Extracting Sequences from the Web

Anthony Fader, Stephen Soderland, and Oren Etzioni
University of Washington, Seattle
{afader,soderlan,etzioni}@cs.washington.edu

Abstract

Classical Information Extraction (IE) systems fill slots in domain-specific frames. This paper reports on SEQ, a novel open IE system that leverages a *domain-independent* frame to extract ordered sequences such as presidents of the United States or the most common causes of death in the U.S. SEQ leverages regularities about sequences to extract a coherent set of sequences from Web text. SEQ nearly doubles the area under the precision-recall curve compared to an extractor that does not exploit these regularities.

1 Introduction

Classical IE systems fill slots in domain-specific frames such as the time and location slots in seminar announcements (Freitag, 2000) or the terrorist organization slot in news stories (Chieu et al., 2003). In contrast, open IE systems are domain-independent, but extract "flat" sets of assertions that are not organized into frames and slots (Sekine, 2006; Banko et al., 2007). This paper reports on SEQ—an open IE system that leverages a domain-independent frame to extract ordered sequences of objects from Web text. We show that the novel, domain-independent sequence frame in SEQ substantially boosts the precision and recall of the system and yields coherent sequences filtered from low-precision extractions (Table 1).

Sequence extraction is distinct from set expansion (Etzioni et al., 2004; Wang and Cohen, 2007) because sequences are *ordered* and because the extraction process does not require seeds or HTML lists as input.

The domain-independent *sequence frame* consists of a sequence name s (*e.g.*, presidents of the United States), and a set of ordered pairs (x, k) where x is a string naming a member of the sequence with name s, and k is an integer indicating its position (*e.g.*, (Washington, 1) and (JFK, 35)). The task of *sequence extraction* is to automatically instantiate sequence frames given a corpus of unstructured text.

Most common cause of death in the United States: 1. heart disease, 2. cancer, 3. stroke, 4. COPD, 5. pneumonia, 6. cirrhosis, 7. AIDS, 8. chronic liver disease, 9. sepsis, 10. suicide, 11. septic shock.
Largest tobacco company in the world: 1. Philip Morris, 2. BAT, 3. Japan Tobacco, 4. Imperial Tobacco, 5. Altadis.
Largest rodent in the world: 1. Capybara, 2. Beaver, 3. Patagonian Cavies. 4. Maras.
Sign of the zodiac: 1. Aries, 2. Taurus, 3. Gemini, 4. Cancer, 5. Leo, 6. Virgo, 7. Libra, 8. Scorpio, 9. Sagittarius, 10. Capricorn, 11. Aquarius, 12. Pisces, 13. Ophiuchus.

Table 1: Examples of sequences extracted by SEQ from unstructured Web text.

By definition, sequences have two properties that we can leverage in creating a sequence extractor: *functionality* and *density*. Functionality means position k in a sequence is occupied by a single real-world entity x. Density means that if a value has been observed at position k then there must exist values for all $i < k$, and possibly more after it.

2 The SEQ System

Sequence extraction has two parts: identifying possible extractions (x, k, s) from text, and then classifying those extractions as either correct or incorrect. In the following section, we describe a way to identify candidate extractions from text using a set of lexico-syntactic patterns. We then show that classifying extractions based on sentence-level features and redundancy alone yields low precision, which is improved by leveraging the functionality and density properties of sequences as done in our SEQ system.

Pattern	Example
the ORD	the fifth
the RB ORD	the very first
the JJS	the best
the RB JJS	the very best
the ORD JJS	the third biggest
the RBS JJ	the most popular
the ORD RBS JJ	the second least likely

Table 2: The patterns used by SEQ to detect ordinal phrases are noun phrases that begin with one of the part-of-speech patterns listed above.

2.1 Generating Sequence Extractions

To obtain candidate sequence extractions (x, k, s) from text, the SEQ system finds sentences in its input corpus that contain an ordinal phrase (OP). Table 2 lists the lexico-syntactic patterns SEQ uses to detect ordinal phrases. The value of k is set to the integer corresponding to the ordinal number in the OP.[1]

Next, SEQ takes each sentence that contains an ordinal phrase o, and finds candidate items of the form (x, k) for the sequence with name s. SEQ constrains x to be an NP that is disjoint from o, and s to be an NP (which may have post-modifying PPs or clauses) following the ordinal number in o.

