Relation Extraction from Biomedical Text Using SVMs With Dependency Tree Kernels

Aaron Munger and Ankur Teredesai
University of Washington Tacoma
mungera@u.washington.com,
ankurt@u.washington.edu

ABSTRACT

Information extraction has become a useful technique in any field with that produces a lot of data. Our project has re-engineered and found improvements for a protein-protien interaction extraction system to work with drug interactions. We have taken advantage of a well formatted training data set from a recently held contest to evaluate the converted system and used this as a performance benchmark for improvements that we have designed. Although our system does not hold up to the performance of the state of the art systems, we have discovered methods to improve specific types of algorithms.

General Terms

DDI, Interaction Extraction, Dependency Tree Kernel

1. INTRODUCTION

Researchers in scientific domains are continually publishing papers regarding the results from individual experimentation and research which, when combined, represent a vast amount of knowledge. As more and more public information becomes available in the form of published papers, text mining techniques that make this vast amount of data meaningful as a whole become more useful. Specified information, e.g. drug-drug interactions as in our case, is usually very sparsely distributed among the large volumes of papers and therefore reliable text mining techniques are very useful in that they can parse through these volumes, extracting the requested specific information, in a reasonable amount of time.

A Drug-Drug Interaction (DDI) occurs when one drug influences the level or activity of another, for example, raising its blood drug levels and possibly intensifying its side effects or decreasing drug concentrations and thereby reducing its effectiveness. The detection of DDI is an important research area in patient safety since these interactions can become very dangerous and increase health care costs. Although there are different databases supporting health care professionals in the detection of DDI, these databases are rarely complete, since their update periods can reach three years [9]. Drug interactions are frequently reported in journals of clinical pharmacology and technical reports, making medical literature the most effective source for the detection of DDI. Thus, the management of DDI is a critical issue due to the overwhelming amount of information available on them.

Information Extraction (IE) can be of great benefit in the pharmaceutical industry allowing identification and extraction of relevant information on DDI and providing an interesting way of reducing the time spent by health care professionals on reviewing the literature. Moreover, the development of tools for automatically extracting DDI is essential for improving and updating the drug knowledge databases. Most investigation has focused on biological relationships (genetic and protein interactions (PPI)) due mainly to the availability of annotated corpora in the biological domain, a fact that facilitates the evaluation of approaches. Few approaches have focused on the extraction of DDIs[9].

We have designed this project to create a system for extracting these drug-drug interactions using the previously constructed system: PRISE, PRotein Interaction Search Engine.

A contest[1] was recently held that called on participants to apply text mining methods that would search through a given set of sentences and pick out and label the interactions that are found. This contest and other similar contests are created with the intent of providing a common framework for the evaluation of text mining techniques applied to the biomedical domain. The challenge task has provided a benchmark forum for comparing the latest advances of Information Extraction techniques applied to the extraction of drug-drug interactions.

The DDI extraction contest simplifies the problem of extracting relations by providing data that has already gone through a lot of preprocessing such as named entity recognition and part of speech tagging. The data is also simplified by only containing the sentences that have drugs in them, rather than entire papers.

PRISE can also be integrated into this project which would help determine if the best found methods for PPI extrac-

[1]http://labda.inf.uc3m.es/DDIExtraction2011/

tion also work well for DDI extraction. PRISE would also help in comparing different configurations more quickly than comparing the result sets by hand.

2. RELATED WORK

The proceedings from the DDI contest have been released and have been helpful in coming up with the ideas used in our project. The entries had two main ways of accomplishing the relation extraction task. One way was to look at the semantics of the sentence and location of the drug entities within the sentence. Trigger words, such as verbs like "interact" or "together", would be searched for and marked certain interactions as positive. An immediate problem with this approach is that sentences commonly have many drug words in them but have a very small portion of positive interactions between those pairs. It then becomes difficult to identify which pair of drugs that the trigger word applies to.

Information extraction methods have been previously applied in biological domains in the extraction of interaction information between proteins. Much of that work will be helpful in coming up with a reliable method for the DDI extraction problem.

