Identifying 1950s American Jazz Musicians: Fine-Grained IsA Extraction via Modifier Composition Supplementary Material

Ellie Pavlick*
University of Pennsylvania
3330 Walnut Street
Philadelphia, Pennsylvania 19104
epavlick@seas.upenn.edu

Marius Paşca
Google Inc.
1600 Amphitheatre Parkway
Mountain View, California 94043
mars@google.com

1 Input IsA Repository ($\mathcal{O}$)

Our IsA repository $\mathcal{O}$ is extracted from a sample of around 1 billion Web documents in English. $\mathcal{O}$ is constructed by applying the following Hearst patterns to the Web corpus: "C such as I", "I is a C", "C including I", and "C especially I", where I is either a single instance or an enumeration of instances. All patterns receive equal weights, but the second is the most productive in practice.

Instances in $\mathcal{O}$ are represented as automatically-disambiguated entity mentions. We use the entity linker described in Lazic et al. (2015). The "entity mentions" may be individuals, like *"Barack Obama"*, or may be concepts like *"jazz"*. When possible, entity mentions are resolved to Wikipedia pages; for example, *"America"* and *"USA"* will be mapped to the same Wikipedia article. Classes in $\mathcal{O}$ are represented as non-disambiguated strings in natural language.

In building the repository, we retain every $\langle e, C\rangle$ tuple which is supported by 5 or more sentences and has a confidence of at least 0.9. Here, "confidence" is a score that reflects a weighted combination of the number of supporting sentences and the frequency of the category C, where weights are tuned on a hand-labeled tuning set to give a good trade-off between precision and recall of true pairs. The resulting repository contains 1.1M IsA relations, covering 412K instances and 9K categories.

2 Input Fact Repository ($\mathcal{D}$)

Our fact repository $\mathcal{D}$ is extracted from a sample of around 1 billion Web documents in English. $\mathcal{D}$ is extracted using in-house implementations of ReVerb (Fader et al., 2011) and OLLIE (Mausam et al., 2012) applied to the Web corpus. We leave predicates as natural language strings, but remove stop words and apply basic lemmatization (e.g. *"is an important part of"* becomes *"be important part of"*). Subjects and objects may be either disambiguated entity references, as in $\mathcal{O}$ described above, or may be natural language strings. Every tuple is included in both the forward and the reverse direction. For example, $\langle$*jazz*, *perform at*, *venue*$\rangle$ also appears as $\langle$*venue*, $\leftarrow$*perform at*, *jazz*$\rangle$, where $\leftarrow$ is a special character signifying inverted predicates.

The weight w associated with each SPO tuple is the aggregated frequency of number of times the tuple was extracted from the corpus. In total, our fact repository contains 30M tuples.

*Contributed during an internship at Google.

3 Weakly-Supervised Reranking

3.1 Features

Recall that our raw scoring function takes the sum, over all the properties in $I(MH)$, of the property weight multiplied by the number of times we observed the instance in question with that property. I.e. Equation 5 from the paper:

$$\hat{\phi}_M(H, e) = \sum_{\langle\langle p,o\rangle,w\rangle \in I(MH)} w \times \mathcal{D}(\langle e, p, o\rangle)$$

The goal of the trained model is to replace this sum with a more intelligent scoring function. Given an instance e and a class label MH, let

$$\texttt{props} = [w_0 \times \mathcal{D}(\langle e, p_0, o_0\rangle) \ldots w_k \times \mathcal{D}(\langle e, p_k, o_k\rangle)]$$

be the list of count $\times$ weight scores for all of the properties in $I(MH)$ (i.e. the list which is summed over in the equation above), sorted in decreasing order. We then extract the following features:

- For K in {1, 10, 100, 1000, len(props)}
  - sum(`props`[:K])
  - arithmetic_mean(`props`[:K])
  - geometric_mean(`props`[:K])
- `headconf`: Confidence score for $\langle e, MH\rangle$ according to $\mathcal{O}$
- `catcount`: Total number of categories in $\mathcal{O}$ of which e is an instance
- `factcount`: Total number of tuples in $\mathcal{D}$ in which e is the subject

- sum(`props`) / `catcount`
- arithmetic_mean(`props`) / `catcount`
- geometric_mean(`props`)/`catcount`
- sum(`props`) / `factcount`
- arithmetic_mean(`props`) / `factcount`
- geometric_mean(`props`)/`factcount`

All of the features are binarized. We use the log of the value, rounded to the nearest integer, in order to assign values to a bin for all features except for `headconf`, for which the bin is simply the value rounded to two decimal places. For features parameterized by K, the features are only defined for length(`props`)≥K. Otherwise, an indicator feature is set to designate that the length of the list was less than K.

3.2 Training

For training, we take a random sample of $\langle e, MH \rangle$ pairs for which we generated fact profiles and which also appear in $\mathcal{O}$ and consider these to be positive training examples. We select another sample of $\langle e, MH \rangle$ pairs for which we generated fact profiles but do not appear in any Hearst pattern in our corpus and consider these to be negative training examples. That is, we have a stricter requirement for our negative training data than simply not appearing in $\mathcal{O}$: $\mathcal{O}$ includes all $\langle e, C \rangle$ tuples which are supported by at least 5 sentences in our corpus, and our negative training data comes only from $\langle e, C \rangle$ tuples which were supported by 0 sentences in the corpus. The resulting training set contains 3M pairs of which 45% are positive and the remaining are negative.

We train a standard logistic regression model implemented in the scikit-learn Python toolkit[1]. We tune the regularization parameter using cross validation on the training data. On cross validation, the trained model achieves 65% accuracy over the 45% majority class baseline.

4 Modifier Chunking in Wikipedia Evaluation Sets

All of the class labels in our evaluation sets contain at least three words, but they represent a mix of single modifier (*"puerto rican sculptors"*) and multiple modifier (*"canadian business journalists"*) phrases. Therefore, we perform noun-phrase chunking as a preprocessing step. We use a parser trained to parse queries (Petrov et al., 2010), which gives good performance on short phrases. Given the parse tree, we group together any tokens which share a common parent other than the root node, with the exception of the rightmost token (the head), which we force to appear as a chunk by itself. This heuristic was chosen since, on manual inspection, it produced good chunks. We use these pre-chunked class labels as input to all of the systems, including baselines, in our evaluation. The experiments in this section assume some method for grouping multiword modifiers (*"puerto rican"*), but are not dependent on this particular method for chunking.

[1] http://scikit-learn.org

References

A. Fader, S. Soderland, and O. Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP-11)*. Edinburgh, Scotland, pages 1535–1545.

N. Lazic, A. Subramanya, M. Ringgaard, and F. Pereira. 2015. Plato: A selective context model for entity resolution. *Transactions of the Association for Computational Linguistics* 3:503–515.

Mausam, M. Schmitz, S. Soderland, R. Bart, and O. Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL-12)*. Jeju Island, Korea, pages 523–534.

S. Petrov, P. Chang, M. Ringgaard, and H. Alshawi. 2010. Uptraining for accurate deterministic question parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP-10)*. Cambridge, Massachusetts, pages 705–713.