The UPC System for Arabic-to-English Entity Translation

D. Farwell, J. Giménez, E. Gonzàlez, R. Halkoum, H. Rodríguez, and **M. Surdeanu**
TALP Research Center, LSI Department
Universitat Politècnica de Catalunya
Jordi Girona Salgado 1–3, E-08034, Barcelona
{farwell, jgimenez, egonzalez,
halkoum, horacio, surdeanu}@lsi.upc.edu

1 Introduction

We describe the UPC Arabic-to-English Entity Translation System presented at the ACE/ET 2007 Evaluation Campaign, and its application to the Arabic-to-English Entity Translation task. Our approach to Entity Translation (ET) is fairly simple. We have divided the task into three separate subtasks: Named Entity Recognition and Classification (NERC), Coreference Resolution (CR), and Machine Translation (MT), which are approached independently.

Because named entitites are not always unequivocally translated, we deal with Entity Translation essentially as a disambiguation problem. Entities are disambiguated by means of a state-of-the-art Statistical Machine Translation (SMT) system. This allows us to robustly translate entities in the context of the whole sentence.

The paper is organized as follows. The System architecture is deeply described in Section 2. This includes a detailed description of every component and its isolated performance, as well as the list of resources (i.e., tools and corpora) utilized for their construction. We analyze the overall system performance in Section 3. We examine the main merits and defficiencies of our system through a rigurous and illustrative error analysis. Finally, in Section 4 main conclusions are drawn and further work is outlined.

2 System Architecture

The System architecture is depicted in Figure 1. Our System receives as input a list of properly formatted ACE '.sgml' files. Prior to entering the system core, these files are linguistically enriched at the level of shallow syntax, as described in Section 2.4.3. Then, all documents are separatedly processed by the NERC, CR, and MT subsystems. Sentences are translated, named entities are recognized and classified, and coreference chains are detected, without any kind of interaction between these modules. Entity translations are obtained by properly combining their outputs.

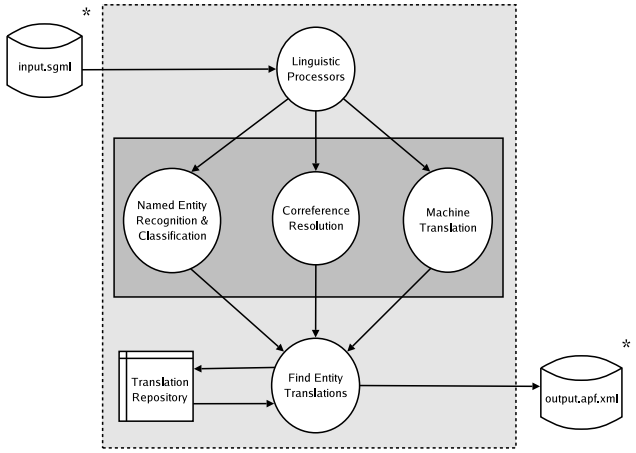

Figure 1: System Architecture

During translation we keep track of the correspondence between source (input) words and target (output) words. This will allow us to provide a translation for almost every named entity recognized in the source sentence. However, there are two exceptional situations:

Untranslated Entities In some cases it is not possible to generate a complete translation for a

named entity. This may happen either because the named entity contains words which are unknown to the MT system, or because the MT system does not know how to correctly handle the translation of the named entity in the context of the source sentence and the target sentence under construction. In these cases we perform a lookup in the *'Translation Repository'*. This repository contains all entities translated so far over all documents. If no candidate translation is found we inspect the bilingual gazeteers described in Section 2.4.2. If still a translation can not be found we simply output the incomplete translation, which contains untranslated Arabic words.

Phrase Boundaries Because the MT system is not word-based but phrase-based, it may happen that a named entity starts or ends in the middle of a phrase. In this case, we simply output the whole translation embedding the actual translation.

Finally, we use the information provided by the CR subsystem to group together the different mentions of every entity in each document.

Below, we provide a description of every component, as well as the tools and data utilized for its construction. The isolated performance of every component is evaluated.

2.1 Named Entity Recognition and Classification

2.1.1 Approach

We use the Arabic NERC system for several purposes: (a) identification of the boundaries of the entities to be translated, (b) detection of the entity types and subtypes (e.g., `Airport` is one of the subtypes of the type `FACILITY`), and (c) identification of the type of the entity mention (e.g., nominal or pronominal). We model all these operations jointly using a sequential tagger that assigns a Begin/Inside/Outside (BIO) label to each of the tokens in the document word sequences. The BIO labels are extended with a concatenation of the entity features to be learned. For example, the label `B-FAC-Plant-NOM` indicates that the corresponding token begins a nominal mention of an entity of type `FACILITY` and subtype `Plant`.