PRISE, a PRotein Interaction Search Engine, was developed as a configurable information extraction system with a standardized evaluation system [4]. This program is an end to end system, starting with querying the website, PubMed, for papers on the subject and also included a front end interface to display the final results. Most of the underlying backend work in PRISE uses the open source data mining program, RapidMiner, which provides a variety of different methods for classification. These methods can be switched or configured in order to quickly compare setups and find the optimal configuration.

A relation extractor, RelEx, was another program developed for protein extraction. RelEx uses a selection of open software products for preprocessing and then applies a set of rules to the processed data to find the most likely candidates of interaction. The preprocessing includes part-of-speech tagging by MedPost, noun-phrase identifying by fnTBL, and a dependency parse tree by the Stanford Lexicalized Parser. Once the parse tree has been created, the program finds paths between genes or proteins to find relations in three different forms stated as rules: effector-relation-effectee, relation of effectee by effector, and relation between effector and effectee. The RelEx program greatly surpassed previously obtained results especially when applied to more stringent requirements such as finding the direction and type of the interaction rather than just the interaction itself [5].

The contest entry from [2] created a composite kernel that was made up of Shallow Linguistic (SL), Mildly Extended Dependency Tree (MEDT) and Phrase Structure Tree (PST) kernels. The SL kernel is feature based, using the similarities of the words around the drugs in the parts of speech or lemmas of the words themselves. The MEDT kernel is very similar to the kernel that our system uses at its core. The Dependency Tree is created by the Stanford Lexicalized Parser from which a shortest path between the two entities is extracted. The shortest path is extended with the rules described later in this paper. The extended path is then input for the SVM-Light classifier which gives a count of similar subtrees between two paths as the kernel measure. The PST kernel uses the smallest common subtree of the phrase structure parse tree that includes both entities in the relation and is described further in [7].

The method that resulted in the top scoring entry in the DDI Extraction Challenge was an ensemble of three different kernels and a case based reasoning classifier called Moara [10]. The sentences were preprocessed with the Charniak-Lease Parser [6] to create parse trees and then the Stanford Converter to create dependency trees. The choice of kernels was based off of the analysis given in [11] where 9 kernels were compared in relation extraction and the All Paths Graph (APG), k-Band Shortest Path Spectrum (kBSPS) and SL Kernels gave the best performance.

The case based reasoning algorithm uses flat features of the sentence from the parts of speech and lemmas of the words. A sample of known cases are stored with whether or not they describe an interaction. Each test instance is compared to find the most similar case using sequence matching and that case is used for the answer. The APG kernel looks for common subtrees of all lengths, including the surface word sequence, between two entity pairs and weights them inversely to their length. The kBSPS kernel is a composite kernel in itself, working with three different types of dependency tree comparison. Further explanations of these kernels can be found in [11]. The three chosen kernels and Moara were used in combinations of three with a simple voting ensemble algorithm.

3. RELATION EXTRACTION SYSTEM

This project was based off of previous works for protein-protein interaction extraction using a Support Vector Machine with a dependency tree kernel. The original system was created for extracting Protein-Protein Interactions and had many more features than have been included for this project. PRISE included the ability to query databases to find the biomedical papers, split them into individual sentence entities and also explicitly label each of the proteins. All of that preprocessing has been done for us in the data provided from the DDI contest.

PRISE also included a visualization feature for the post-processing of the data, showing which interactions have been found in an easy to understand format. This is another unnecessary step in our system since we are merely analyzing the performance of the algorithm and we do not intend to use the specifics of the resulting classifications.

RapidMiner is an open source data mining utility and provides the framework for PRISE. RapidMiner compliant operators are created and chained together through this framework, giving quite a few constraints to how the system is developed. Our system follows in the same framework with the idea that future works could re-incorporate the rest of the system.

Our goal is to maximize the performance of the classification algorithm, looking for the methods that will increase the accuracy of the system output. Focusing on the inner workings of the system will give an appropriate scope for

this size of project.

3.1 PRISE System Architecture

Despite reducing the complexity for this project, our system still goes through a number of significant steps. As mentioned before, the input to the system is in the form of individual sentences with the drug entities labeled. This section will describe the architecture and sequence of our system.

