NYU's English ACE 2005 System Description

Ralph Grishman **David Westbrook** **Adam Meyers**

Proteus Project
Department of Computer Science
New York University
New York, NY, 10003, USA
(grishman,westbroo,meyers)@cs.nyu.edu

Abstract

In this report we present the overall architecture for the NYU English ACE 2005 system. We focus on two components which were evaluated this year: reference resolution, where we experimented with features for relating anaphor and antecedent, and event recognition, where we sought to take advantage of a rich combination of logical grammatical structure and predicate-argument structure, building on the recent work on PropBank and NomBank.

1 System Overview

Our English extraction system, developed over the course of the last several ACE evaluations, includes facilities for the EDR (entity), RDR (relation), and VDR (event) tasks; TIMEX and Value detection have not yet been implemented. Based on our most recent development work, we participated this year in two evaluations, for entities given mentions (to test coreference) and for events given entities, values, and time expressions. We present a complete system overview, although some components (such as ACE name recognition) were not used for the evaluations we performed this year.

The English ACE system is built on top of the JET (Java Extraction Toolkit), which was developed at NYU for instructional purposes and is freely available for research purposes.[1] JET uses a Tipster Architecture model with documents with stand-off annotations.

Lexical Lookup. The input text is divided into sentences and tokenized. Tokens are looked up in a large general English dictionary that provides part-of-speech information and the base form of inflected words.

Named Entity Analysis. Named entities are tagged using an HMM trained on the ACE training corpora. The HMM has six states for each name type (PERSON, GPE, …), as well as a not-a-name state. These six states correspond to the token preceding the name; the single name token

[1] http://cs.nyu.edu/cs/faculty/grishman/jet.html

(for names with only one token); the first token of the name; an internal token of the name (neither first nor last); the last token of the name; and the token following the name. These multiple states allow the HMM to capture context information and limited information about the internal structure of the name.

Parsing. The sentences are parsed using a parser trained on the Penn Tree Bank.[2]

GLARFing. We generate a regularized syntactic structure / predicate-argument structure called GLARF (Grammatical and Logical Argument Representation Framework) (Meyers et al. 2001). This incorporates both logical grammatical information (logical subject and object) and predicate-argument information from PropBank and NomBank. This information is used for event identification and for one version of relation identification. This GLARF representation is described in more detail in section 3.1.

Reference resolution: Reference resolution first identifies mentions (referring expressions) and then links co-referring expressions. Until last year we used a hand-coded rule-based system. For this year's evaluation we used a corpus-trained system based on an SVM classifier and a rich feature set. This system is described in more detail in section 2.

ACE Entity Detection and Recognition. Given the output of reference resolution, ACE entity detection is primarily a task of semantic classification of the co-referring mention groups. Classification is performed differently for entities with and without named mentions. If there is a named mention, the type of the entity is obtained from the name tagger. The subtype is determined using a MaxEnt model whose features are the individual tokens of the name, trained on the ACE 2005 corpus. The type and subtype of a nominal mention is determined from the head of the mention (with the exception of a small number of hand-coded cases), with the frequencies of the types and subtypes learned from the ACE corpus.

ACE Relation Detection and Recognition. A relation detection component was implemented for the 2004 ACE evaluation. This component used a nearest-neighbor method, in which the training example most similar to the test example is used to classify the test case. (In case of multiple training examples equally similar to a test case, frequency in the training corpus is used to break ties.) All pairs of mentions within the same sentence and separated by no more than two other mentions were used as training examples and were classified during the test phase.
As features for determining example similarity, we used

- the head of each mention
- the ACE type and subtype of each mention
- the sequence of the heads of the constituents between the two mentions (basically following the highest cut through the parse tree between the two mentions)
- the syntactic relation path between the two mentions, as obtained from a parse tree (This may be either a single syntactic relation, such as *subject* or *object*, or a sequence of such relations interspersed with the heads of the intervening constituents. For example, the subject and object of the verb kill would be connected by the path $subject^{-1}$ *kill* *object*, where $subject^{-1}$ represents the inverse subject relation.)

[2] For historical reasons, we are currently using two different PTB parsers. We use the Charniak parser as the input to GLARF analysis, while we use the Bikel parser (http://www.cis.upenn.edu/~dbikel/software.html) to separately identify constituent structure for the EDR task and some other ACE processing. These should be merged in the future so that only one parser needs to be run.

