Event extraction using iterative optimization

Aju Thalappillil Scaria ajuts@stanford.edu
Rishita Anubhai rishita@stanford.edu
Rose Marie Philip rosep@stanford.edu

Abstract

The web has an enormous amount of information stored as text content. Designing automated mechanisms to create knowledge bases using the content of the web and building automatic question answering systems have been an area of research for many years. In this project, we aim to build a system that can extract events, their associated entities and assign semantic roles to the entities that are not specific to any domain. We achieve this by first predicting event triggers and using these triggers to predict their arguments. Since the models are not joint, there are cascading errors that result in low recall. We overcome this by a technique called iterative optimization which makes use of the fact that identifying entities in a sentence can help us predict event triggers better. Thus, we predict entities associated with triggers and also predict triggers from the entities identified iteratively till we reach an optimal stage.

1 Introduction

The World Wide Web is accessible to anyone with an internet connection. But analyzing this vast amount of data to extract meaningful information is rather difficult because of its sheer volume. There have been a lot of research focusing on extracting content from the web to build large knowledge bases. For instance, extracting events and the associated entities from biomedical and molecular biology text has drawn a lot of attention, especially because of the introduction of BIONLP tasks targeting fine-grained information extraction. The knowledge bases built by information extraction systems can find application in a variety of fields, for example, to supplement the structured information available to build quiesting answering systems.

In this paper, we focus on the task of event and entity extraction, with semantic role labeling. Most of the prior work either focuses on subsets of these tasks or look at very specific types of events or entities that are to be extracted. For instance, Toutanova et. al. focuses on entity extraction and semantic role labeing assuming that the event triggeres are known, while McClosky focuses on event and argument extraction for only binding, regulation and phosphorylation event types. These works show that the task of extracting information from text is possible with reasonable accuracy level. But, the performance of the state-of-the-art systems is still quite distant to that of a perfect extraction system. This is mostly because interpreting meaning from complex structures of natural language is a hard task. In this work, we combine the representations, feature extraction methods and models adopted in the previous work.

2 Previous Work

Most of the prevoius work dealt only with a subset of the three tasks listed as the project goals and dealt with very specific domains. For instance, Chambers and Jurafsky (2008) talks about event identification and temporal ordering using narrative event chains.Toutanova et al. (2005) comes up with a joint model for argument extraction and semantic role labeling assuming that the trigger words are provided. Bjorne et al. (2009) solves the problem of event extraction and semantic role labeling assuming the entities are known. Also, their event categories and arguments are closely tied to the BIONLP task and hence, would not generalize to domain-independent text. Chambers and Jurafsky (2009) addresses the area of event identification, and also combines argument extraction with semantic role labeling using an unsupervised approach addressing narrative event chains in further

detail.

Different approaches have been tried to come up with a solution to these problems. While some work have used unsupervised learning methods (Chambers and Jurafsky, 2008), (Chambers and Jurafsky, 2009), there are others who viewed each task as a separate machine learning problem (Bjorne et al., 2009) or tried to combine the problems using joint models that can overcome the problem of cascading errors and does not assume independence that separate classifiers for event and enitity extraction do (Toutanova et al., 2005), (Riedel and McCallum, 2008).

Events and entities have been represented in different ways; the work of Bjorne et al. (2009) and Riedel and McCallum (2008) use graphs to encode event-entity and event-event relationships and looks at the problem of joint event extraction as a structure prediction problem. Toutanova et al. (2005) and McClosky et al. (2011), use tree structure to represent the nodes and use semantic role labeling to solve this problem. Also, some of the work rely on the constituency parse structure of sentences (Toutanova et al., 2005), while others (Bjorne et al., 2009), (McClosky et al., 2011) use the dependency parse structure for finding features.

There has also been work on temporal ordering of events thus extracted. The temporal relation between events can be considered as a classification problem, the number of classes depending on the number of relations being considered in the task (2008), (2008-temporal ordering). McClosky et al. (2011) uses a timeline model in addition to this that allows events to be associated with precise time intervals, which improves human interpretability and also simplifies the global inference formulations required to solve this task.

3 Data and Evaluation

In this project, the dataset was prepared by annotating 125 paragraphs from different chapters of the text book *Biology (Eighth Edition)* by *Neil A. Campbell* and *Jane B. Reece*. Each paragraph is a text file and has an associated annotation file that indicates the different events and entities (by their character offsets in the original paragraph) and the event-entity and event-event relationships. The annotations were done by experts in the field (employees of the company Vulcan). Since there is not much data at our disposal, we split the data by a proportion of 70-30% for training and testing. We randomly permute the order of files to avoid similarities in adjacent files and then use 10 fold cross validation on the training set.