The class model is broken up in a way that corresponds to the input data as shown in figure 3.1. In this way we cycle through each abstract, creating sentence objects for each given sentence and DrugPair objects for each co-occurring pair of drugs. The Pair objects are labelled as interactions if when that pair is described as interacting in the sentence.

Once all the sentence objects have been created from the input data, we can begin preprocessing the data for the SVM classifier. The first step is to relabel all the drug names in the sentence to ENTITY#, where # is replaced with its position in the sentence. The names are conformed in this way so that the name of the drug will not affect the classifiers decision. We also try to clean up the sentences by removing the '?' characters that have seemingly been added when a parser outside of this system could not identify a character. The question marks were causing problems for our sentence parser by splitting the sentence in two.

At this point the Stanford Lexicalized Parser can be used to create dependency trees from each of the sentence. This operation is the slow point in our system, taking over an hour to parse the entire data set. Fortunately, the parser creates the exact same set of dependency trees every time and therefore we can speed up our operation by serializing the dependency trees to be loaded up each time the system is run. When each sentence object has its corresponding dependency tree, we then break down the tree for each pair object.

Our base system simply uses the shortest path between the pair of drugs, found with Dijkstra's shortest path algorithm, as the dependency path. The SVMTrainer object, a RapidMiner operator, writes out the dependency trees in the format that SVMLight can read. The SVMLight classifier takes these shortest paths as input to create the optimal model from the given training data.

With a trained model, we go through the same steps of parsing the training data but then use SVMLight to classify the test data using the model. All that remains is to check the accuracy of the labels given by SVMLight and analyze the performance.

3.2 Dependency Tree Kernel

An SVM requires a kernel function to provide a distance metric to specify a relation among items being classified. In the training phase, the SVM will generate a set of vectors to create a hyperplane for which one side will signify one class and the other side signifies the other class. Many standard kernels are able to use flat features, such as integers, and simply use the distance between them. In the case of classifying fragments of a sentence, the feature space quickly becomes unmanageably broad from the variety in language.

In our system, each sentence is first transformed into graphical representations using the Stanford Lexical Parser[2] to extract more specialized information as shown in Figure 2. In these graphs, each word is represented by a node and each edge represents a dependency between the words in the sentece. Each node is also labeled with the part of speech of the node in the sentence. Each concurrently appearing pair of drugs can then be represented by the path between them within the sentence graph. These paths can then be passed into the SVM to be classified.

The SVM in our system uses a dependency tree kernel that computes the similarity between the path being classified and those in the model. The graphical representation of the paths must have a feature space in which they can be compared. The standard way to perform this is converting the the graphical information into a binary vector to create flat features. This is done by first enumerating features that are seen to correlate with positive instances and then finding similar instances in the graph. The SVM will then create a vector from all observed configurations in the graph that match configurations in the model as described in [7]. The problem with this is that syntactic features can be far too complex to be represented in a linear way. The structure may not be well represented by flat features and our method of naming these features may be flawed. To solve these problems, [7][8] both propose convolution tree kernels which allow for a more abstract definition of feature space so that the classifier may dynamically decide which features to use in the model.

Rather than explicitly defining features, our model is expressed in terms of examples from the training set represented as their original structure. By inputing the structures into the kernel function, we allow the SVM to work with a feature space that is exponential in terms of the size of the tree, sub structures, without needing to compute over that entire space [12]. The convolution kernels that we use find the similarity between paths by finding the similarity between each subtree in the paths and then combining the results. This algorithm is based off of the definition in [1] but avoids the unreasonable running time as it was reported to take 18 hours to parse 75 sentences. The run time is improved by only considering the subtrees that occur in the paths being compared rather than all possible subtrees.

4. INTERACTION PREDICTION

After recreating the PRISE system to work for DDI extraction, we wanted to attempt to improve the accuracy of the algorithm starting with PRISE as a baseline for performance. We hoped to find some domain specific attributes that could help signal interactions to augment the classifier already in place. We also borrowed techniques from other contestants that we considered viable as additions to our system. The rest of this section will describe the efforts that we have attempted.