The combination of the constituent sequence and syntactic path provides somewhat greater coverage than either alone. The syntactic path in principle provides greater generalization; however, in cases where there is a syntactic analysis error (either in training or test) the constituent sequence may match even if the syntactic path does not.

Subsequent to the 2004 evaluation, we implemented a modified version of relation detection (Zhao and Grishman 2005) which employs an SVM classifier and a slightly different feature set (using word bi- and tri-grams, and using logical grammatical relations obtained from GLARF rather than directly from the parse tree).

ACE Event Detection and Recognition. Event detection is anchor-based and uses a two-stage matching process. This process is described in more detail in Section 3.

2 Reference Resolution

The NYU coreference resolver takes input that has been tagged for part-of-speech and parsed, and on which name recognition has been performed. Part-of-speech tags and syntactic analysis are used to identify non-name mentions.

Coreference resolution then proceeds incrementally. Mentions are resolved in order of appearance in the document, and feature extraction depends on previous decisions. Each mention resolution decision has two phases. The system first checks to see whether a simple, high-precision rule can resolve the mention (whether there is an exact string match between two multi-word names, for example), and immediately applies the rule if one is found. Otherwise the decision is passed on to a Support Vector Machine model.

The SVM uses a Gaussian kernel. Each instance in the training and test sets represents a pairing of a mention with a candidate entity. For each mention to be resolved, one such pair is created for every distinct entity constructed so far. While many of the features of an instance pertain only to the noun phrase being resolved and to the furthest-right mention of the candidate entity, some features will be derived from more distant mentions of the entity. For example, the most recent mention of a candidate entity may be a pronoun (which will supply some gender and number agreement features), but we will also include name-derived features for the entity if the pronoun has been linked to a proper noun in a previous coreference decision. During training entities are constructed using the correct coreference links from the key.

The feature set for the SVM model includes standard lexical, syntactic, and morphological features. In addition, the feature set incorporates two semantic similarity measures trained on ACE coreference data. One of these measures tests the semantic compatibility of two mentions, while the other tests the semantic compatibility of the syntactic contexts (e.g., predicates) of two mentions.

In both cases the measure consists of a Jaccard coefficient for co-occurrence sets. In the case of the mention similarity measure, we say that two mention heads co-occur when they corefer somewhere in the corpus. In the case of the context similarity measure, we define the context of a mention as the head of the mention's parent in a dependency tree together with the type of the dependency relation between mention and parent (“subject,” “object”, etc.). Two contexts are then said to co-occur when they appear as the contexts of coreferring mentions somewhere in the corpus. For either a mention or context, then, we define the co-occurrence set as the set of all objects that co-occur with that mention or context. For two objects with co-occurrence sets A and

B, respectively, the value of the similarity measure will be the Jaccard coefficient of A and B, or $|A \cap B| / |A \cup B|$.

For each mention-entity instance, the mention similarity and context similarity are computed between the mention to be resolved and each of the individual mentions of the entity. The maxima of these two similarity values across all entity mentions are then included in the feature set for the instance.

For the evaluation run, parsing (with the Bikel parser) took a total of about 235 minutes, or about 1.5 minutes per document; coreference resolution took a total of about 170 minutes, or about 1.1 minutes per document (on a PC).

3 Event Detection

Event detection is anchor-based and uses a combination of pattern matching and a statistical model.

For every instance of an event in the training corpus, we construct two types of patterns representing the connection between the anchor and the event arguments (this is similar to our approach for relations), and record the type and subtype of the associated event. One pattern is the sequence of constituent heads separating the anchor and arguments (again, following the highest cut in the tree connecting the anchor and an argument). The other pattern is the subgraph of the sentence GLARF graph connecting the anchor to all the event arguments. For each argument, we record its ACE type and subtype and its head.

In a second pass over the corpus, we determine the accuracy of predicting an event given a partial match to either pattern. We record the fraction of matches to the anchor alone which predict a correct event, and the fraction of matches to the anchor and an argument which predict a correct event. Patterns which overgenerate (mostly predict incorrect or spurious events) are discarded.

