# An Information Extraction Customizer

Ralph Grishman and Yifan He

Department of Computer Science, New York University
New York, NY, USA
grishman@cs.nyu.edu, yhe@cs.nyu.edu
http://nlp.cs.nyu.edu

**Abstract.** When an information extraction system is applied to a new task or domain, we must specify the classes of entities and relations to be extracted. This is best done by a subject matter expert, who may have little training in NLP. To meet this need, we have developed a toolset which is able to analyze a corpus and aid the user in building the specifications of the entity and relation types.

**Keywords:** information extraction, distributional analysis

## 1 Introduction

Information extraction (IE) systems, by rendering selected types of relations and events in natural language text into a structured form, make it possible to access, analyze, and reason about this information. A major obstacle to the wider use of IE is the need to customize the IE system to the particular relations and events of interest. There has been considerable research over the past 20 years to reduce the effort required to customize a system, using semi-supervised methods, unsupervised methods, and most recently distant supervision [7]. While there has been substantial progress, there are still significant limits on the performance of systems produced through largely-automatic customization.

In consequence, a number of research groups have developed interactive tools to aid in this customization. This paper describes one such system, integrating tools for corpus analysis and linguistic model building. In particular, our long term objective is to see whether such a tool can be effectively used after limited training by someone who may be a subject matter expert but is not an expert in computational linguistics. To this end, we set the following design goals:

- interaction with the user should be in terms of text examples; use of NLP terminology and formalisms should be minimized;
- the system should have a flexible control structure: a user who knows what to do next should be free to do it; a user who is temporarily at a loss can be guided by the system;
- guidance and assistance should be provided through corpus analysis (in particular, distributional analysis) tools.

P. Sojka et al. (Eds.): TSD 2014, LNAI 8655, pp. 3–10, 2014.

## 2 Extraction System

The system which is being customized is the NYU JET [Java Extraction Toolkit] [8], which was initially developed both for instructional purposes and for participation in the ACE [Automatic Content Extraction] evaluations [1]. It includes general linguistic analysis tools (for part-of-speech tagging, chunking, name tagging, coreference resolution, etc.) and tools for extracting mentions of entities, relations, and events.

The entity extractor identifies noun phrases headed by names or common nouns and classifies them by semantic type and subtype. The base system uses the 7 main types from ACE (person, organization, geo-political entity, location, facility, vehicle, and weapon). Nouns are classified by look-up in a semantic dictionary; names are identified and classified using a Maximum Entropy Markov Model (MEMM). Both dictionary and MEMM are trained from the ACE corpora.

The relation extractor identifies semantic relations between pairs of entities. There are approximately 20 types and subtypes of relations defined by ACE; the details changed from year to year during the series of evaluations. ACE required that a relation mention link two entity mentions in the same sentence, so a relation extractor can be structured as a classifier over pairs of entity mentions in the same sentence. JET includes several such classifiers, including one reliant only on the lexicalized dependency path between the entity mentions and one based on a maximum entropy classifier involving several dozen features.

The linguistic analysis tools of JET are used for the corpus analysis functions of the customizer. The one major analysis component imported into JET is a fast dependency analyzer developed by Tratz and Hovy at ISI [15] based on Goldberg's Easy-First parsing procedure [6]. The parser produces trees with fine-grained dependency labels similar to those of the Stanford parser. As we shall see, it is used as the basis for both distributional analysis and feature generation for relations.

## 3 Customization Tool: Entities

The seven main entity types cover many of the entities in news stories. Collections on specific topics, however, are likely to involve additional types. Reports on border security will involve drugs and other contraband; reports on energy will involve types of fuel; business news will involve financial instruments. Given a new corpus, we will want to identify the new classes and assemble the entity sets for these classes.

Our basic approach is to leave the user in control but to provide as much support as possible. Typically, the user can provide a few examples of a phenomenon (entity or relation type) but has difficulty producing a comprehensive list; users will find it much easier to judge examples provided by the system than to come up with additional examples on their own. So we ask the user to provide some *seed* examples, let the system display additional examples based on some similarity metric, and ask the user to pass judgement on these. Since the user is in control, it is also important that the state of the system be relatively *transparent*, so that the user is able to inspect (and possibly correct) the current models.

To find good candidate seeds around which to build our new entity classes, we start with a simple word frequency tool. Words in the corpus are POS tagged and lemmatized;

we provide a ranked list with the frequency of each lemma, or the relative frequency compared to a background corpus (e.g., a general news corpus). We can limit the list to terms not yet assigned a semantic type.

The list will also include common multi-word terms in the corpus. To identify multi-word terms, we first tag the corpus with a noun-phrase chunker. If the noun chunk has $N$ adjectives and nouns preceding the head noun, we obtain $N + 1$ multi-word term candidates consisting of the head noun and its preceding $i$ ($0 \leq i \leq N$) nouns and adjectives. We count the absolute frequency of these candidates and rank them with a ratio score, which is the relative frequency compared to a background corpus. We use the ratio score $S_t$ to measure the representativeness of term $t$ with regard to the given domain.