The alternative MTMX data format divided the listed drugs

[2]http://nlp.stanford.edu/software/lex-parser.shtml

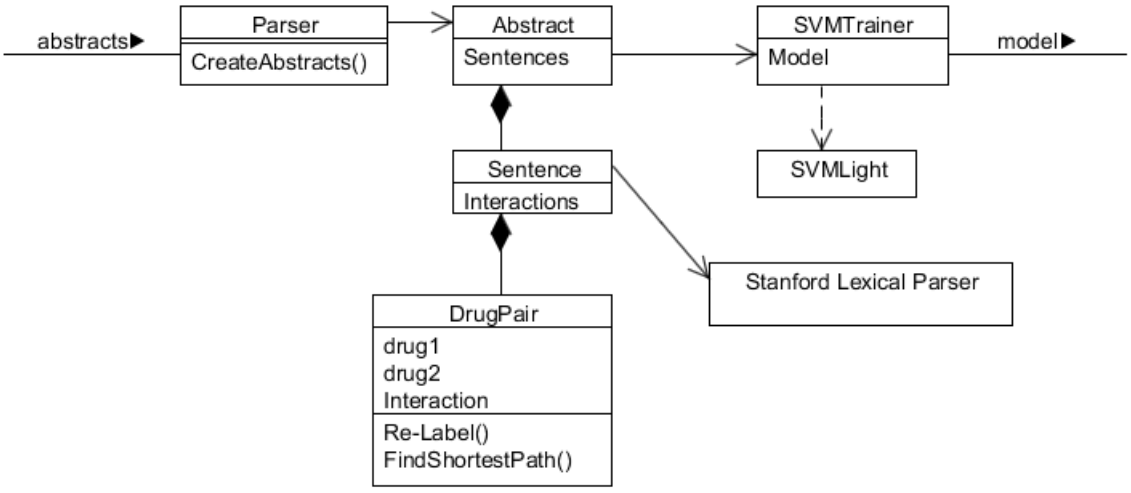

Figure 1: Class diagram of our system. The hierarchy corresponds to the input data and both the Stanford Lexical Parser and SVMTrainer are third party programs.

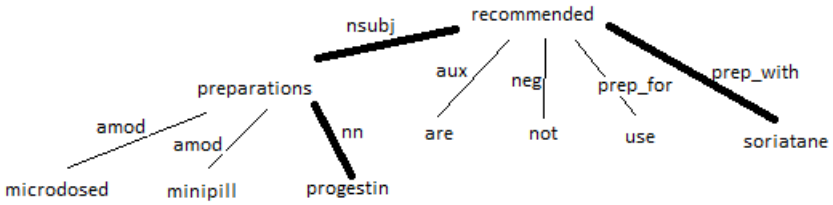

Figure 2: Semantic and lexical information as a dependency tree with the shortest path of an interaction in bold from the sentence: "Microdosed minipill progestin preparations are not recommended for use with Soriatane."

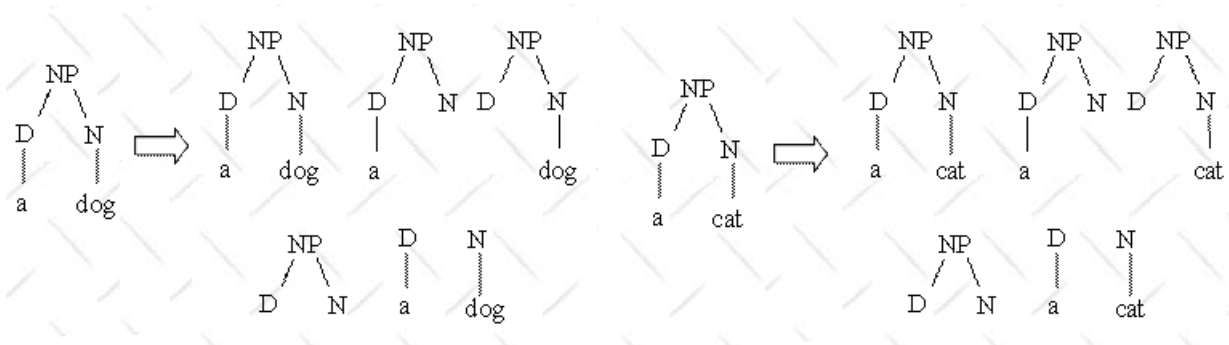

Figure 3: Total breakdowns of "a dog" and "a cat", three subtree structures are identical so the similarity score is 3. Image from [?]

into six different types and included a semantic type attribute to each drug phrase. The six drug semtypes are: Clinical Drug, Pharmacological Substance, Antibiotic, Biologically Active Substance, Chimical Viewed Structurally and Amino Acid, Peptide or Protien. There are also other semtypes listed along with the drugs that are not restricted to drugs and are also listed with other entities.