In addition, we train three (MaxEnt) statistical classifiers:

- A (binary) classifier that predicts, given an anchor and a mention, whether this mention is an argument of an event anchored there ("argument classifier")
- A classifier which, given an anchor and mention which is an argument of an event anchored there, predicts the event role to be assigned ("role classifier")
- A (binary) classifier that predicts, given an anchor, an event type, and a set of arguments of this possible event, whether there is a reportable event mention ("event classifier").

The intuition is that the patterns capture precise structures, possibly involving several arguments, associated with particular anchors. For these specific structures, we capture statistics (counts) of whether they are reliable event indicators. The patterns, however, do not capture generalizations over different anchors or even over different event types; the statistical classifiers, while less precisely targeted, do capture these generalizations.

The test corpus is scanned for instances of anchors from the training corpus. When an instance is found, we first try to match the environment of the anchor against the set of patterns associated with that anchor – both constituent sequence and GLARF patterns. In each case, a partial match (matching just a subset of the arguments) is allowed. If multiple patterns match, they are ranked based on the degree of match: the number of arguments matched, and whether the argument

matched an ACE mention of some type (weakest match), the same ACE type, the same ACE subtype, or the same head (strongest match).

This pattern-matching process, if successful, will assign some of the mentions in the sentence as arguments of a potential event. The argument classifier is applied to the remaining mentions in the sentence; for any passing that classifier, we use the role classifier to assign a role to the mention. Finally, once all arguments have been assigned, we apply the event classifier to the potential event; if the result is successful, we report this as an event mention.

Finally, we train a (MaxEnt) classifier for event coreference, using as features the event type and subtype, the anaphor anchor, the distance between anaphor and anchor, whether the anaphor and antecedent have the same anchor, the roles for which the anaphor and antecedent have the same argument value, and the roles which are filled in the anaphor and antecedent with different (non-null) values. Given an event mention, we search for the prior event (of the same subtype) which is most likely to corefer, and merge it with that prior event as long as the probability of coreference exceeds some threshold (40% in the official run).

For the evaluation run, parsing (with the Charniak parser) took about 90 minutes, GLARF generation took about 45 minutes, and event generation took less than 15 minutes (on a PC).

3.1 GLARF

For each sentence in the corpus, we derived a predicate argument representation in two stages, using the 2001 Charniak parser for the first stage and an updated version of a GLARF program (http://nlp.cs.nyu.edu/meyers/GLARF.html) for the second stage. This second stage output includes a predicate argument structure graph. From this graph, we generate a set of n-tuples, one for each predicate argument relation. These n-tuples include both logical grammatical relations (subject, direct / indirect object, etc. factoring in the effects of passive, relative clauses, control, etc.) and PropBank/NomBank relations (filling in roles according to frames, taking into account verb alternations, nominalization, and sense differences). In past measurements, the GLARF processor has achieved about 80% accuracy (precision/recall on tuples) when applied to Charniak parser output. Below is the set of GLARF n-tuples derived from the sentence (from the training corpus): *At least 19 people were killed and 114 people were wounded in Tuesday's southern Philippines airport blast, officials said, but reports said the death toll could climb to 30.* The columns represent the following information about a relation between two words: (1) The GLARF role; (2) The PropBank/NomBank role; (3) The functor (the word and character offset); (4) the verb corresponding to a nominalization (or model verb for non-nominalization); (5) the PropBank/NomBank roleset number (sense); and (6) the argument (word and character offset).

To derive the predicate argument analysis underlying the n-tuples, the GLARF processor put the parser output through a series of filters, each filter completing one of the following functions: (1) fixing errors in constituent boundaries, part of speech, etc.; (2) identifying grammatical roles (heads, subjects, objects, etc.) in phrases, labeling passives, adding constituent boundaries (e.g., for conjoined phrases), and other syntactic information; (3) regularizing special constructions, e.g., determining that *19 people* and *114 people* are the logical objects of *wounded* and *killed*; and (4) assigning PropBank/NomBank roles and senses. These procedures were based on hand-coded rules and dictionary resources including: NOMLEX, Comlex Syntax, the frame lexicons for PropBank/Nombank, and others. We are in the process of adding some corpus-based procedures for use in the next version of our system.