Input: A training sample $\mathbf{z} = (\mathbf{x}, \mathbf{y}) \in (\mathcal{X} \times \mathcal{Y})^m$
Input: Number of epochs T

$\mathbf{M} = \mathbf{0}$ ($\mathbf{M} \in I\!R^{|\mathcal{Y}| \times |\mathcal{X}|}$)
for $t = 1$ **to** T **do**
  for $i = 1$ **to** m **do**
    predict $\hat{y}_i = \arg\max_{r=1}^{|\mathcal{Y}|} \{\langle \mathbf{M}_r, \mathbf{x}_i \rangle\}$
    set $E = \{r \neq y_i : \langle \mathbf{M}_r, \mathbf{x}_i \rangle \geq \langle \mathbf{M}_{y_i}, \mathbf{x}_i \rangle\}$
    if $E \neq \emptyset$ **then then**
      for all r **in** E **do do**
        $\mathbf{M}_r = \mathbf{M}_r - \mathbf{x}_i / |E|$
      end for
      $\mathbf{M}_{y_i} = \mathbf{M}_{y_i} + \mathbf{x}_i$
    end if
  end for
end for

Output: $H(\mathbf{x}) = \arg\max_r \{\langle \mathbf{M}_r, \mathbf{x} \rangle\}$

Algorithm 1: The Ultraconservative Multiclass Perceptron.

We developed the token-level BIO classifiers using the Ultraconservative Multiclass Perceptron Algorithm (UMPA) (Crammer and Singer, 2003). The UMPA is a variation of the Perceptron Algorithm (PA) (Rosenblatt, 1958). Unlike the original PA, which is a binary classifier that learns a single prediction vector, the UMPA maintains a prediction matrix $\mathbf{M}$ with one row for each class to be modeled. The UMPA is ultraconservative in the sense that it updates $\mathbf{M}$ only on rounds with prediction errors. Furthermore, the algorithm updates only the vector of the incorrectly-predicted class and of the classes that scored higher than the correct class. We summarize the algorithm in Algorithm 1. For each training example $\mathbf{x}_i$, the algorithm constructs the set E of classes that score higher than the correct class y_i. If the set is non-empty, i.e., there was a prediction error, the algorithm adds the vector of the current sample to the vector of the correct class $\mathbf{M}_{y_i}$, and subtracts $\mathbf{x}_i / |E|$ from the vector of each class in E. The UMPA was empirically shown to perform well in several classification problems, even though it makes a smaller number of updates than the regular PA (Crammer and Singer, 2003).

We have extended the original UMPA with averaged predictions (Freund and Shapire, 1999). In other words, in our actual UMPA implementation the predicted label $\hat{y}$ of an unlabeled example $\mathbf{x}$ is calculated as: $\hat{y} = \arg\max_r \{\langle \mathbf{Avg}_r, \mathbf{x} \rangle\}$, where $\mathbf{Avg}_r$ is the sum of all the instances of $\mathbf{M}_r$ seen in training, weighted by the number of learning itera-

tions they survived unchanged.

It is straightforward to extend this algorithm with kernel functions (see (Freund and Shapire, 1999) for the explanation). Nevertheless, in order to minimize training and prediction times, in this work we used only the default algorithm in primal form, as illustrated in Algorithm 1.

The UMPA algorithm gives a prediction for each individual token in the document stream. We select a consistent solution at sentence level using a simple greedy inference strategy: for every token we select the label with the highest score that is consistent with the previously-assigned label. For example, an `I` label cannot follow an `O` label.

2.1.2 Features

For feature generation we explored four models that include lexical, morphological, syntactic, and semantic attributes. We detail these models next.

Model $\mathbf{M}_1$ - contains the following lexical attributes:

- The token lexem.
- The suffixes and prefixes of length 2, 3, and 4.
- The sequence obtained by removing all letters from the token.
- The sequence obtained by removing all alphanumeric characters from the token.
- *isAllDigits* - Boolean flag set to true if the word contains only digits.
- *isAllDigitsOrDots* - Boolean flag set to true if the word contains only digits or dots.

Model $\mathbf{M}_2$ - adds part of speech (POS) attributes.

Model $\mathbf{M}_3$ - adds syntactic chunk labels.

Model $\mathbf{M}_4$ - adds the following class and gazetteer-based attributes:

- *isNumber* - Boolean flag set to true if the token is a word-spelled number (e.g., سبعة, *sbEp*, Seven).
- *isMultiplier* - Boolean flag set to true if the token is a multiplier typically used to compose numbers (e.g., الف, >*lf*, Thousand).
- *isDay* - Boolean flag set to true if the token is the name of a day of the week (e.g., السبت, *Alsbt*, Saturday).
- *isMonth* - Boolean flag set to true if the token is the name of a month (e.g., مارس, *mArs*, March).
- *isPersonTrigger* - BIO flag that indicates if the token begins or is inside a sequence of words that typically precedes a person entity (e.g, السفير, *ALsfyr*, Ambassador).
- *knownPerson* - BIO flag that indicates if the token is part of a sequence that is an entry in the person name gazetteer.

