Event Extraction Using Distant Supervision

Kevin Reschke1, Martin Jankowiak1, Mihai Surdeanu2
Christopher D. Manning1, Daniel Jurafsky1
1Stanford University, 450 Serra Mall Stanford, CA 94305 USA
2University of Arizona, Gould-Simpson 811, 1040 E. 4th Street Tucson, AZ 85721 USA
kreschke@cs.stanford.edu, jankowiak@gmail.com, msurdeanu@email.arizona.edu
manning@cs.stanford.edu, jurafsky@stanford.edu

Abstract
Distant supervision is a successful paradigm that gathers training data for information extraction systems by automatically aligning vast databases of facts with text. Previous work has demonstrated its usefulness for the extraction of binary relations such as a person’s employer or a film’s director. Here, we extend the distant supervision approach to template-based event extraction, focusing on the extraction of passenger counts, aircraft types, and other facts concerning airplane crash events. We present a new publicly available dataset and event extraction task in the plane crash domain based on Wikipedia infoboxes and newswire text. Using this dataset, we conduct a preliminary evaluation of four distantly supervised extraction models which assign named entity mentions in joint learning (here using the S_{earn} algorithm (Daumé III, 2012), it has not previously been applied to event extraction. We make three main contributions. First, we present a new research dataset for distantly supervised event extraction centered around airplane crash events. The dataset consists of a plane crash knowledge base derived from Wikipedia infoboxes and distantly generated entity-level labels covering a corpus of newswire text. Second, we use this dataset to conduct a preliminary evaluation of a number of extraction models. Our results serve as a baseline for further research in this domain. Third, our experiments demonstrate that joint learning (here using the S_{earn} algorithm (Daumé III, 2006)) performs better than several strong baselines, even in this complex and noisy setup.

1. Introduction
This paper explores a distant supervision approach to event extraction for knowledge-base population. In a distantly supervised setting, training texts are labeled automatically (and noisily) by leveraging an existing database of known facts. While this approach has been applied successfully to the extraction of binary relations such as a person’s employer or a film’s director (Mintz et al., 2009; Surdeanu et al., 2012), it has not previously been applied to event extraction. We make three main contributions. First, we present a new research dataset for distantly supervised event extraction centered around airplane crash events. The dataset consists of a plane crash knowledge base derived from Wikipedia infoboxes and distantly generated entity-level labels covering a corpus of newswire text. Second, we use this dataset to conduct a preliminary evaluation of a number of extraction models. Our results serve as a baseline for further research in this domain. Third, our experiments demonstrate that joint learning (here using the S_{earn} algorithm (Daumé III, 2006)) performs better than several strong baselines, even in this complex and noisy setup.

2. Dataset and Slot-filling Task
We began by compiling a knowledge base of 193 plane crash infoboxes from Wikipedia’s list of commercial aircraft accidents.1 An example is shown in Table 1. From these we selected 80 single-aircraft crashes (40 for training; 40 for testing) that occurred after 1987. This is the timespan covered by our news corpus, which is comprised of Tipster-1, Tipster-2, Tipster-3, and Gigaword-5.2

3. Experiments
Having introduced the general framework for distantly supervised event extraction, in this section we present experiments testing various models in the framework.

Keywords: Distant-Supervision, Event-Extraction, S_{earn}

1http://en.wikipedia.org/wiki/List_of_accidents_and_incidents_involving_commercial_aircraft
2Available at catalog.ldc.upenn.edu/LDC93T3A and catalog.ldc.upenn.edu/LDC2011T07.

---

1E.g., *Airbus* is an alias for *Airbus A320-211*, and *eighth* is an alias for *8*.
2Available at catalog.ldc.upenn.edu/LDC93T3A and catalog.ldc.upenn.edu/LDC2011T07.
3http://nlp.stanford.edu/projects/dist-sup-event-extraction.shtml
4http://nlp.stanford.edu/software/CRF-NER.shtml
5http://nlp.stanford.edu/software/CRF-NER.shtml
3.1. Experiment 1: Simple Local Classifier

First we used multi-class logistic regression to train a model which classifies each mention independently, using the noisy training data described above. Features include the entity mention’s part of speech, named entity type, surrounding unigrams, incoming and outgoing syntactic dependencies, the location within the document and the mention string itself. These features fall into five groups, detailed in Table 3. Each of the models described in this paper uses these five features sets.

We compare this local classifier to a majority class baseline. The majority baseline assigns the most common label for each named entity type as observed in the training documents (see Table 2). Concretely, all locations are labeled ⟨Site⟩, all organizations are labeled ⟨Operator⟩, all numbers are labeled ⟨Fatalities⟩, and all other named entities are labeled NIL. The remaining five labels are never assigned.