For event prediction, we use F1 score based on the trigger predictions made. In entity prediction, the F1 score is based on whether an entity was predicted correctly along with its association with the corresponding event. In addition, our evaluation of the entity prediction model is very conservative as we consider only an exact match with the whole text of the entity to be a correct prediction. Entities typically span many words and the text chosen to represent an entity is subjective. For instance, the question whether 'plants in our original population' or 'plants' is to be marked as an entity does not have a clear answer. For semantic role labeling, we measure the F1 score based on the role predicted for entity-event pairs.

4 Models

In this section, we present the models we developed. We combine the learnings from the different approaches adoped in prior literature to build a model that can be used for event and entity extraction not specific to any domain. Even though our dataset is based on paragraphs from a biology textbook, we believe our models would generalize well to deal with more general content as our features and event/entity classes are not tied to the biological domain in any way. Our system takes a paragraph of text as input and does the following tasks:

1. Identify the events by locating the trigger words.
2. For each event, identify entities that are its arguments.
3. For each argument that is identified with a specific event, assign a semantic role.

We model events and entities as sub-trees headed by a node in the constituency tree. Each sentence is assumed to be independent of each other as far as entity-event relationships are concerned. Events are denoted by their trigger words and are hence pre-terminals in the parse tree. Entities are denoted by a parse tree-node covering a sub-tree that spans over the whole text of the entity. In some cases when there is no single node that covers the entire entity (mostly because

of parser errors, for e.g., PP attachment), we use an approximation by repeatedly removing tokens from the end or beginning of the span of text to identify a node that covers it. We manually verified that this heuristic works well in practice and results in entities that convey almost the full meaning of original span, and are well-formed.

In our project, we use Stanford Core NLP tools. We use the annotation pipeline available in the toolkit including tokenization, lemmatization, dependency and constituency parsers, and POS taggers. The events, entities and their relationships are represented as annotations on the already existing sentence annotations, by implementing the CoreAnnotation interface. This helps us to integrate our codebase with the existing features of the CoreNLP toolkit.

For all the classification tasks, we use maximum entropy model based on an implementation of L-BFGS for Quasi Newton unconstrained minimization. We use features based on the dependency graph of the sentence in conjunction with the constituency parse, since it contains information about the dependencies between tokens, which are critical in identifying event-entity relations. This is done by identifying the position of the headword of an entity or event from the dependency tree and analyzing the property of the head word. We use Collins head finding algorithm for finding the head word of a parse tree node.

First, we present our model for task 1 which is an independent classification task. Then, we talk about task 2 where we identify entities that are arguments to event triggers. Since the information about entities can be used to improve event trigger prediction, we then cover the iterative optimization algorithm we developed that iteratively predicts event triggers from entities and entities from event triggers. Then, we talk about the joint reranking model we developed for semantic role labeling.

4.1 Event trigger prediction

As we mentioned earlier, events are represented as pre-terminal nodes in the parse tree of a sentence. Let N_{pt} be the set of all n_{pt} pre-terminal parse tree nodes $\{e_1, e_2, ...e_{n_{pt}}\}$. The trigger prediction algorithm generates a list of triggers in the sentence, $T : N_{pt} \rightarrow \{0,1\}^{n_{pt}}$ that indicates if a word is a trigger or not. Formally, our goal is to maximize the likelihood of the triggers $P(T|x)$ given the sentence, where x is the tokens within the sentence.

The baseline model predicted every pre-terminal node whose part-of-speech tag started with 'VB' to be an event trigger. As the next step, we designed a MaxEnt model that trained on the annotated samples using several lexical, dependency tree based and parse tree based features. The mode is local, in the sense that it predicts if a pre-terminal node is an event trigger independent of other predictions. Hence,

$$P_T(T|x) = \prod_{e_i \in N_{pt}} P_T(e_i|x)$$

The features we started with for event trigger prediction were part-of-speech tag of the word, its lemma, the path from root to the node in parse tree and the label of the outgoing edges from the node in the dependency graph. On doing initial error analysis, we found that our classifier failed to identify all nominalized verb forms as event triggers. We used NomLex and NomBank, which are collections of nominalizations, to include a feature that indicates nominalization. NomLex proved useful and the classification accuracy improved by around 2-3%. But, since NomLex was not an extensive dictionary, some other nominalizations were still misclassified. So, we also used WordNet derivations to replace any nominalized verbs with its actual verb form and this improved the performance by another 2%.