In addition to the features introduced above we generate a set of context-based features for each token: (a) a set of static-context features based on the features of the tokens preceding/following the current token, and (b) a set of dynamic-context features based on the NERC BIO labels of the previous tokens. For the dynamic-context features we used the gold labels during training and the predicted labels during actual tagging.

2.1.3 Experimental Setting

We trained the Arabic NERC on a corpus that concatenates the ACE Arabic training data from 2005 and 2007. We tuned our system's parameters on the ACE 2007 development corpus. After eliminating documents from the training corpus that appear in the development corpus as well, the training corpus contains 780 documents. The development corpus contains 19 documents.

Our initial goal was to develop a single NERC system that would jointly recognize and classify all the needed entity attributes: entity types and subtypes and entity mention types. Unfortunately, such a system generates 223 individual classes and does not fit in the memory of our development machines. Given this limitation, we decided to build two separate taggers: the first recognizes the entity types and subtypes while the second identifies the types of the entity mentions. The first tagger works with 89 BIO classes and the second with 7. Both taggers use the feature sets described in the previous subsection. Both taggers were trained on the heads of the entity mentions. We currently do not use the entity extents.

2.1.4 Performance

Table 1 shows the performance of the tagger for the detection and classification of entity types and

Model	P	R	F_1	Best epoch
M1	76.54%	75.27%	75.90	15
M2	76.43%	77.32%	76.87	18
M3	77.51%	**77.81%**	**77.66**	19
M4	**79.91%**	70.38%	74.84	29

Table 1: NERC results on the development set for the entity type/subtype problem.

Model	P	R	F_1	Best epoch
M1	78.25%	78.79%	78.52	31
M2	78.54%	**79.77%**	**79.15**	35
M3	78.30%	79.37%	78.83	35
M4	**80.20%**	69.70%	74.58	35

Table 2: NERC results on the development set for the entity mention type problem.

subtypes, for the four feature models. For this analysis we used a strict scorer that considers an entity as correct only if both its boundaries and its class match the key. The table indicates that, as expected, morphological and syntactic features help, but surprisingly, class and gazetteer features do not. We do not have a complete understanding of why this happens, but we believe that our tagger cannot generalize well when an ambiguous gazetteer is used on such a sparse corpus. With respect to the quantitative performance, this tagger spent 175 second/epoch when training on a 3.2GHz Pentium IV machine. During prediction, the tagger labels 1,600 words/second on the same computer.

Table 2 shows the performance of the second tagger, which identifies the types of entity mentions. The results reported in the table support the intuition that lexical and morphological information is sufficient for the identification of the mention types: the best model is model M2 with an F_1 score of 79.15. Because the second tagger has to handle fewer classes (7 versus 89) it trains faster than the first one (54 second/epoch), and labels new text quicker (8100 words/second).

2.2 Coreference Resolution

2.2.1 Approach

We have incorporated a baseline generic pronominal anaphora resolution component in our system. This subsystem uses a machine learning approach to coreference resolution, specifically, the competition learning approach of (Yang et al., 2003) and (Connolly et al., 1997).

The problem of pronominal anaphora resolution can be seen as a classification problem: for each pronoun whose antecedent has to be identified a set of candidates is extracted, the pronoun is paired with each candidate, and for each pair a classifier is used to determine if they corefer, that is, if the candidate is the antecedent of the pronoun. If several candidates are classified as coreferring, most approaches use the confidence of the classifier as the criterium to break ties.

On the contrary, (Yang et al., 2003) and (Connolly et al., 1997) propose a different approach: first candidates are filtered using a single-candidate classifier, and then for all the remaining candidates the system decides which one is the most likely antecedent through a set of pair-wise comparisons. It remains a binary classification problem, but instead of focusing on:

- Is candidate **X** an antecedent of pronoun **P**?

the question is:

- Is candidate **X** a better antecedent of pronoun **P** than candidate **Y**?

Even if this is a general approach, as a first step towards a more complex system our work has only covered the resolution of pronouns. Moreover, only nouns and other pronouns have been considered as antecedent candidates.

We have considered two different algorithms. The first one, closest to the proposal of (Yang et al., 2003), is detailed in Algorithm 2. For each pronoun in the text, the set of candidates is constructed. We have taken as candidates all the pronouns and nouns in a window of the previous sentences, as well as those in the same sentence as the pronoun but appearing before it (we have not considered cataphoric relations). This window size is the same as in (Yang et al., 2003). This candidate set is filtered. If no candidates remain after the filtering phase, the pronoun is considered unsolved. Otherwise, each remaining candidate is compared to the others. Initially all candidates are scored zero, and each time a candidate is judged better than another one its score increases one. After all pairwise comparisons have been performed, the highest scored candidate is selected as

antecedent of the pronoun. If there is more than one such candidates, the usual criterion of selecting the closest to the pronoun is used.

