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 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. 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. After this, we used a top-k re-ranking model for semantic role labeling. We also show that the models are not specific to any domain and generalize well to predict events and entites on news headlines and articles on the web.

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 difficult because of its sheer volume. Extracting content from the web to build knowledge-bases is an active area of research. 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 question answering systems.

In this paper, we focus on the task of event and entity extraction, with semantic role labeling. The prior literature in this field shows that the task of extracting information from text is possible with reasonable accuracy levels. But, the performance of the state-of-the-art systems is still quite distant from that of a perfect extraction system. This is mostly because interpreting the meaning of complex structures in natural language is a hard task. In this work, we combine the representations, feature extraction methods and models adopted in the previous work. We also introduce an iterative method to combine event and entity extraction that can benefit from each other which leads to a model that is superior to the independent models. This reduces the cascading errors that are common in independent models. Also, the semantic role labeling task identifies entities and assigns semantic roles for each event trigger identified.

2 Previous Work

Most of the previous work addresses a subset of tasks we focus on in this paper or look at very specific types of events or entities to be extracted. For instance, Toutanova et al. (2005) comes up with a joint model for argument extraction and semantic role labeling assuming that the trigger words are provided while McClosky et al. (2011) focuses on event and argument extraction for only binding, regulation and phosphorylation event types. 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 adopted for event and entity extraction. While some of the previous work has 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 entity 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) uses 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 relies 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.

To test if the models and algorithms we developed generalize to text content on the web, we built a general dataset containing 5 paragraphs. We included data from Wikipedia articles, New York Times news headlines and news articles. The articles and headlines were chosen to cover main topics like general news, politics, world news and sports. We manually annotated the data for event triggers and entities that were its arguments.

4 Models

In this section, we present the models we developed. We combine the different approaches adopted in prior literature to build a model that can be used for event and entity extraction not specific to any domain. We demonstrate this by testing our models on the general dataset, the results of which are presented in the next section. 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. Assign semantic roles like Result, Agent, Theme, Location etc. to event-entity pairs identified.

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 event-entity 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, there is no single node that covers the entire entity (mostly because of parser errors, for e.g., PP attachment). In this case, we use an approximation to mark entities, by first repeatedly removing tokens from the end of the span to find a single subtree that covers the full text, otherwise, we repeat the same process from the beginning of the span of text. 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 described in this paper, 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. The features are extracted by identifying the position of the headword of an entity or event from the dependency tree and analyzing the properties 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 $\{e_1, e_2, ...e_{n_{pt}}\}$ and let TRIG be the set of triggers. The trigger prediction algorithm generates a mapping T of all pre-terminal nodes in the tree to TRIG or NONE, $T : N_{pt} \rightarrow \{0, 1\}^{n_{pt}}$. Formally, our goal is to maximize the likelihood of the labeling T, $P(T|x)$, given the sentence, where x is the tokens within the sentence.

We trained a MaxEnt model on the annotated samples using several lexical, dependency tree based and parse tree based features. The model is local, in the sense that it predicts if a pre-terminal node is an event trigger independent of other predictions. Hence,

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

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. In the initial error analysis, we found that our classifier failed to identify many nominalized verb forms as event triggers. So we used NomLex to add a feature to identify nominalization. We also used WordNet derivations to replace any nominalized verbs with its actual verb.

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 labeling, A of parse tree nodes to ARG or $NONE$ for the event trigger e.

We implemented a MaxEnt based model using more features between the event triggers and the candidate nodes to classify the node as either an

ARG or $NONE$ if not. This is also a local model, hence we try to maximize

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

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. This is motivated by the work of Toutanova et al. (2005). The dynamic program works from the pre-terminal nodes of the tree and finds best assignments for each sub-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 argument 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 a boost in F1 score from 0.68 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 head word of the span covered by the node, its POS tag, its parents POS tag, the path to the node in the parse tree and the CFG rule expansion of its parent as features. We used the same dynamic programming approach for non-overlapping constraints of entities.

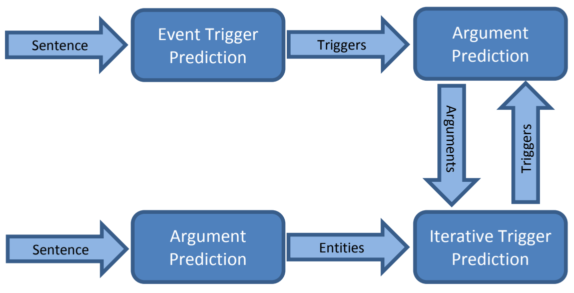

Figure 1: Stages in iterative optimization algorithm

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 the most likely labeling T that maximizes

$P(T|x, A) = \prod_{e_i \in N_{pt}} P(e_i \in TRIG|x, A)$, where $TRIG$ and A are as defined earlier.

Once this model predicts the event triggers, we use the same model $P(A|e, x)$ to predict arguments associated with each event trigger that satisfy the non-overlapping constraint. We repeat the iterative trigger and argument prediction till there is no improvement in F1 score.