GLARF	PropBank/ NomBank	Functor	Verb	Sense	Argument
CONJ		*but* 305			*said* 299
CONJ		*but* 305			*said* 317
SBJ	ARG0	*said* 299		1	*officials* 289
CONJ		*and* 212			*killed* 205
CONJ		*and* 212			*wounded* 232
T-POS		*people* 193			*19* 190
ADV		*19* 190			*At* 181
A-POS		*19* 190			*least* 184
OBJ	ARG1	*killed* 205		1	*people* 193
AUX		*killed* 205		1	*were* 200
T-POS		*people* 220			*114* 216
COMP	LOC	*wounded* 232		1	*in* 240
OBJ	ARG1	*wounded* 232		1	*people* 220
AUX		*wounded* 232		1	*were* 227
OBJ		*in* 240			*blast* 282
T-POS	TMP	*blast* 282	BLAST	1	*Tuesday* 243
A-POS		*blast* 282	BLAST	1	*southern* 253
N-POS	ARG0	*blast* 282	BLAST	1	*Philippines* 262
N-POS	ARG1	*blast* 282	BLAST	1	*airport* 274
SUFFIX		*Tuesday* 243			*'s* 250
COMP	ARG1	*said* 299		1	*killed* 205
COMP	ARG1	*said* 299		1	wounded 232
SBJ	ARG0	*said* 317		1	*reports* 309
COMP	ARG1	*said* 317		1	*climb* 343
SBJ	ARG0	*climb* 343		1	*toll* 332
T-POS		*toll* 332	TALLY	2	*the* 322
N-POS	ARG1	*toll* 332	TALLY	2	*death* 326
COMP	ARG1	*climb* 343		1	*to* 349
AUX	MOD	*climb* 343		1	*could* 337
OBJ		*to* 349			*30* 352
	ARG1	*reports* 309	REPORT	1	*climb* 343
	Support	*reports* 309	REPORT	1	*said* 317

3.2 The role of GLARF in event recognition

A desirable representation of sentence structure will provide a common structure for a wide variety of sentences which mean the same things (or, more precisely, which represent the same ACE event), and will make explicit the arguments of predicates (of events). The version of GLARF that we used for this evaluation satisfies this goal in several regards: it maps several linguistic variations into the same structure:

- it incorporates logical grammatical structure, so that corresponding active and passive clauses have the same structure

- it makes explicit the subjects controlled by raising and control verbs; subjects in relative and reduced relative sentences; and subjects of sentence modifiers ("Looking over the proposals, Joe … ")
- it incorporates the predicate-argument labels for verbs from PropBank, thus reducing some argument alternations to a common representation (for example, the victim is the surface and logical subject in "He said Sharif drowned." while in "the women from Texas who heinously drowned her five kids" it is the surface and logical object; however, in predicate-argument structure it is the ARG1 in both sentences)
- similarly, it incorporates the predicate-argument labels for nominals from NomBank; this includes not only direct modifiers of a nominal which act as arguments, but also more distant arguments which are connected by support verbs ("Germany launched an attack on X")
- through NomBank, we can also capture alternations between clauses and the corresponding nominalizations (for example, if we see in the training corpus an example of "Germany's destruction of Poland", we will also generate a pattern which would match "Germany destroyed Poland" as an *attack* event)

In addition, GLARF explicitly represents co-ordinate conjoined structures and (in this version of GLARF) connects a separate argument link from the predicate to each of a set of conjoined elements. For example, in "Fred and Jim shot Harry" there will be separate ARG0 arcs from shot to Fred and Jim. The pattern matching procedure will match a single arc in the pattern against multiple arcs with the same source and label, so a non-conjoined training example will be sufficient to capture multiple arguments in a conjoined test example.

Acknowledgement

This research was supported in part by the Defense Advanced Research Projects Agency.

References

(Meyers et al. 2001) Adam Meyers, Michiko Kosaka, Satoshi Sekine, Ralph Grishman and Shubin Zhao . Parsing and GLARFing. In *Proceedings of RANLP-2001*.

(Zhao and Grishman 2005) Shubin Zhao and Ralph Grishman. Extracting relations with integrated information using kernel methods. *Proc. 43rd Annl. Meeting Assn. for Computational Linguistics*, Ann Arbor, MI, June 2005.