The second algorithm, detailed in Algorithm 3, uses a strategy similar to that of (Connolly et al., 1997). In this case, after the filtering phase the closest remaining candidate is selected as best candidate so far. Then, this best candidate is compared successively to the next candidate, moving from right to left (that is, increasing the distance to the pronoun). If at any step, the next candidate is better that the best candidate so far, it becomes the new best candidate so far, and the process continues. After all candidates have been compared, the best candidate found is selected as antecedent of the pronoun.

The binary classification functions F_1 and F_2 are the same for both approaches. They can be learned with different Machine Learning approaches from an annotated corpus. For each annotated pronoun p in the corpus, several examples can be generated:

- For each candidate c in the sentence window, the tuple (p, c) is a positive example for F_1 if the candidate and the pronoun corefer, and a negative example otherwise.
- For each pair of candidates c_1 and c_2 in the sentence window, where c_1 is closer to the pronoun, the tuple (p, c_1, c_2) is a positive example for F_2 if c_1 corefers with the pronoun and c_2 does not, and a negative example if c_2 corefers with the pronoun and c_1 does not. If both candidates corefer or neither does, no examples for F_2 are generated.

We have chosen to learn these functions using Support Vector Machines (Vapnik, 1995), which are known to give good performance in Natural Language Processing tasks. A polynomial kernel of degree 2 and shallow syntactic features have been used for the learning. For each element in the example tuples (pronouns or candidates), the form, part-of-speech and base phrase chunk tag (as given by ASVM-Tools) of the words in a window around the pronoun or candidate are considered as features. Two more simple but Arabic-specific features have also been considered for all words in this window: whether ASVM-Tools had to change the form to restore the feminine marker (which we have taken as

Input: A text T

for all Pronouns p in T **do**
  Find candidate set C
  Filter candidate set
    $C' = \{c \in C \mid F_1(p, c) > 0\}$
  if C' is empty **then**
    Pronoun p is considered unsolved
  else
    Initialize scores $\forall c \in C'\ score[c] = 0$
    for all Pairs $c_1, c_2 \in C'$ where
      $dist(c_1, p) < dist(c_2, p)$ **do**
      if $F_2(p, c_1, c_2) > 0$ **then**
        Increment $score[c_1]$
      else
        Increment $score[c_2]$
      end if
    end for
    Set $c_a = \arg\max_c score[c]$ as the
      antecedent of p
  end if
end for

Output: The text T with pronouns resolved

Algorithm 2: Round Robin Resolution

Input: A text T

for all Pronouns p in T **do**
  Find candidate set C
  Filter candidate set
    $C' = \{c \in C \mid F_1(p, c) > 0\}$
  if C' is empty **then**
    Pronoun p is considered unsolved
  else
    Set as best candidate c_b the candidate in C' closest to p
    for all Candidates $c \in C'$
      from closest to furthest to p **do**
      if $F_2(p, c_t, c) < 0$ **then**
        Set c as new best candidate c_b
      end if
    end for
    Set the best candidate c_b as the
      antecedent of p
  end if
end for

Output: The text T with pronouns resolved

Algorithm 3: Lineal Resolution

Model		Overall	Evaluable		
		Assignation	Assignation	Precision	Recall
Round Robin	Filter	46%	52%	65%	34%
	No	100%	100%	11%	11%
Lineal	Filter	46%	52%	63%	33%
	No	100%	100%	50%	50%

Table 3: Coreference resolution performance

a simple indicator of genre) and whether the word begins with the determinant ال, *Al*.

The presence of few language-specific features makes our approach almost language independent.

For the training of SVM we have used the freely available package TinySVM[1].

2.2.2 Experimental Setting

To evaluate the performance of the system, we have built a single collection with the Arabic Newswire sections of the ACE 2005 and ACE 2007 Multilingual Training Data. The Entity Mention coreference information has been used. This collection has been randomly split into a training set containing 453 documents, and a test set containing 40. No cross-fold validation has been possible due to the high computational cost of SVM training.

Antecedent candidates have been looked for in the previous two sentences (as in (Yang et al., 2003)), and a window size of 5 words for the features of the words has been used.

In addition to comparing the Round Robin and Lineal algorithms, we have tested versions of these algorithms without the candidate filtering phase. Without filtering, an antecedent will be found for all pronouns, but we expect the precision of the assigned antecedents to decrease.