The mode $P(T|x, A)$ uses the path to entities in parse tree, path length to the nearest entity, the actual path to nearest entity and the number of words between closest entity before and after the node as features.

4.4 Semantic Role Labeling

For argument identification, we had used a binary classifier based on MaxEnt model that predicts a probability value for it to be an argument. For semantic role labeling, we extended this model to a multi-class classifier to generate probability values for a parse-tree node being assigned to a specific semantic role. 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. Let L be a labeling of semantic roles to the parse tree nodes, including $NONE$ if it is not an entity. Our goal is to maximize the probability of the labeling we generate.

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 assignments, we take all combinations of assignments of children nodes and compute the probability for each of these possible joint labeling.

The next step is to re-rank these possible joint assignments 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. We are currently using k=5 in our 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. The baseline model predicted every pre-terminal node whose part-of-speech tag started with 'VB' to be an event trigger. As seen in the results table, the baseline model performed quite well, indicating that most of the event triggers were verbs. The basic MaxEnt model for trigger prediction gave a good improvement over the baseline. The iterative optimization algorithm, converged after 2 iterations and 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 knowledge 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. We also show the results of the iterative optimization algorithm on the general dataset we built.

5.2 Entity prediction

The results for argument prediction are presented in Table 2. We built a baseline model that predicts a node in the parse tree as an argument to an event trigger if it is of POS 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. The baseline model intuitively captures the relation between event triggers and its arguments as it gave

	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	-	-	-	0.792	0.679	0.731

Table 1: Results of event trigger prediction. MaxEnt_Basic is the basic model and MaxEnt_IO is the model that uses iterative optimization. MaxEnt_IO_Gen is the performance on the general dataset.

	Train			Test		
	P	**R**	**F1**	**P**	**R**	**F1**
Baseline	0.447	0.512	0.474	0.443	0.503	0.471
MaxEnt_Oracle	0.727	0.540	0.618	0.754	0.608	0.673
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	-	-	-	0.474	0.4	0.434

Table 2: Results of entity prediction for event triggers. MaxEnt_Oracle takes the gold event triggers and predict their arguments. MaxEnt_Basic is the basic model and MaxEnt_IO is the model that uses iterative optimization. MaxEnt_IO_Gen is the performance on the general dataset.

an F1 score of 0.505 considering the simplicity of the model. 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 performance of the argument prediction module. After we implemented the iterative optimization algorithm, the score got boosted to 0.513 and 0.522 in train and test set respectively. Again, this was the best performing model. The improvement in F1 score is mainly because we are now able to predict event triggers better. We also tested our entity prediction model on the general dataset and the results were lower than what we had in our actual dataset. We analyze this in the next section.

5.3 Semantic role labeling

The results for semantic role labeling are presented in Table 3. The baseline model for SRL builds on the baseline for entity prediction. As per the entity prediction baseline, we identify all nodes that are linked to a particular event and are of the type "NP". For these nodes, the baseline just assigns the most common semantic role encountered in the training set as the predicted role for this node. Since our dataset is small and the number of possible semantic roles is large, the results are not extraordinary as expected. We present a more detailed analysis for this in the next section.

Multi-class classification without considering joint labelings for the whole tree improves over the baseline. Finally, multi class classification considering joint role assignments and non-overlapping constraints gives a marginal boost in the score again, making it the best model for this task.

6 Analysis

In this section we first do general error analysis, then we talk about results of each task in detail.

Lexical vs Non-lexical features- Using lexical features helps to capture words or phrases that are repeating. At the same time, using features from the parse tree and dependency tree, including paths, helps us to identify structural traits. Since our training data was limited, using lexical features alone wasn't very helpful because most of the words and phrases did not repeat. At the same time, adding many features based on path resulted in high precision but low recall. So, we've adopted a mix of both lexical and non-lexical features.

Annotation errors- There were many instances when we thought the annotations were not correct. For example, events like 'transposition' and 'duplication' were not marked as a triggers in some sentences, maybe, because those were not important events according to the annotator. Also, while marking phrases that constitute an entity, there were many instances when multiple annotations

	Train			Test		
	P	**R**	**F1**	**P**	**R**	**F1**
Baseline	0.210	0.235	0.220	0.189	0.211	0.199
MaxEnt_Oracle	0.570	0.232	0.328	0.536	0.216	0.315
MaxEnt_Basic	0.523	0.210	0.296	0.465	0.193	0.273

Table 3: Results of semantic role labeling. MaxEnt_Oracle takes the gold event triggers and predict their arguments with semantic role. MaxEnt_Basic is the basic model with re-ranking.

were possible. This raises an import concern with all manually annotated corpora - that choices of annotations are subjective.

Parser errors- Since many of the features that we used are based on the parses generated for a sentence, any mistakes by the parser would automatically result in misclassifications of event and entities. A typical example is problems arising due to PP attachment of phrases that were marked as entities.