The top-ranked terms will typically include good candidates to serve as seeds for building new entity sets. If a user has some acquaintance with the domain, they will pick out a few terms as a seed set.

Given a seed set, we compute the distributional similarity of all terms in the corpus with the centroid of the seeds, using the dependency analysis as the basis for computing term contexts. We represent each term with a vector that encodes its syntactic context, which is the label of the dependency relation attached to the term in conjunction with the term's governor or dependent in that relation.

Consider the entity set of *drugs* in a corpus that describes drug enforcement actions. Drugs will often appear in the dependency relations *dobj(sell, drug)* and *dobj(sieze, drug)* (where *dobj* is the direct object relation), thus members in the *drugs* set will share the features *dobj_sell* and *dobj_sieze*. We use PMI [pointwise mutual information] to weight the feature vectors and use a cosine metric to measure the similarity between two term vectors.

The terms are displayed as a ranked list, and the user can accept or reject individual members of the entity set. At any point the user can recompute the similarities and rerank the list (where the ranking is based the centroids of the accepted and rejected terms, following [11]). When the user is satisfied, the entity set can be saved within Jet, and the set of accepted terms will become a new semantic type for tagging further text.

If the user is exploring a new domain, the choice of suitable seeds may not be clear. In such cases the system can make a suggestion. We select seeds with an agglomerative clustering procedure, which consists of the following steps:

- **Intialization.** We assign each term to a cluster which contains only the term itself.
- **Clustering** We perform agglomerative clustering for several iterations. In each iteration, we merge the two most similar clusters. Similarity is measured using the cosine similarity of the cluster centroids. We stop when we have a cluster that has more than six terms.
- **Cluster scoring** For each cluster we obtain, we calculate a cluster score $S_c$ which is log sum of its members. i.e. $S_c = \sum_{t \in c} \log S_t$
- **Seed selection** We select the two terms that are closest to the centroid of the cluster which has the maximum cluster score.

If the suggested seeds seem reasonable, the user can then proceed to expand the entity set as described above.

## 4 Evaluating the Entity Tool

To test the effectiveness of our procedure for acquiring new entity sets, we measured its performance in creating new *drug* and *law enforcement agent* (*agents*) entity types from a collection of approximately 5,000 web news posts from the U.S. Drug Enforcement Administration[1]. We first extracted the terms in this corpus and manually produced two lists of terms: drug names and law enforcement agent mentions. We extracted 3,703 terms from this corpus and identified 119 drug names and 97 law enforcement agent mentions in our list (the "gold standard" sets).

We then ran a simulated version of our customizer in the following manner: 1) we provided the entity set expansion program with two seeds. For the *drug* set, we provided "methamphetamine" and "oxycodone" (frequent terms in the corpus), while for the *agents* set, we provided "special agents" and "law enforcement officers"; 2) the program produced a ranked list of terms; 3) in each iteration, we examined the top N ($N = 20$ in our setting) new terms that had not been examined in previous iterations; 4) if a term is in the gold standard set, we added it to the expander as a positive seed, otherwise, we added it as a negative seed; 5) we continued the expansion with the updated seed set, and stopped after the $k$th ($k = 10$ in our setting) iteration.

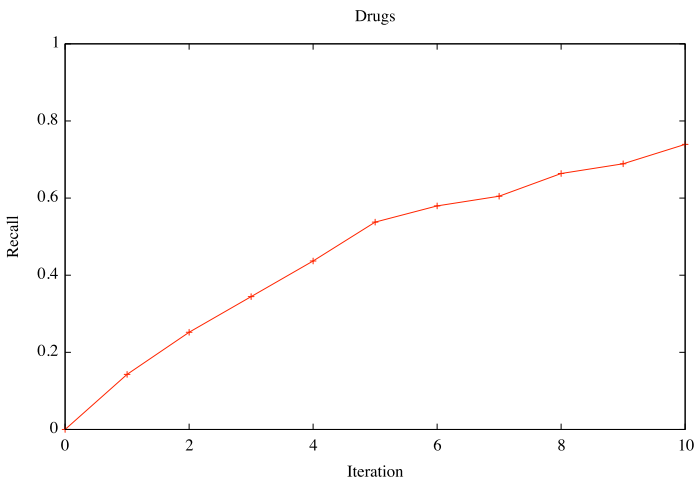


**Fig. 1.** Entity set expansion: recall at each iteration for the *drugs* set.

We measured recall (fraction of terms found) at each iteration of our active learning process and show the result in Figures 1 and 2. Given two seeds, our active learning procedure can help the user to define a new entity set rapidly: for the *drugs* set, the user will be able to find more than 30% of all drugs after 3 iterations (i.e. reviewing 60 of the 3,703 terms); for the *agents* set, the initial 3 iterations will cover more than 40% of