We will consider the following metrics for evaluation:

Assignation Fraction of pronouns for which an antecedent has been proposed, from the total number of pronouns: $Assignation = \frac{\#Proposed}{\#Total}$.

Precision Fraction of pronouns for which the proposed antecedent is correct, from the pronouns for which an antecedent has been proposed assigned: $Precision = \frac{\#Correct}{\#Proposed}$.

Recall Fraction of pronouns for which the proposed antecedent is correct, from the total number of pronouns: $Recall = \frac{\#Correct}{\#Total}$.

Given that not all pronouns in the ACE Data have been annotated, we will consider in our evaluation only those pronouns which have been tagged as Entity Mentions.

Training	F_1	6h 8min
	F_2	167h 39min

Round Robin	Filter	7min
	No	4h 55min
Lineal	Filter	7min
	No	26min

Table 4: Coreference resolution running time

2.2.3 Performance

Table 3 shows the performance of the four considered algorithms on the test data set, and Table 4 gives the running times for the training of the two functions, as well as for the resolution of the coreference in the test set[2].

As it can be seen in the tables, if filtering is used, there is no significant difference in the behavior of the two algorithms, neither in terms of performance nor in computational cost. The precision is around 65%, low for the state of the art, but this is not surprising, as these are baseline results. Given that an antecedent is proposed for only 52% of the evaluable pronouns, the recall is also low, around 35%.

If filtering is disabled, antecedents are assigned for all pronouns in the data set and, as expected,

[1] `http://chasen.org/~taku/software/TinySVM/`

[2] On a machine equiped with a Pentium 4 3.2GHz

precision decreases. Nevertheless, whereas in the case of the lineal algorithm, precision only lowers to 50% and there is an increase in recall; for round robin precision goes down to 11%, and recall also decreases. We believe the reason for this behavior is that, without filtering, a great number of irrelevant candidates can appear (those which would have been filtered), and some of them can achieve high scores due to comparisons among themselves in the round robin approach. It is hence possible that they are selected as antecedent. In the lineal approach, these irrelevant candidates are more likely to be discarded in the direct comparison against the best candidate so far, and their influence on the overall process is lower.

Moreover, the disabling of filtering produces an increase in the computational cost of the resolution process, specially for the round robin algorithm, which being quadratic multiplies its running time by a factor of more than 40. In the case of the lineal algorithm the factor is of about 4.

However, in spite of its benefits the fact that filtering leaves 54% of the pronouns in the whole data set without candidates indicates that the learner may need more generalization.

Bearing these results in mind, we have chosen to use the round robin algorithm with filtering enabled for the resolution of the pronouns in the final system, as it is the system which is less computationally demanding (its difference with respect to the lineal algorithm is insignificant) and which gives the best results in precision. We consider in our case precision to be more important than recall: we would rather not have an antecedent at all than have a wrong one.

2.3 Machine Translation

2.3.1 Approach

The MT component is a phrase-based SMT system (Koehn et al., 2003), built almost entirely using freely available components.

We use the *SRI Language Modeling Toolkit* (Stolcke, 2002) for language modeling. We build trigram language models applying linear interpolation and Kneser-Ney discounting for smoothing.

Translation models are built on top of word-aligned parallel corpora, as described by Giménez and Màrquez (2005). We used the *GIZA++ SMT Toolkit*[3] (Och and Ney, 2003) to generate word alignments[4]. We apply the phrase-extract algorithm, as described by Och (2002), on the Viterbi alignments output by GIZA++. We work with the union of source-to-target and target-to-source alignments, with no heuristic refinement. Phrases up to length five are considered. Also, phrase pairs appearing only once are discarded, and phrase pairs in which the source/target phrase was more than three times longer than the target/source phrase are ignored. Phrase pairs are scored on the basis of unsmoothed maximum likelihood estimation (MLE).

Regarding the argmax search, we used the *Pharaoh* beam search decoder (Koehn, 2004), which naturally fits with the previous tools. We use M language models, N generative translation models, and N discriminative translation models. Probability models are combined in a log-linear fashion:

$$\begin{aligned} logP(e|f) \propto\ & \lambda_{lm1} logP(e)_1 + ... + \lambda_{lmN} logP(e)_M \\ & + \lambda_{fe1} logP(f|e)_1 + ... + \lambda_{feN} logP(f|e)_N \\ & + \lambda_{ef1} logP(e|f)_1 + ... + \lambda_{efN} logP(e|f)_N \end{aligned}$$

$P(e)_i$ ($i \in [1, M]$) correspond to the language models. $P(f|e)_j$ and $P(e|f)_j$ ($j \in [1, N]$) correspond to the traditional generative and discriminative translation models, estimated on the basis of MLE.