To compare performance on the final slot prediction task, we define precision and recall as follows. Precision is the number of correct guesses over the total number of guesses. Recall is the number of slots correctly filled over the number of correct guesses.

\[
\text{Precision} = \frac{\text{Number of correct guesses}}{\text{Total number of guesses}}
\]

\[
\text{Recall} = \frac{\text{Number of slots correctly filled}}{\text{Total number of correct guesses}}
\]

Table 2: Label frequency in noisy training data.

<table>
<thead>
<tr>
<th>Label</th>
<th>Frequency</th>
<th>Named Entity Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIL</td>
<td>19196</td>
<td></td>
</tr>
<tr>
<td>Crash Site</td>
<td>10365</td>
<td>LOCATION</td>
</tr>
<tr>
<td>Operator</td>
<td>4869</td>
<td>ORGANIZATION</td>
</tr>
<tr>
<td>Fatalities</td>
<td>2241</td>
<td>NUMBER</td>
</tr>
<tr>
<td>Aircraft Type</td>
<td>1028</td>
<td>ORGANIZATION</td>
</tr>
<tr>
<td>Crew</td>
<td>470</td>
<td>NUMBER</td>
</tr>
<tr>
<td>Survivors</td>
<td>143</td>
<td>NUMBER</td>
</tr>
<tr>
<td>Passengers</td>
<td>121</td>
<td>NUMBER</td>
</tr>
<tr>
<td>Injuries</td>
<td>0</td>
<td>NUMBER</td>
</tr>
</tbody>
</table>

Table 3: Feature sets for mention classification.

<table>
<thead>
<tr>
<th>Feature set</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maj. Baseline</td>
<td>0.026</td>
<td>0.237</td>
<td>0.047</td>
</tr>
<tr>
<td>Local Classifier</td>
<td>0.158</td>
<td>0.394</td>
<td>0.218</td>
</tr>
</tbody>
</table>

Table 4: Performance of local classifier vs. baseline.

3.2. Experiment 2: Sequence Model with Local Inference

The local model just presented fails to capture dependencies between mention labels. For example, ⟨Crew⟩ and ⟨Passenger⟩ go together; ⟨Site⟩ often follows ⟨Site⟩; and ⟨Fatalities⟩ never follows ⟨Fatalities⟩:

- 4 crew and 200 passengers were on board.
- The plane crash landed in Beijing, China.
- *20 died and 30 were killed in last Wednesday’s crash.

In this experiment, we compare our simple local model to a sequence model with local inference (SMLI). We implement SMLI using a maximum entropy markov model (MEMM) approach. In the local model, mentions in a sentence are classified independently. In contrast, at each step

---

6Parsing, POS tagging, and NER: Stanford Core NLP. nlp.stanford.edu/software/corenlp.shtml

7LiD features.

8Named Entity Features: Unigrams and part-of-speech tags within the named entity mention, the number of tokens in the mention, and the named entity type of the mention.

9Local Context: Unigrams and part-of-speech tags within five tokens of the named entity mention, with specific features for one, two, and three tokens before and after the mention.

10Sentence Context: Unigrams and part-of-speech tags in the same sentence as the target named entity.

11Dependency Features: Incoming and outgoing dependency arcs, lexicalized and unlexicalized.

12Location in document: Is the target named entity mention in the first, second, third, or fourth quarter of the document?
in SMLI, the label of the previous non-NIL mention is used as a feature for the current mention. At training time, this is the previous non-NIL mention’s noisy “gold” label. At test time, this is the classifier’s output on the previous non-NIL mention.

Table 6 shows test-set results. SMLI boosted recall with only a slight decrease in precision. The difference in recall was statistically significant (p < 0.05). Qualitative analysis of SMLI’s feature weights revealed that the classifier learned the patterns mentioned above, as well as others.

### 3.3. Experiment 3: Noisy-OR Aggregation

So far we have assumed exhaustive label aggregation—as long as at least one mention of a particular value gets a particular slot label, we use that value in our final slot-filling decision. For example, if three mentions of *Mississippi* receive the labels ⟨Crash Site⟩, ⟨Operator⟩, and NIL, then the final slot-fills are Crash Site = *Mississippi* and Operator = *Mississippi*. Intuitively, this approach is suboptimal, especially in a noisy data environment where we are more likely to misclassify the occasional mention. In fact, a proper aggregation scheme can act as fortification against noise induced misclassifications.

With this in mind, we adopted Noisy-OR aggregation. The key idea is that classifiers give us distributions over labels, not just hard assignments. A simplified example is given below for two mentions of *Stockholm*.