[1] `http://www.justice.gov/dea/index.shtml`

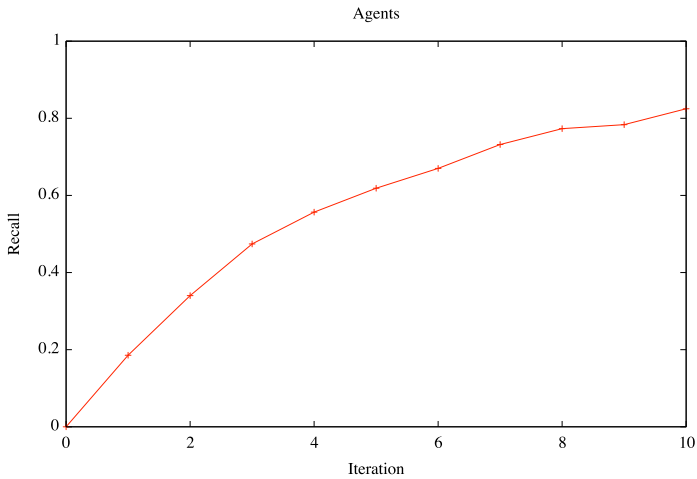


**Fig. 2.** Entity set expansion: recall at each iteration for the *agents* set.

the agents. After 200 terms are reviewed in 10 iterations, the user will be able to build entity sets that cover more than 70% of the *drugs* or 80% of the *agents*.

## 5 Customization Tool: Relations

The semantic relation (if any) between two entity mentions is generally (but not always) determined by the structure connecting the entities in the sentence, and in particular by the lexicalized dependency path between them. The lexicalized dependency path (LDP) includes both the labels of the dependency arcs and the lemmatized form of the lexical items along the path. For example, for the sentence "Cats chase mice." the path from "cats" to "mice" would be $nsubj^{-1}$ *chase dobj*, where the $^{-1}$ indicates that the *nsubj* arc is being traversed from dependent to governor. The LDP connecting two entity mentions, together with the semantic types of the two entities, has been shown to be a good predictor of the semantic relation (if any) between the entities [3].

We collect *all* the dependency paths connecting entity mentions in the corpus, and present them to the user ranked by frequency. As in the case of terms, we can present either the absolute frequencies or the relative frequency compared to a background corpus. This gives the user an overview of the most salient structures in the new corpus. In keeping with our general approach of hiding the linguistic notation from the user, we convert each dependency path back to a phrase which could have generated that path before displaying it to the user.

Just as the user has built up entity sets from seed terms, the user can now build up models of relations from seed LDPs. We are experimenting with two models.

The first model is simply a set of LDPs. The learner is a bootstrap learner: starting with a seed LDP, it gathers all the pairs of arguments (endpoints) which appear with this LDP in the corpus. It then collects all other LDPs which connect any of these pairs in

the corpus, and presents these LDPs to the user for assessment. If the set of argument pairs connected by any of the seeds is $\mathcal{S}$ and the set of argument pairs of a candidate LDP $x$ is $\mathcal{X}$, the candidate LDPs are ranked by $| \mathcal{S} \cap \mathcal{X} | / | \mathcal{X} |$, so that LDPs most distributionally similar to the seed set are ranked highest. The LDPs which are accepted by the user as alternative expressions of the semantic relation are added to the seed set. At any point the user can terminate the bootstrapping and accept the set of LDPs as a model of the relation.

In using the model for extraction, the simplest approach is to accept only exact matches to one of the LDP's. In many cases this is likely to produce high-precision, low-recall extractors. A more flexible approach trains a classifier from both the positively and negatively tagged LDPs, using both the exact LDP and a set of generalizations of the LDP (for example, replacing a word by its semantic class) [3]. This enables some trade-off of recall and precision.

The benefit of this model is that it is relatively transparent. Each LDP can be rendered to the user as a phrase connecting two entities of the specified types, so the user can see the list of such phrases growing as the relation definition develops. On the other hand, this model has limited power: it will not capture words not on the shortest dependency path, and cannot capture generalizations across such paths.

We have therefore also incorporated a second model, a maximum entropy model based on a rich set of features involving the context of the two entity mentions. Examples are presented to the user one-by-one using an extension of an active learning strategy termed *cotesting*; details are described in [13] and [5].

## 6 Related Work

Several groups have developed integrated systems for IE development:

The extreme extraction system from BBN [4] is similar in several regards: it is based on an extraction system initially developed for ACE, allows for the customization of entities and relations, and uses bootstrapping and active learning. However, in contrast to our system, it is aimed at skilled computational linguists.