2.3.2 Experimental Setting

We train several translation and language models. Translation models are built on top of parallel texts. We build separate translation models for each of the parallel corpora described in Section 2.4.1:

AE Arabic English Parallel News.

AR Arabic News Translation Text.

UN United Nations (2000-2002). For practical reasons we limit to the portion covering years 2000-2002 (1,339,339 sentence pairs, 50.3 million Arabic words, 45.5 million English words).

Supposedly, 'AE' and 'AR' models are in-domain data, and thus provide higher precision, while 'UN'

[3] `http://www.fjoch.com/GIZA++.html`

[4] The default configuration $1^5 H^5 2^0 3^4 4^3 5^0 6^0$ was used.

is out-of-domain, providing high recall. Language models are estimated over monolingual texts, corresponding to the target language. We use the English side of:

AE Arabic English Parallel News.

AR Arabic News Translation Text.

AM ACE 2005 Multilingual Training Corpus.

AU ACE 2005 Multilingual Unsupervised Training Data.

UN United Nations (1993-2002).

During decoding, we assume that entities tend to be translated as blocks, and therefore reordering is not very important in the context of the overall task. Based on this assumption we decide to perform monotonic translations. This will allow us to speed up translation time and, thus, save up a considerable amount of time.

2.3.3 Adjustment of Parameters

Parameters are adjusted following a Minimum Error Rate strategy (Och, 2003). We simply try around 100 configurations over a development set and pick the best one, according to a given metric.

Since models are built on top of out-of-domain data, we decided to use two different development sets. The first set, DEV$_{AE}$ consists of 961 sentence pairs extracted from the 'AE' corpus. We use this set to get an estimate of the in-domain performance of the MT system. The second set, DEV$_{ET}$ is based on a subset of the 'REFLEX' training and development set (v5.0). Specifically, we used the subset of sentence pairs for which both sides consist of more than 6 words and no more than 20. A total of 987 sentence pairs fulfill these requirement.

Moreover, because the final purpose of the MT system is to translate named entities, we optimize it over a set of metrics specialized in this aspect of quality. We use the 'NE' family of metrics available in the IQMT Framework for MT Evaluation based on Human Likeness (Giménez and Amigó, 2006; Amigó et al., 2006). Named entities in the output and reference translations are automatically annotated, and compared. Specifically we use two metrics:

metric	DEV$_{AE}$	DEV$_{ET}$
BLEU-4	0.19	0.06
GTM-1	0.17	0.12
MTR-wnsyn	0.56	0.23
NIST-5	5.55	2.65
RG-W-1.2	0.23	0.15
NE-overlap-**	0.30	0.12
NE-match-*	0.37	0.10

Table 5: MT results on two development sets: in-domain set (DEV$_{AE}$), and out-of-domain set (DEV$_{ET}$, according to several automatic evaluation metrics.

NE-overlap-** which considers the average proportion of word overlapping over all NE types.

NE-match-* which considers the average lexical matching over all NE types.

We measure NE quality as:

$$\frac{(\text{NE-overlap-**} + \text{NE-match-*})}{2}$$

2.3.4 Performance

Table 5 shows the MT subsystem performance over two different development sets, according to several well-known metrics, for the parameter configuration which yields a maximum NE quality in each case.

Results on the DEV$_{ET}$ test set are much lower than on the DEV$_{AE}$ in-domain set. This means that the domain of the data used to train the MT system is very different from the shared-task domain. This will penalize the system very severely. The decrease in the case of the two NE specialized metrics may be due as well to the serious disfluencies in the MT output which may have caused a number of spurious errors comitted by the NERC system when annotating it.

Regarding efficiency, we must distinguish between system construction and translation times. Training time depends on the amount of data used for the construction of the MT component. A baseline MT system, using only the small 'AE' corpus was built in one day on a 3.2GHz Pentium IV machine. However, processing the very large 'UN' corpus required nearly three weeks.

The whole entity translation of the 150 test documents took around 3 hours: 45 minutes spent on Arabic linguistic pre-processing, 45 minutes properly on translation, and 1.5 hours on finding entity NE counterpart translations.

Memory requirements are governed in our case by the the size of language and translation models (MT component), which must be stored into memory during translation. Current models require around 1 Gigabyte. It is also true that, since the set of documents to be translated is known a priori, we could have optimized memory consumption by storing only information related to those words which actually appear in the source. By proceeding in this way memory consumption reduces to less than 100 Megabytes.

2.4 Resources

2.4.1 Corpora

AE Arabic English Parallel News Text Part 1 (LDC2004T18) – 68,685 sentence pairs; 2 million Arabic words, 2.5 million English words.

AR Arabic News Translation Text Part 1 (LDC2004T17) – 18,564 sentence pairs; 440K Arabic words, 579K English words.

AM ACE 2005 Multilingual Training Corpus (LDC2006T06) – 345K words.