For example, given the sentence "With help from his father, JFK was elected as the 35th President of the United States in 1960", SEQ finds the candidate sequences with names "President", "President of the United States", and "President of the United States in 1960", each of which has candidate extractions (JFK, 35), (his father, 35), and (help, 35). We use heuristics to filter out many of the candidate values (*e.g.*, no value should cross a sentence-like boundary, and x should be at most some distance from the OP).

This process of generating candidate extractions has high coverage, but low precision. The first step in identifying correct extractions is to compute a confidence measure $localConf(x, k, s|sentence)$, which measures how likely (x, k, s) is given the sentence it came from. We do this using domain-independent syntactic features based on POS tags and the pattern-based features "x {is,are,was,were} the kth s" and "the kth s {is,are,was,were} x". The features are then combined using a Naive Bayes classifier.

In addition to the local, sentence-based features, we define the measure *totalConf* that takes into account redundancy in an input corpus $\mathcal{C}$. As Downey *et al.* observed (2005), extractions that occur more frequently in multiple distinct sentences are more likely to be correct.

$$totalConf(x, k, s|\mathcal{C}) = \sum_{sentence \in \mathcal{C}} localConf(x, k, s|sentence) \quad (1)$$

2.2 Challenges

The scores *localConf* and *totalConf* are not sufficient to identify valid sequence extractions. They tend to give high scores to extractions where the sequence scope is too general or too specific. In our running example, the sequence name "President" is too general – many countries and organizations have a president. The sequence name "President of the United States in 1960" is too specific – there were not multiple U.S. presidents in 1960.

These errors can be explained as violations of functionality and density. The sequence with name "President" will have many distinct candidate extractions in its positions, which is a violation of functionality. The sequence with name "President of the United States in 1960" will not satisfy density, since it will have extractions for only one position.

In the next section, we present the details of how SEQ incorporates functionality and density into its assessment of a candidate extraction.

Given an extraction (x, k, s), SEQ must classify it as either correct or incorrect. SEQ breaks this problem down into two parts: (1) determining whether s is a correct sequence name, and (2) determining whether (x, k) is an item in s, assuming s is correct.

A joint probabilistic model of these two decisions would require a significant amount of labeled data. To get around this problem, we represent each (x, k, s) as a vector of features and train two Naive Bayes classifiers: one for classifying s and one for classifying (x, k). We then rank extractions by taking the product of the two classifiers' confidence scores.

We now describe the features used in the two classifiers and how the classifiers are trained.

Classifying Sequences To classify a sequence name s, SEQ uses features to measure the functionality and density of s. Functionality means

[1] Sequences often use a superlative for the first item ($k = 1$) such as "the deepest lake in Africa", "the second deepest lake in Africa" (or "the 2nd deepest ..."), etc.

that a correct sequence with name s has one correct value x at each position k, possibly with additional noise due to extraction errors and synonymous values of x. For a fixed sequence name s and position k, we can weight each of the candidate x values in that position by their normalized total confidence:

$$w(x|k, s, \mathcal{C}) = \frac{totalConf(x, k, s|\mathcal{C})}{\sum_{x'} totalConf(x', k, s|\mathcal{C})}$$

For overly general sequences, the distribution of weights for a position will tend to be more flat, since there are many equally-likely candidate x values. To measure this property, we use a function analogous to information entropy:

$$H(k, s|\mathcal{C}) = -\sum_{x} w(x|k, s, \mathcal{C}) \log_2 w(x|k, s, \mathcal{C})$$

Sequences s that are too general will tend to have high values of $H(k, s|\mathcal{C})$ for many values of k. We found that a good measure of the overall non-functionality of s is the average value of $H(k, s|\mathcal{C})$ for $k = 1, 2, 3, 4$.

For a sequence name s that is too specific, we would expect that there are only a few filled-in positions. We model the density of s with two metrics. The first is $numFilledPos(s|\mathcal{C})$, the number of distinct values of k such that there is some extraction (x, k) for s in the corpus. The second is $totalSeqConf(s|\mathcal{C})$, which is the sum of the scores of most confident x in each position:

$$totalSeqConf(s|\mathcal{C}) = \sum_{k} \max_{x} totalConf(x, k, s|\mathcal{C}) \quad (2)$$

The functionality and density features are combined using a Naive Bayes classifier. To train the classifier, we use a set of sequence names s labeled as either correct or incorrect, which we describe in Section 3.