4.2 Argument identification for triggers

An entity is represented as a node in the parse tree spanning over the full text of the entity along the leaves of the tree. The fact that there is more than one event in almost all sentences makes our task of event-entity association harder. This is because, instead of just predicting a node in the parse tree as an entity, we have to predict if a node is associated with a specific trigger from task 1. To overcome this, we assign a probability for each node in the parse tree to be an entity associated with each event. If N is the set of n nodes $\{t_1, t_2, ...t_n\}$ in the parse tree, given an event trigger e, our goal is to predict the best set of entities A for e.

As a first step, we built a baseline model that predicts a node in the parse tree as an argument to an event trigger, if it is of part-of-speech tag 'NP' and if the head word of the node in the parse tree is a child of the event trigger in the dependency tree of the sentence. We used Collins head finder algo-

rithm to identify the headword of a parse tree node. We then implemented a MaxEnt based model using more features between the event triggers and the candidate nodes to classify the node as either an argument or $NONE$ if not. This is also a local model, hence

$P_A(A|e,x) = \prod_{t_i \in N} P_A(t_i|e,x)$

The features we use for finding event arguments are combinations of features of both event node and the candidate entity node. This includes POS tag of candidate + POS tag of event trigger, headword of candidate + POS tag of event trigger, path from the candidate to the event trigger and an indicator feature denoting whether the headword of the candidate is a child of the trigger in the dependency tree. The model assigns a probability value for each node in the parse tree to be an argument to a specific event trigger.

Dynamic Program for non-overlapping constraint. Since we predict a node in the parse tree as an argument to an event trigger or not, there are instances when predicted entities overlap. For instance, a sub-tree of a tree node already marked as an entity may also be tagged as an entity. To avoid this, we devised a bottom-up dynamic program that tags a node as entity or not, looking at the probability of the node and its immediate children being entities. The dynamic program works from the pre-terminal nodes of the tree and find best assignments for each tree using the already computed assignments for its children. This ensures that a sub-tree that is a part of A in itself doesn't have smaller subtrees that belongs to A. The most likely assignment for a tree t to A or $NONE$ is the one that corresponds to the maximum of:

1. The sum of the log-probabilities of the most likely assignments of the children sub-trees $t_1, t_2, ..t_k$, plus the log-probability for assigning the node t to $NONE$.

2. The sum of the log-probabilities for assigning all of t_i's nodes to $NONE$ plus the log-probability of assigning the node t to A.

Another addition we did to the dynamic program was that an entity node cannot subsume a node in the parse tree that is an event trigger.

4.3 Iterative Optimization Algorithm

One of the drawbacks of our approach to event trigger and argument prediction was that the models were independent. As a result, any errors made in trigger prediction will cascade and affect the argument identification phase. For instance, if an event trigger is not predicted, we are going to miss all the arguments predictions for the trigger as well. At the same time, any incorrect event triggers predicted may result in lots of event-argument pairs being predicted that do not exist. Even after we used NomLex to find nominalizations, our classifier missed predicting some of the triggers. But, if we knew that there were entities that were children of the word in the dependency graph, then it gives more evidence that the word is an event trigger. At the same time if there are words that we might have predicted as triggers that do not have any entities as children in the dependency tree, they are less likely to be even triggers. Hence, we hypothesized that knowledge about entities in the sentence can help in event trigger prediction as we can use more features from the dependency tree. To test this, we used the gold entities from a sentence to add more features for event trigger prediction and we found that it gave us an F1 score boost from 0.67 to 0.82. Since gold entities are not available during the test phase, we designed a separate model that predicts entities in a sentence independent of the triggers. As before, entities are sub-trees under a parse tree node and the model predicts the probability of a node to be an entity or not. We used the same dynamic programming approach for non-overlapping constraints of entities.