- Stockholm ⟨NIL:0.8; Crash Site: 0.1, Crew:0.01, etc.⟩
- Stockholm ⟨Crash Site: 0.5; NIL: 0.3, Crew:0.1, etc.⟩

Given a distribution over labels ℓ for each mention m in M (the set of mentions for a particular candidate value), we can compute Noisy-OR for each label as follows.

\[
\text{Noisy-OR}(ℓ) = Pr(ℓ|M) = 1 - \prod_{m \in M} (1 - Pr(ℓ|m))
\]

In the *Stockholm* example above, the Noisy-OR for ⟨Crash Site⟩ and ⟨Crew⟩ are 0.95 and 0.11 respectively. A value is accepted as a slot filler only if the Noisy-OR of the slot label is above a fixed threshold. We found 0.9 to be an optimal threshold by cross-validation on the training set.

Table 7 shows test-set results comparing Noisy-OR and exhaustive aggregation on the local and SMLI classifiers. We see that Noisy-OR improves precision while decreasing recall. This is expected because Noisy-OR is strictly more conservative (NIL-prefering) than exhaustive aggregation. In terms of F₁ score, Noisy-OR aggregation is significantly better at p < 0.1 for the local model and p < 0.05 for the SMLI.

### 3.4. Experiment 4: Joint Models

In the previous two experiments, SMLI had better recall than our local model, but overall improvement was modest. One possible explanation comes from an error propagation problem endemic to this class of models. Consider the example in Figure 1. At training time, *USAirways* has the feature PREV-LABEL-INJURY. But suppose that at inference time, we mislabel J5 as ⟨Survivors⟩. Now *USAirways* has the feature PREV-LABEL-SURVIVOR, and we are in a feature space that we never saw in training. Thus we are liable to make the wrong classification for *USAirways*. And if we make the wrong decision there, then again we are in an unfamiliar feature space for *USAirways*. And if we make the wrong decision there, then we are in an unfamiliar feature space for *Boeing 747* which may lead to another incorrect decision.

This error propagation is particularly worrisome in our distant supervision setting due to the high amount of noise in the training data. To extend the example, suppose instead that at distant supervision time, J5 was given the incorrect “gold” label (Fatalities). Now at test time, we might correctly classify J5 as ⟨Injuries⟩, but this will put us in an unseen feature space for subsequent decisions because *USAirways* saw ⟨Fatalities⟩ at training time, not ⟨Injuries⟩. An ideal solution to this error propagation problem should do two things. First, it should allow suboptimal local decisions that lead to optimal global decisions. For the previous example, this means that our choice for J5 should take into account our future performance on *USAirways* and *Boeing 747*. Second, models of sequence information should be based on actual classifier output, not gold labels. This way we are not in an unfamiliar feature space each time our decision differs from the gold label.

In essence, we want a joint mention model—one which optimizes an entire sequence of mentions jointly rather than one at a time. To this end, we tested two joint models: i) a linear-chain CRF, and ii) the Searn algorithm (Daumé III, 2006). The following sections describe our implementation of these models and experimental results.

### 3.4.1. Linear-Chain CRF Model

Conditional random fields (CRFs) provide a natural language for joint modeling of sequences of mentions and their associated labels (Lafferty et al., 2001). CRFs are particularly well-suited to classification because they are discrim-

---

7 All significance tests reported in this paper were computed using bootstrap resampling on test events with 10,000 trials.

8 Implemented using Factorie (McCallum et al., 2009)
inative models, i.e. they do not involve modeling dependencies among mention features. For a sequence of mentions $m$ and associated labels $l$, the conditional probability is given as

$$P(l|m) = \frac{1}{Z(m)} \prod_{i} \exp \left\{ \sum_{j} \lambda_{ij} f_{ij}(l, m) \right\}$$

where we have introduced a set of factors $\{\Psi_{i}\}$, weights $\{\lambda_{ij}\}$, and features $\{f_{ij}\}$. We specialize to a linear-chain CRF with three factors: $\Psi_{L}$, $\Psi_{LL}$, and $\Psi_{ML}$ (see Figure 2). The first factor captures label dependencies, the second captures dependencies between labels of adjacent mentions, and the third captures dependencies between labels and mention features.

Learning proceeds via stochastic gradient descent in conjunction with the max-product algorithm, which is also used during inference. Parameter updates are made using confidence weighting with a learning rate of unity. Hyperparameters are chosen by maximizing the F1 score on a dev set, i.e. after Noisy-OR aggregation, which resulted in the following choices. Learning stops after $n_T = 4$ rounds. In Eqn. Noisy-OR, only the $n_{TOP} = 3$ most probable mentions enter the product for any given label. Finally, the $\Psi_{ML}$ weights corresponding to NIL were reduced by a multiplicative factor $x = 1.7$ to prevent too many NIL labels at inference.