Classifying Sequence Items To classify (x, k) given s, SEQ uses two features: the total confidence $totalConf(x, k, s|\mathcal{C})$ and the same total confidence normalized to sum to 1 over all x, holding k and s constant. To train the classifier, we use a set of extractions (x, k, s) where s is known to be a correct sequence name.

3 Experimental Results

This section reports on two experiments. First, we measured how the density and functionality features improve performance on the sequence name classification sub-task (Figure 1). Second, we report on SEQ's performance on the sequence-extraction task (Figure 2).

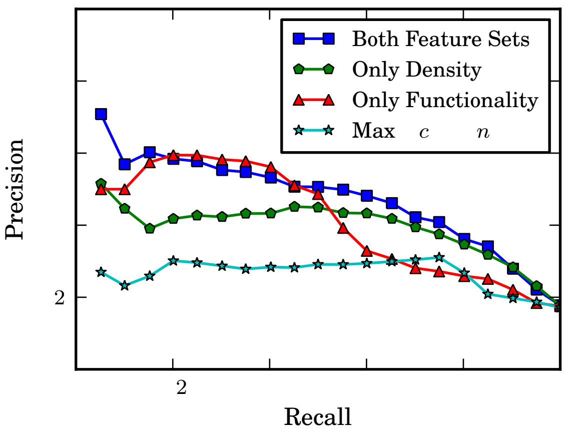

Figure 1: Using density or functionality features alone is effective in identifying correct sequence names. Combining both types of features outperforms either by a statistically significant margin (paired t-test, $p < 0.05$).

To create a test set, we selected all sentences containing ordinal phrases from Banko's 500M Web page corpus (2008). To enrich this set O, we obtained additional sentences from Bing.com as follows. For each sequence name s satisfying $localConf(x, k, s|sentence) \geq 0.5$ for some sentence in O, we queried Bing.com for "the kth s" for $k = 1, 2, \ldots$ until no more hits were returned.[2] For each query, we downloaded the search snippets and added them to our corpus. This procedure resulted in making $95,611$ search engine queries. The final corpus contained $3,716,745$ distinct sentences containing an OP.

Generating candidate extractions using the method from Section 2.1 resulted in a set of over 40 million distinct extractions, the vast majority of which are incorrect. To get a sample with a significant number of correct extractions, we filtered this set to include only extractions with $totalConf(x, k, s|\mathcal{C}) \geq 0.8$ for some sentence, resulting in a set of $2,409,211$ extractions.

We then randomly sampled and manually labeled $2,000$ of these extractions for evaluation. We did a Web search to verify the correctness of the sequence name s and that x is the kth item in the sequence. In some cases, the ordering relation of the sequence name was ambiguous (*e.g.*,

[2]We queried for both the numeric form of the ordinal and the number spelled out (e.g "the 2nd ..." and "the second ..."). We took up to 100 results per query.

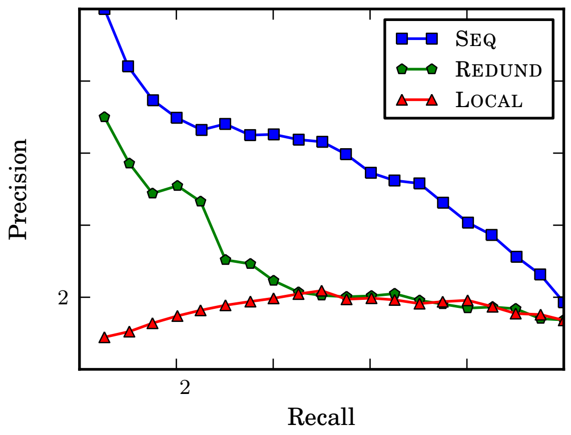

Figure 2: SEQ outperforms the baseline systems, increasing the area under the curve by 247% relative to LOCAL and by 90% relative to REDUND.

"largest state in the US" could refer to land area or population), which could lead to merging two distinct sequences. In practice, we found that most ordering relations were used in a consistent way (*e.g.*, "largest city in" always means largest by population) and only about 5% of the sequence names in our sample have an ambiguous ordering relation.