AU ACE 2005 Multilingual Unsupervised Training Data, English v1.0 (LDC2005E20) – 6.2 million words.

UN United Nations Arabic English Parallel Text Version 2 (LDC2004E13) – 3,776,776 sentence pairs; 138.5 Arabic words, 129.2 English words.

2.4.2 Gazeteers

The gazetteers used in our system belong to *BADR (Barcelona Arabic Database for Named Entity Recognition)*. BADR is being created for investigation scopes for Arabic NERC, MT, and other NLP tasks. BADR is freely available on-line[5]. The whole of the entries of BADR are in MSA (Modern Standard Arabic). Its composition is as follows:

BARTIme: temporal expressions.

BARMOney: monetary expressions.

BARNAme1..BARNAme6: names of people.

BARCO: organizations, associations, names of companies.

BARLO: locations, cities, districts.

Currently, English Translations for 1,945 items are available.

2.4.3 Tools

Linguistic Processing of Arabic is carried out by means of the ASVMTools[6] for Arabic (Diab et al., 2004), which are based on Support Vector Machines (SVM). Sentences are transformed into Buckwalter's encoding, tokenized, lemmatized, part-of-speech (PoS) tagged, and base phrase chunked. Language models are built using the *SRI Language Modeling Toolkit* (Stolcke, 2002) for language modeling. Word alignments are obtained using the *GIZA++ SMT Toolkit* (Och and Ney, 2003).

3 Evaluation

3.1 Results

Recognition

In turning to an analysis of the system's errors, of the 4189 references, the system identified 853 (20.36%) correctly, partially identified 1089 (26.00%) and failed to identify 2247 (53.64%). In addition, it identified 3431 false positives. Of the 4189 references, 1832 were by way of proper names, 2093 by way of common noun phrases and 264 by way of pronouns. Here, the system identified 499 (27.24%) proper names correctly while partially identifying 389 (21.23%) and failing to identify 944 (51.53%). It identified 355 (16.96%) common noun phrases correctly while partially identifying 623 (29.77%) and failing to identify 1115 (53.27%). It identified 8 (03.03%) of the pronouns correctly while partially identifying 68 (25.76%) and failing to identify 188 (71.21%). In every case, the system incorrectly identified a significant number of false positives including 1360 false proper name references, 1760 false common noun phrase references

[5] http://www.lsi.upc.edu/~halkoum/badr.html

[6] http://www1.cs.columbia.edu/~mdiab/software/ASVMTools_2.0.tar.gz

and 311 false pronominal references. Clearly the major problem was in handling pronominal references which was at least in part due to an inadequate reference resolution procedure. The system performed significantly better with respect to recognizing proper name references as compared to common noun phrase references.

Translation

In regard to the system's translation errors, the results of the diagnostic run with perfectly correct entity detection indicate that the system translated 1066 (25.45%) correctly, translated 1068 (25.50%) partially correctly and failed to translate 2055 (49.05%). In addition, it translated 4635 false positives. Of the 1832 proper names, the system translated 627 (34.22%) correctly while translating 318 (17.36%) partially correctly and mistranslating 887 (48.42%). Of the 2093 common noun phrases, the system translated 433 (20.69%) correctly while translating 667 (31.87%) partially correctly and mistranslating 993 (47.44%). Of the 264 pronouns, the system only translated 6 (02.27%) correctly while partially translating 83 (31.44%) and failing to translate 175 (66.29%). In every case, the system incorrectly translated a significant number of false positives including 1917 false proper names, 2365 false common noun phrases and 353 false pronouns. The system performed better with respect to translating common noun phrases as compared to proper names and significantly better than pronominal expressions. Still while it is difficult to tease apart errors of translation from errors of detection and recognition, it is clear that the translation system did not perform as well as could be expected.

3.2 Error Analysis

In order to glean some additional information about the system's behavior, we categorized 498 of the 1089 partial identifications with respect to six types of errors. These included examples of:

1. misidentified entities:
   - المراسل (*"the journalist"*), wrongly identified as NE because the article has not been separated by the light-stemmer.
   - الانفصالية هى المسءولة (*"separatism is responsible of"*), wrongly identified as NE because partial overlapping with a real NE.
   - الشرطة (*"the police"*), wrongly identified as NE because the article has not been separated by the light-stemmer.
2. partially identified entities:
   - رءيس (*"President"*), translated as *"British Prime Minister"*.
   - وزير (*"Minister"*), translated *"Trade Minister"*.
   - حكومة (*"Government"*), translated *"Clinton administration"* .
3. misclassified entities:
   - محافظة (*"Province government"*), classifed as PER (i.e., person), where the correct class is LOC (i.e., location)
   - بوركينا فاسو (*"Burkina Faso"*), classified as PER, where the correct class is LOC.
4. mistranslated entities:
   - هم, translated as *"them-in"*, should be *"them"*.
   - ايرلاندا الشمالية translated as *"a a"*, should be *"North Ireland"*.
   - وكالة شينخوا ل ال #٢ #نباء translated as *"of of of of"*, should be *XINHUA press agency"*.
5. partially translated entities:
   - الاسلحة translated as *"of weapons"*, *"of"* has been wrongly added.