The Language Computer Corporation has described several tools developed to rapidly extend an information extraction system to a new task [9,14]. Here too the emphasis is on tools for use by experienced IE system developers. Events and relations are recognized using finite-state rules, with meta-rules to efficiently capture syntactic variants and a provision for supervised learning of rules from annotated corpora.

A few groups have focused on use by NLP novices:

The WizIE system from IBM Research [10] is based on a finite-state rule language. Users prepare some sample annotated texts and are then guided in preparing an extraction plan (sequences of rule applications) and in writing the individual rules. IE development is seen as a rule programming task. This offers less in the way of linguistic support (corpus analysis, syntactic analysis) but can provide greater flexibility for extraction tasks where linguistic models are a poor fit.

The Propminer system from T. U. Berlin [2] takes an approach more similar to our own. In particular, it is based on a dependency analysis of the text corpus and emphasizes exploratory development of the IE system, supported by search operations

over the dependency structures. However, the responsibility for generalizing initial patterns lies primarily with the user, whereas we support the generalization process through distributional analysis.

## 7 Future Work

One limitation of the current system is the absence of a tool for capturing new event types (beyond those present in ACE). Some event types can be represented as binary relations, and further accomodation can be achieved by allowing time and location modifiers. However, to handle genuine 3-argument events ("X gave Y to Z") as well as binary events where one argument is omitted ("the American attack") a more general mechanism is required, such as the merging of alternative argument patterns described in [12].

A second limitation is the use of lists (rather than a statistical model) for names belonging to a new entity type. This can be addressed by annotating a single corpus using both the baseline statistical name tagger and the list of new names, training a new statistical model from this corpus, and then (optionally) refining this combined model through active learning.

The greatest challenge, however, will lie in creating a system which is useable by non-NLP experts. This will undoubtably require a number of iterations as we address the needs, concerns, and misunderstandings of users.

**Acknowledgments.** Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Air Force Research Laboratory (AFRL) contract number FA8650-10-C-7058. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, AFRL, or the U.S. Government.

## References

1. Automatic Content Extraction (ACE). `https://www.ldc.upenn.edu/collaborations/past-projects/ace`
2. Akbik, A., Konomi, O., and Melnikov, M. Propminer: a workflow for interactive information extraction and exploration using dependency trees. Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: Systems Demonstrations (2013) 157–162
3. Bunescu, R., and Mooney, R. A shortest path dependency kernel for relation extraction. In: Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (2005) 724–731
4. Freedman, M., Ramshaw, L., Boschee, E., Gabbard, R., Kratkiewicz, G., Ward, N., Weischedel, R. Extreme Extraction – Machine Reading in a Week. In: Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pp. 1437–1446.
5. Fu, L., and Grishman, R. An Efficient Active Learning Framework for New Relation Types. In: Proceedings of International Joint Conference on Natural Language Processing (IJCNLP), pp. 692–698, Nagoya, Japan, 2013.

6. Goldberg, Y., and Elhadad, M. An efficient algorithm for easy-first non-directional dependency parsing. In: Human Language Technologies: the 2010 Annual Conference of the North American Chapter of the ACL, pp. 742–750.
7. Grishman, R. Information Extraction: Capabilities and Challenges. The 2012 International Winter School in Language and Speech Technologies, Rovira i Virgili University, Tarragona, Spain. `http://cs.nyu.edu/grishman/survey.html`. (2012)
8. Java Extraction Toolkit, `http://cs.nyu.edu/grishman/jet/jet.html`
9. Lehmann, J., Monahan, S., Nezda, L., Jung, A., and Shi, Y. Approaches to Knowledge Base Population at TAC 2010. In: Proceedings of the 2010 Text Analysis Conference. (2010)
10. Li, Y., Chiticariu, L., Yang, H., Reiss, F., and Carreno-fuentes, A. WizIE: a best practices guided development environment for information extraction. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics. (2012) 109–114
11. Min, B., and Grishman, R. Fine-grained entity refinement with user feedback. In: Proceedings of RANLP 2011 Workshop on Information Extraction and Knowledge Acquisition. (2011) 2–6
12. Shinyama, Y., and Sekine, S. Preemptive information extraction using unrestricted relation discovery. In: Proceedings of the Human Language Technology Conference of the NAACL. (2006) 304–311
13. Sun, A., and Grishman, R. Active learning for relation type extension with local and global data views. Proc. 21st ACM International Conf. on Information and Knowledge Management (CIKM 2012). (2012) 1105–1112
14. Surdeanu, M., and Harabagiu, S. Infrastructure for Open-Domain Information Extraction. HLT 2002, Proceedings of the Second International Conference on Human Language Technology Research. (2002) 325–330
15. Tratz, S., and Hovy, E. A fast, effective, non-projective, semantically-enriched parser. Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing. Edinburgh, Scotland, UK. (2011) 1257–1268