We compute precision-recall curves relative to this random sample by changing a confidence threshold. Precision is the percentage of correct extractions above a threshold, while recall is the percentage correct above a threshold divided by the total number of correct extractions. Because SEQ requires training data, we used 15-fold cross validation on the labeled sample.

The functionality and density features boost SEQ's ability to correctly identify sequence names. Figure 1 shows how well SEQ can identify correct sequence names using only functionality, only density, and using functionality and density in concert. The baseline used is the maximum value of $localConf(x, k, s)$ over all (x, k). Both the density features and the functionality features are effective at this task, but using both types of features resulted in a statistically significant improvement over using either type of feature individually (paired t-test of area under the curve, $p < 0.05$).

We measure SEQ's efficacy on the complete sequence-extraction task by contrasting it with two baseline systems. The first is LOCAL, which ranks extractions by $localConf$.[3] The second is REDUND, which ranks extractions by $totalConf$. Figure 2 shows the precision-recall curves for each system on the test data. The area under the curves for SEQ, REDUND, and LOCAL are 0.59, 0.31, and 0.17, respectively. The low precision and flat curve for LOCAL suggests that $localConf$ is not informative for classifying extractions on its own.

REDUND outperformed LOCAL, especially at the high-precision part of the curve. On the subset of extractions with correct s, REDUND can identify x as the kth item with precision of 0.85 at recall 0.80. This is consistent with previous work on redundancy-based extractors on the Web. However, REDUND still suffered from the problems of over-specification and over-generalization described in Section 2. SEQ reduces the negative effects of these problems by decreasing the scores of sequence names that appear too general or too specific.

4 Related Work

There has been extensive work in extracting lists or sets of entities from the Web. These extractors rely on either (1) HTML features (Cohen et al., 2002; Wang and Cohen, 2007) to extract from structured text or (2) lexico-syntactic patterns (Hearst, 1992; Etzioni et al., 2005) to extract from unstructured text. SEQ is most similar to this second type of extractor, but additionally leverages the sequence regularities of functionality and density. These regularities allow the system to overcome the poor performance of the purely syntactic extractor LOCAL and the redundancy-based extractor REDUND.

5 Conclusions

We have demonstrated that an extractor leveraging sequence regularities can greatly outperform extractors without this knowledge. Identifying likely sequence names and *then* filling in sequence items proved to be an effective approach to sequence extraction.

One line of future research is to investigate other types of domain-independent frames that exhibit useful regularities. Other examples include *events* (with regularities about actor, location, and time) and a generic *organization-role* frame (with regularities about person, organization, and role played).

[3]If an extraction arises from multiple sentences, we use the maximal $localConf$.

6 Acknowledgements

This research was supported in part by NSF grant IIS-0803481, ONR grant N00014-08-1-0431, DARPA contract FA8750-09-C-0179, and an NSF Graduate Research Fellowship, and was carried out at the University of Washington's Turing Center.

References

Michele Banko and Oren Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *Proceedings of ACL-08: HLT*, pages 28–36.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI*, pages 2670–2676.

H. Chieu, H. Ng, and Y. Lee. 2003. Closing the gap: Learning-based information extraction rivaling knowledge-engineering methods. In *ACL*, pages 216–223.

William W. Cohen, Matthew Hurst, and Lee S. Jensen. 2002. A flexible learning system for wrapping tables and lists in html documents. In *In International World Wide Web Conference*, pages 232–241.

Doug Downey, Oren Etzioni, and Stephen Soderland. 2005. A probabilistic model of redundancy in information extraction. In *IJCAI*, pages 1034–1041.

O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2004. Methods for domain-independent information extraction from the Web: An experimental comparison. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-2004)*, pages 391–398.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana maria Popescu, Tal Shaked, Stephen Soderl, Daniel S. Weld, and Er Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165:91–134.

D. Freitag. 2000. Machine learning for information extraction in informal domains. *Machine Learning*, 39(2-3):169–202.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING*, pages 539–545.

Satoshi Sekine. 2006. On-demand information extraction. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 731–738, Morristown, NJ, USA. Association for Computational Linguistics.

Richard C. Wang and William W. Cohen. 2007. Language-independent set expansion of named entities using the web. In *ICDM*, pages 342–350. IEEE Computer Society.