Of the 498 items inspected, 274 (55.02%) were misidentified named entities (of which 115, i.e., 23.09%, were due to poor stemming), 20 (4.02%) were partially identified named entities, 16 (3.21%) were misclassified, 132 (26.51%) were mistranslated, 38 (7.63%) were partially translated and 18 (3.61%) were both misclassified and mistranslated.

4 Conclusions and Further Work

From the nine teams which had originally communicated their intention to participate, only three managed to present a system to the ET task. However,

our system performance is far from the top-scoring system.

The room for improvement is large. First, the isolated behaviour of the different components must be improved. In the case of the MT subsystem, the main limitation is that traditional phrase-based SMT models work on sequences of words (i.e., phrases) independently from their entity type, and the context surrounding them in the source sentence is not fully exploited. Other kinds of models should be employed instead. For instance, ET specialized discriminative models in the shape of Giménez and Màrquez (2007)'s could be better suited for this task, specially taking into account that the goal is not to translate the whole sentence but only some pieces of it. In the case of the coreference resolution component, it could be easily extended to handle non-pronominal coreference, as well as some forms of cataphora. Given that in the context of this system the component should deal with coreference between entities, we think there is still further room for a more accurate tuning of this generic subsystem to the task. In addition, the influence of the used features, as well as the addition of new ones, such as the ones used by (Mitkov et al., 1998), should be studied.

Second, we can improve the integration of the system components. Currently, components have been integrated in a hard manner, both at construction and usage times. The three subsystems have been trained independently from each other, and are run independently, thus they do not benefit from each other's results for the purpose of the whole task. Better integration methods must be studied.

Finally, although we focus in Arabic-to-English, our approach is language independent. At the short term, we plan to apply it to new language pairs, such as Chinese-English.

Acknowledgements

This research has been funded by the Spanish Ministry of Science and Technology, project ALIADO (TIC-2002-04447-C02), the European Union, project CHIL (IP 506909), and the by the United States Central Intelligence Agency (Arabic WordNet project). We are recognized as a Quality Research Group (2005 SGR-00130) by DURSI, the Research Department of the Catalan Government. Mihai Surdeanu is a research fellow within the Ramón y Cajal program of the Spanish Ministry of Education and Science.

References

Enrique Amigó, Jesús Giménez, Julio Gonzalo, and Lluís Màrquez. 2006. MT Evaluation: Human-Like vs. Human Acceptable. In *Proceedings of COLING-ACL06*.

D. Connolly, J.D. Burger, and D.S. Day. 1997. A machine learning approach to anaphoric reference. *New Methods in Language Processing*, pages 133–144.

K. Crammer and Y. Singer. 2003. Ultraconservative algorithms for multiclass problems. *Journal of Machine Learning Research*, 3.

Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. 2004. Automatic Tagging of Arabic Text: From raw text to Base Phrase Chunks. In *Proceedings of HLT/NAACL*.

Y. Freund and R.E. Shapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37.

Jesús Giménez and Enrique Amigó. 2006. IQMT: A Framework for Automatic Machine Translation Evaluation. In *Proceedings of the 5th LREC*.

Jesús Giménez and Lluís Màrquez. 2005. Combining Linguistic Data Views for Phrase-based SMT. In *Proceedings of the Workshop on Building and Using Parallel Texts, ACL*.

Jesús Giménez and Lluís Màrquez. 2007. Discriminative Phrase Translation for Phrase-based Statistical Machine Translation. In *(forthcoming)*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of HLT/NAACL*.

Philipp Koehn. 2004. Pharaoh: a Beam Search Decoder for Phrase-Based Statistical Machine Translation Models. In *Proceedings of AMTA*.

R. Mitkov, L. Belguith, and M. Stys. 1998. Multilingual robust anaphora resolution. In *Proceedings of the EMNLP*, pages 7–16.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

Franz Josef Och. 2002. *Statistical Machine Translation: From Single-Word Models to Alignment Templates*. Ph.D. thesis, RWTH Aachen, Germany.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*.

F. Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6).

Andreas Stolcke. 2002. SRILM - An Extensible Language Modeling Toolkit. In *Proceedings of ICSLP*.

V.N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer.

X. Yang, G. Zhou, J. Su, and C.L. Tan. 2003. Coreference resolution using competition learning approach. In *Proceedings of the ACL*, pages 176–183.