6.1 Event trigger prediction

Verb Nominalizations- 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. We tried NomLex and NomBank, which are collections of nominalizations. NomLex proved useful and the classification accuracy improved by around 2-3%. But, since NomLex was not an extensive dictionary, some nominalizations were still misclassified. Using WordNet derivations helped us to counter this and to improve the performance by another 2%. After including the features for word nominalizations, the model now identifies event triggers like 'effect', 'change', 'impact', 'degradation', 'concentration' etc. as event triggers. At the same time, there are many instances when nouns get tagged as event triggers even when they were not event triggers, for instance 'damage' was tagged as trigger from the phrase 'strand containing the damage'. Also, more complex nominalizations like 'synthesis', 'bronchitis' etc. were missed. This could be fixed by including gazetteers from biological or medical domain, but that would make our models very specific to that domain.

Iterative Optimization- The iterative optimization algorithm we developed was quite efficient in correcting some of the mistakes made by the basic trigger prediction algorithm. As an example, in the sentence, *The original cell produces a copy of its chromosome and surrounds it with a tough multilayered structure, forming the endospore.*, 'copy' was predicted as an event trigger, possibly because of strong lexical features. But, this gets corrected with the iterative algorithm as it does not have any child entities in the dependency tree. Similarly, in the sentence, *Each gene on one homolog is aligned precisely with the corresponding gene on the other homolog.*, 'aligned' was not initially predicted as an event trigger, but the iterative algorithm predicts it as a trigger.

We feel that including word clustering features that cluster together words that are semantically close would help us solve the problems arising with sparsity of labeled data. It is also interesting to note that the models performed well in the trigger identification task on a general dataset and it matched the intuition that our models should work for general datasets since we are not using any domain specific features or dictionaries.

6.2 Entity prediction

The task of argument prediction is a much harder task as evident from the lower F1 scores, mainly because of two reasons. Firstly, entities that are arguments to event triggers are phrases and marking boundaries of entities are subjective. We partly overcame this difficulty by modeling entities as parse tree nodes and devising a dynamic program for enforcing non overlapping constraints. The dynamic programming approach gave us a boost of around 4% in F1 score. Secondly, since we are now scoring event-entity combinations, identifying correct triggers to connect the entity to is a hard problem, especially in the cases when an entity is shared between two triggers or when the entities and trigger words are far apart in the sentence. We tried to overcome this by introducing features based on dependency parse structure of sentences as well.

The model to predict entities related to an event trigger did not perform well on the general dataset. We found that news headlines and articles on the web have a lot of named entities as arguments to event triggers, but, our model did not have any features to identify named entities. Once we include features to capture named entities, the performance of the model should be comparable to the results on our main dataset.

6.3 Semantic role labeling

1. **Domain Roles and Annotations-** The roles for our domain include 'Agent', 'Result', 'Location', 'Origin', 'Destination', and 'Theme'. These roles have overlapping semantic interpretations i.e. Location, Origin, Destination have very subtle semantic differences. Similarly, 'Result' and 'Theme' are often interchangeable in many scenarios. As a result, the annotations generated are subjective and too sparse to tease apart the subtle semantic differences during training.

2. **Parse Tree Node labeling-** Since this is the approach we chose across all our tasks, another error we commonly see is as follows. Consider (NP (NNS gametes)) in the sentence: *'Subsequently, the haploid organism carries out further mitoses, producing the cells that develop into gametes'*. Now we have the label (NP (NNS gametes)) as 'Result' in gold when actually the node (NNS gametes) is predicted as 'Result' in our model. These prediction errors are caused by the usage of parse tree nodes. We observe that non-overlapping constraints and re-ranking resolves these issues to a significant extent.

3. **Choice of k-** We experimented with different values of k in our top-k re-ranking model. The tradeoff is between space optimal calculations versus considering a large set of labelings for the best label. For our dataset, k=5 works well in terms of performance.

From our analysis on Semantic Role labeling we conclude that:

1. Having a smaller set of possible roles which are more mutually distinct would be very likely to improve the results.

2. A joint model for Semantic role labeling and entity prediction with the re-ranking approach could be attempted since these two tasks are closely tied together.

7 Conclusion

Event extraction is a promising area of research that will find a lot of application with the ever-increasing amount of data being generated with the increase in both newly published material and also with many popular books and articles being converted to electronic format. In this paper, we present a set of models for event and entity extraction with semantic role labeling. We have a basic trigger prediction model and an entity prediction model that predicts event triggers. Since we found that knowledge of entities can help to predict event triggers, we devised the iterative optimization algorithm that iteratively improves trigger and its argument prediction. Following this, we developed a dynamic programming approach for semantic role labeling.

We feel that combining the features from parse tree and dependency tree is something that hasn't been extensively used before and it proves very helpful. Also, the iterative approach to event and entity prediction works well considering the simplicity of the model. Further work would include adding semantic role labeling also into the iterative algorithm and also to include a temporal ordering of events to form a chain of events making a story.

Acknowledgments

We would like to thank Professor Christopher Manning and Jonathan Berant who is a post doc at Stanford NLP group for their constant guidance and support throughout the project. We would also like to thank the employees of Vulcan who helped to generate the dataset by manually annotating the paragraphs.

References

David McClosky, Mihai Surdeanu, and Christopher D. Manning 2011. Event Extraction as Dependency Parsing 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*.

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*.

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

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

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

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

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

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*.