It is possible that these types of drugs could correlate with how drugs interact with eachother. In other words, one type of drug might commonly interact with another types and have little interaction with others. By including these semantic attributes as data for the classifier we hoped to help the algorithm by either increasing efficiency if the data was preprocessed so that certain interactions are not considered if it is highly unlikely given their attributes, or increasing the accuracy by initially favoring interactions that are more likely.

Another method to increase the accurracy of the system would implement graph theory on a graph representation of the interactions between drugs. Each drug would be represented with a node with an edge connecting them if there is a known interaction between them. The characteristics of a neighborhood around a drug node could be used to pick out other likely interactions. An example would be a situation that several drugs that interact with drug A also interact with drug B, when new interactions are found with either drug A or B, it is a higher probability that they will also react with the other.

The graph theory method would also be a helper method to RelEx, or some other text mining algorithm, in the context of the DDI contest in that it could give weight results to favor finding that interaction. The reason for this is that the contest is only looking to find drug interactions that are described in the text, and more specifically, described within one sentence. It would be interesting to see if these methods would work on there own as well, which would be done by leaving out the documents that described these interactions and see if they would still be resolved.

When looking through the training and test data created from the sentences, we found that many of the paths were too short to actually give much useful information. From this observation, we came up with the idea of extending the path tree in hopes that more features would be exposed for a more in depth comparison. Our first technique was to add on any part of the sentence that did not include any non-pair drugs in the sentence to get a more complete picture of the sentence. However, This technique resulted in the graphs being too large and multiple trees from a single sentence ended up being very similar. Part of the issue that lead to this poor result was that many of the sentence trees were formed in unforseen ways due to lack of experience in working with the sentence parser. One example was that a drug node could be a parent to a pair drug or even the root of the entire sentence graph.

[3] suggested an algorithm that they called Mildly Extended Dependency Tree (MEDT) that they used for PPI extraction. This algorithm was much more subtle in adding to the subgraph by opportunistically adding nodes to the tree when appropriate. Nodes are added when certain conditions are observed in the path that result in too little sentence structure for a meaningful comparison.

The MEDT algorithm used three rules for extending an interaction tree using the part of speech and dependency labels of the graph. The first rule is: if a dependency tree does not have a verb or modifier as the root node, try to add one from the parent or children or the root. The idea here is that verbs are often the trigger words that will signal an interaction but extending too far to include a verb could easily result in including a trigger word that does not actually relate to the path.

The second rule is: if the root of the path is a verb and the subject of that verb is not included in the path, include it in the path. This rule may seem counter-intuitive since adding a subject outside of the path might make it look much more like an interaction, however, the pair drugs are labelled in a way that distinguishes them from all other words in the sentence. Adding the subject of the verb allows more insight to what the sentence is about.

The third and final rule is: if the root is one of the drug entities in the pair being classified, add the parent of the root as the new root. This is a strange case to occur within a sentence

5. EVALUATION

There are standard methods to analyze the performance of a classifier, the two that we focus on are precision and recall, and also the combination of the two in a measure called the F-score. For simplicity, we will refer to a described interaction between a pair of drugs as a positive instance and a pair of drugs without a stated interaction as a negative instance. The recall will give the ratio of positives found to the total number of positives and the precision is the ratio of positives found to total instances labeled as postives.