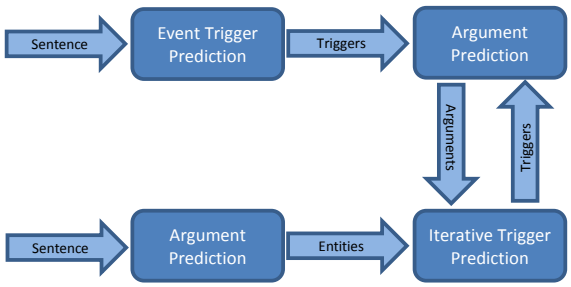

Figure 1: Stages in iterative optimization algorithm

The different stages involved in the iterative optimization algorithm is depicted in Figure 1. We use the event triggers predicted to predict its arguments. Now, we combine the set of entities predicted as arguments, with those predicted by the independent argument prediction model to create the set of all entities and is used in trigger prediction. This model predicts

$P_{IO}(T|x,A) = \prod_{e_i \in N_{pt}} P_{IO}(e_i|x,A)$, where A is the set of all entities in the sentence.

Once this models predicts the event triggers, we use the same model $P_A(A|e,x)$ to predict arguments associtated with each event trigger that satisfy the non-overlapping constraint. Hence, $P_{IO}(A|x,e) = P_A(A|x,e)$. We repeat the iterative trigger and argument prediction till there is no improvement in f1 score.

TODO- Describe the featuers we used.

4.4 Semantic Role Labeling

The baseline model predicts the most frequent semantic role as the role of an entity. For argument identification, we had used a binary classifier based on MaxEnt model that predicts a probabilty value it to be an argument. For semantic role labeling, we extended this model to a multiclass classifier to generate probability values for a parse-tree node to belong to the different semantic roles. We also modified the dynamic program for non-overlapping constraint used in argument prediction to a re-ranking model that jointly assigns semantic role to all nodes in a sub-tree. We also combine the probability values generated by the P_A model while predicting the semantic roles for parse tree nodes. The values that we use correspond to probabilities generated in the last stage of argument prediction in the iterative optimization algorithm. Let L be the semantic role labels that have been assigned to the entities, including $NONE$ if it is not an entity.

$P_{SRL}(L|e,x) = P_A(A|e,x)P_L(L|e,x)$

We use a bottom up re-ranking approach by keeping the top-k joint assignment of semantic roles to all nodes in a sub-tree. This algorithm is similar to the dynamic programming for non-overlapping constraints. At the pre-terminal nodes, we keep just the semantic roles of the word subsumsed by the node in descending order of probability. At nodes above the pre-terminal nodes, there are two scenarios:

1. The node is an argument to the trigger: In this case, the node has a non-NONE semantic role and none of the children nodes can be entities and hence all children would have semantic role $NONE$. For each non-NONE semantic role of the node, we compute the probability value of the joint labeling of the sub-tree.

2. The node is not an argument to the trigger: In this case, the node has a semantic role of $NONE$. Since at each of the child nodes we have top-k possible assignements, we take all combinations of assignments of children nodes and compute the probability for each of these possible joint labelings.

The next step is to re-rank all the possible joint assignments possible at the node and then retain only the top-k at the node. This algorithm proceeds until we reach the root and the joint labeling that has the highest probability will be the semantic roles predicted by the model.

5 Results

5.1 Event trigger prediction

The results for event prediction tasks in train and test sets are presented in Table 1. The results on train set is based on 10 fold cross validation of training data. For test result, we trained on the whole train set and then tested on the test set. As seen in the results table, the baseline model that predicted every verb as an event trigger performed quite well indicating that most of the eevnt triggers were verbs. The basic MaxEnt model for trigger prediction gave a good improvement over the baseline. The iterative optimization algorithm, after 1 iteration gave an F1 score of 0.712 and 0.646 respectively on the train and test sets and these were the best results we obtained. This clearly indicates that knowldge of entities can help improve the prediction of event triggers. We also note that the increase in F1 score is contributed by both increase in precision and recall. This is expected as we are now able to weed out words that are not event triggers as they don't have any children in the dependency tree. At the same time, we are able to predict more event triggers when we know that there are entities that are its children in the dependency tree.

5.2 Entity prediction

The results for argument prediction are presented in Table 2. The baseline model intuitively captures the relation between event triggers and its arguments as it gave an F1 score of 0.505 in comparison to the simplicity of the approach. The basic version of the MaxEnt model gave good improvements over the baseline model, but was much lesser than what we had expected. So, we ran an oracle experiment that assumed all the gold event triggers are known, which helped us to isolate the

	Train			Test		
	P	**R**	**F1**	**P**	**R**	**F1**
Baseline	0.426	0.664	0.517	0.396	0.644	0.491
MaxEnt_Basic	0.704	0.667	0.681	0.656	0.606	0.630
MaxEnt_IO	0.740	0.692	0.712	0.676	0.619	0.646
MaxEnt_IO_Gen	0.	0.	0.	0.	0.	0.