### 3.4.2. S E A R N Model

For our second joint model, we use the S E A R N algorithm to infuse global decisions into a sequence tagger. S E A R N is a general framework for training classifiers which make globally optimized choices in a structured prediction task (Daumé III, 2006). In our setting, S E A R N generates a model in which a mention’s label depends not only on its features and the previous non-NIL label, but also on the impact of this label for subsequent decisions later in the sentence.

The algorithm operates by associating training mentions with cost-vectors corresponding to the global, sequence-wide impact of different label choices. These mentions and cost-vectors are passed to a cost-sensitive classifier for learning. In our implementation, we follow Vlachos and Craven (2011) in using the cost-sensitive classifier described in Crammer et al. (2006), which amounts to a passive-aggressive multiclass perceptron.

Inherent in this setup is the following chicken-and-egg problem: we want to train an optimal classifier based on a set of global costs, but we would like these global costs to be computed from the decisions made by an optimal classifier. S E A R N gives an iterative solution to this problem. Algorithm 1 illustrates the basic framework. The algorithm is seeded with an initial policy based on gold labels (akin to our local sequence model, which uses gold labels for previous-label features during training). At each iteration, a new policy is learned from a cost-sensitive classifier and interpolated with previous policies.

S E A R N has a number of hyperparameters. By cross-validation on the training set, we arrived at the following settings: 4 S E A R N iterations; 8 perceptron epochs per iteration; interpolation $\beta = 0.3$; perceptron aggressiveness $= 1.0$.

### 3.4.3. Joint Model Results

The test-set results comparing these joint models to SMLI and our local model are shown in Table 8. All results use Noisy-OR aggregation. Our S E A R N model outperformed the other models in precision and F1 score ($p < 0.15$). The S E A R N algorithm was able to model the inter-mention dependencies described in Section 3.2 while avoiding the error propagation problem affecting SMLI.

Our CRF model was able to learn useful weights for label pairs. For example, it learned a high positive weight for ‘Passengers, Crew’ and a low negative weight for ‘Fatalities, Fatalities’. However, performance did not improve over our non-joint models. One explanation for this
comes from a key structural difference between our CRF model and our SERN and Pipeline models. In our CRF model, edges connect adjacent named entities. In both SERN and SMLI, the dependency is with the previous non-NIL named entity, ignoring any NIL labels that intervene. This means the latter two models are more directly sensitive to non-NIL labelings much earlier in the sentence. The lack of a non-NIL label early in a sentence turns out to be a strong signal that the sentence is not relevant to the plane crash domain. Without this signal, the CRF classifier frequently makes false-positive mislabelings in irrelevant sentences, e.g. assigning (Site) to a location not related to the crash. In general, the CRF model assigned labels more liberally than the other models, leading to high recall, but lower precision.

### 3.5. Experiment 5: Feature Ablation

In this final experiment, we conducted a features ablation study to explore the impact of different input features. Our models use five types of features as described in Table 3. Table 9 shows the performance of our SERN model as feature sets are removed (without retuning hyperparameters). Performance actually increases as location in document (LiD) features are removed, but this result is not statistically significant. Removing dependency (Dep) features causes a significant drop in F₁ score (p < 0.1). Removing sentence context (SCon) features causes a less significant drop (p = 0.16). Finally, removing local context (LCon) features causes a major decrease in performance (p < 0.01).

### 4. Conclusion

This paper has presented a preliminary study of distant supervision applied to event extraction. We described a new publicly available dataset and extraction task based on plane crash events from Wikipedia infoboxes and newswire text. We presented five experiments. In the first experiment, we showed that a simple local classifier with a rich set of textual features outperforms a naive baseline, despite having access only to noisy, automatically generated training data. In the second experiment, we extended our approach to a sequence tagging model with local inference, showing that by considering previous label decisions as features, recall improves. In our third experiment, we demonstrated the effectiveness of a Noisy-OR model for label aggregation. In experiment four, we evaluated two models which apply joint inference to the sequence labeling task. Our linear-chain CRF model learned reasonable weights and improved recall, but overall performance suffered. Our second joint model, based on the SERN algorithm, performed best, with considerable boost to both precision and F₁ score. Lastly, with a post-hoc ablation experiment, we showed that syntactic information and local context are both important for model success.

### Acknowledgements

We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) contract no. FA8750-13-2-0040. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

### 5. References


