily and productively incorporate additional partially-labeled data. We also presented a novel modification of the joint system training that allows for the direct incorporation of additional data, which increased the system performance significantly. The all-in-one model can only incorporate additional data in an indirect way, resulting in little to no overall improvement.

Finally, the performance obtained by the cascade model is very competitive: when paired with a coreference module, it ranked very well in the "Entity Detection and Tracking" task in the ACE'04 evaluation.

References

R. Caruana, L. Pratt, and S. Thrun. 1997. Multitask learning. *Machine Learning*, 28:41.

Stanley F. Chen and Ronald Rosenfeld. 1999. A gaussian prior for smoothing maximum entropy models. Technical Report CMU-CS-99-108, Computer Science Department, Carnegie Mellon University.

A. P. Dempster, N. M. Laird, , and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal statistical Society*, 39(1):1–38.

R. Florian and G. Ngai. 2001. Multidimensional transformation-based learning. In *Proceedings of CoNLL'01*, pages 1–8.

R. Florian, H. Hassan, A. Ittycheriah, H. Jing, N. Kambhatla, X. Luo, N Nicolov, and S Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 1–8.

R. Florian. 2002. Named entity recognition as a house of cards: Classifier stacking. In *Proceedings of CoNLL-2002*, pages 175–178.

Kadri Hacioglu, Benjamin Douglas, and Ying Chen. 2005. Detection of entity mentions occuring in english and chinese text. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 379–386, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Kadri Hacioglu. 2005. Private communication.

J. Hajic and Hladká. 1998. Tagging inflective languages: Prediction of morphological categories for a rich, structured tagset. In *Proceedings of the 36th Annual Meeting of the ACL and the 17th ICCL*, pages 483–490, Montréal, Canada.

H. Jing, R. Florian, X. Luo, T. Zhang, and A. Ittycheriah. 2003. HowtogetaChineseName(Entity): Segmentation and combination issues. In *Proceedings of EMNLP'03*, pages 200–207.

Dan Klein. 2003. Maxent models, conditional estimation, and optimization, without the magic. Tutorial presented at NAACL-03 and ACL-03.

C. D. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330.

Andrew McCallum, Dayne Freitag, and Fernando Pereira. 2000. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of ICML-2000*.

G. A. Miller. 1995. WordNet: A lexical database. *Communications of the ACM*, 38(11).

MUC-6. 1995. The sixth message understanding conference. www.cs.nyu.edu/cs/faculty/grishman/muc6.html.

MUC-7. 1997. The seventh message understanding conference. www.itl.nist.gov/iad/894.02/related_projects/muc/proceedings/muc_7_toc.html.

NIST. 2003. The ACE evaluation plan. www.nist.gov/speech/tests/ace/index.htm.

NIST. 2004. The ACE evaluation plan. www.nist.gov/speech/tests/ace/index.htm.

L. Ramshaw and M. Marcus. 1994. Exploring the statistical derivation of transformational rule sequences for part-of-speech tagging. In *Proceedings of the ACL Workshop on Combining Symbolic and Statistical Approaches to Language*, pages 128–135.

C. Sutton, K. Rohanimanesh, and A. McCallum. 2004. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *In Proceedings of the Twenty-First International Conference on Machine Learning (ICML-2004)*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.

E. F. Tjong Kim Sang and J. Veenstra. 1999. Representing text chunks. In *Proceedings of EACL'99*.

E. F. Tjong Kim Sang. 2002. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2002*, pages 155–158.