Table 1: Event trigger prediction

	Train			Test		
	P	**R**	**F1**	**P**	**R**	**F1**
MaxEnt_Oracle	0.727	0.540	0.618	0.754	0.608	0.673
Baseline	0.447	0.512	0.474	0.443	0.503	0.471
MaxEnt_Basic	0.590	0.431	0.495	0.577	0.462	0.513
MaxEnt_IO	0.568	0.471	0.513	0.553	0.494	0.522
MaxEnt_IO_Gen	0.	0.	0.	0.	0.	0.

Table 2: Entity prediction for event triggers

performance of the argument prediction module. The score got boosted to 0.513 and 0.522 in train and test set respectively, after we implemented the iterative optimization algorithm. Again, this was the best performing model. The improvement in F1 score is mainly because we are now able to predict event triggers better. But, it is surprising that the precision dropped after we performed iterative optimization, and the increase in F1 score was due to the large improvement in recall. We analyze this in the next section.

5.3 Semantic role labeling

The results for semantic role labeling are presented in Table 3.

TODO - Fill the section for SRL

6 Analysis

In section, we analyze the results in detail and perform error analysis.

6.1 Event trigger prediction

Most of the event triggers are verbs which makes it easy to achieve F1 scores of around 0.6 without many features. But, including features to classify nominalized verbs that are event triggers is a hard task. Especially because there is no single source of word nominalizations that is perfect. Some are very general and include a lot of words that are not exactly word triggers because of which precision is affected, while others have very few and doesn't improve recall much.

TODO - IO - what all does it fix? Lexical vs non-lexical features?

6.2 Entity prediction

We saw that the task of argument prediction is much harder because of the lower F1 score mainly because of two reasons. Firstly, non-overlapping which we fixed. Word boundaries subjective. Sharing of triggers and we score event-entity combinations.

The dynamic programming approach gave us a boost of 0.04 (0.60 to 0.64) in F1 score.

TODO- IO - what all does it fix?

Lexical vs non-lexical features

6.3 Semantic role labeling

7 Conclusion

This section is for conclusion

Acknowledgments

Do not number the acknowledgment section. Do not include this section when submitting your paper for review.

References

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08*.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised Learning of Narrative Schemas and their Participants In *Proceedings of ACL-09*.

	Train			Test		
	P	**R**	**F1**	**P**	**R**	**F1**
Baseline	0.	0.	0.	0.	0.	0.
MaxEnt_Oracle	0.	0.	0.	0.	0.	0.
MaxEnt_Basic	0.	0.	0.	0.	0.	0.
MaxEnt_IO	0.	0.	0.	0.	0.	0.
MaxEnt_IO_Gen	0.	0.	0.	0.	0.	0.

Table 3: Semantic role labeling

Nathanael Chambers and Dan Jurafsky 2011. Template-Based Information Extraction without the Templates In *Proceedings of ACL-11*.

Jari Bjorne, Juho Heimonen, Filip Ginter, Antti Airola, Tapio Pahikkala, Tapio Salakoski 2009. Extracting Complex Biological Events with Rich Graph-Based Feature Sets. In *Proceedings of the Workshop on BioNLP: Shared Task,*.

Kristina Toutanova, Aria Haghighi, Christopher D. Manning 2005. Joint Learning Improves Semantic Role Labeling. In *Proceedings of ACL 2005*.

Sebastian Riedel Andrew McCallum 2008. Fast and Robust Joint Models for Biomedical Event Extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*.

Makoto Miwa, Rune Saetre, Jin-Dong D. Kim, and Junichi Tsujii 2010c. Event extraction with complex event classification using rich features In *Journal of bioinformatics and computational biology*.

David McClosky, Mihai Surdeanu, and Christopher D. Manning 2011. Event Extraction as Dependency Parsing In *Proceedings of ACL-11*.

Nathanael Chambers and Dan Jurafsky 2008. Jointly Combining Implicit Constraints Improves Temporal Ordering In *EMNLP-08*.

Quang Xuan Do, Wei Lu, Dan Roth 2012. Joint Inference for Event Timeline Construction In *EMNLP-12*.