So, for example, lets say we have a data set with 100 members, 32 of which are positive instances, and we correctly label 24 positive instances and incorrectly label another 6 as positive. Our recall is 24/32, or .75, and our precision is 24/30 or .6. In order to combine these two scores into one measure, we use the harmonic of the recall and precision as shown below to give us the F-score.

$$FScore = \frac{2(Recall * Precision)}{Recall + Precision}$$

By maximizing the F-score we are aiming at both identifying as many positive instances as possible while also avoiding the inclusion of excessive false positives. This evaluation system is also used to score the contest entries in the DDI extraction contest and is also the standard performance metric used in many of the related works that were researched.

We run the evaluation process in a ten fold cross validation format in which the data set is split up into training and testing segments in ten different groupings. Each run trains a model on its training set independently and then the model is evaluated on the test set in order to avoid evaluating on

Method	Precision	Recall	F-Score
Basic PRISE	25.1	38.8	30.3
MEDT PRISE	38.7	27.7	32.1
Preclustered PRISE	42.7	27.1	33.0

Figure 5: Table of the improvement measure of our system. The Preclustered PRISE also makes use of the MEDT algorithm.

particularly biased data set splits. The F-scores from each of these ten runs is averaged to give the overall performance of the system.

6. RESULTS

Using a recent contest as the platform for this project gives us an opportunity to compare our performance to other modern techniques that were submitted. Unfortunately, we found that our system did not perform on the same level as the top entries so our goal became finding techniques that would improve our system. In this section we will go over our previously stated additions to PRISE and show the effect of each technique on the performance of our system.

Earlier we mentioned that there was another data set available that has extra information available about the words and phrases of each sentence. In order for any significant correlation between semantic types and plausibility of interaction, the types must have a fair amount of distribution. After analyzing the data further, it was discovered that the semantic types were very poorly distributed, with 81% of the drugs being in one type. Because so much of the data set is in one type, that type becomes too generalized and takes away any significant effect on that majority. There was still hope, however, that one or more of the remaining drug types would signify the likelihood of an interaction with another drug.

After going through the evaluation process we found no noticeable effect of including semantic type information in the classifier. Whether or not this means that the type actually has nothing to do with how likely it is to interact is still unknown but we do know that it has no effect on whether or not it is described in a sentence.

The next technique implemented was to give include graph trait information that was created by setting up a graph with drugs as nodes and interactions as edges. This technique was also drawn up with the idea that the likelihood of two drugs interacting would influence an interaction being described in the sentence. The graph data that we created did have a favorable amount of variety in the statistics of nodes. The end result however, once again, showed no improvement our system's performance.

Our third addition to the system was to extend the dependency paths being fed into the classifier. This was implemented in two different ways, and with very different results. The first method was using our own algorithm to extend the interaction paths to include all branches in the sentence tree that did not include another drug. This solution ended up having a few problems with it, making it perform worse than the original algorithm. One problem was that the sentence trees could be formed in unforseen ways, such as a drug being the ancestor of a drug in the pair, even as the root in some cases. Another issue was that we were putting much larger paths into the classifier, causing it to slow down more than was acceptable. Aside from these two issues, the most fundamental problem was that multiple pairs from the same sentence had very similar dependency paths, making it harder for the classifier to find correlating features.

We then discovered the MEDT algorithm that only added one node at a time and only in specific cases. By adding nodes to the path that are linguistically significant, we end up with a much more meaningful extension to our trees. After implementing this algorithm we observed a two point increase in F-score.

The last technique that we used was to precluster the data using flat features extracted from the data. In this way we combine two types of classification that were used among the contestants: flat feature kernels and dependency tree kernels. After a lot of fine tuning to the clustering algorithm, we were able to add on one more point to our F-score by using the preclustering.

In summary, of the five techniques that were attempted to increase performance in our system, two were able to moderately achieve that goal. With these two new techniques we were able to raise our F-score from thirty points to thirty three, still not performing at the same level as the top contest entries but a step in the right direction.

7. DISCUSSION

We have concurred with another contest submission [2] that the likelihood of two drugs interacting has no noticeable correlation with whether or not that interaction is described in a sentence in which they co-occur. This was shown by both MMTX data type and graph attributes having no effect on the performance of our system. These attempts at predicting interactions could possibly have their own use in choosing which drugs to experiment with but it does not help in our case.

The improvement by expanding the dependency path has led us to the conclusion those kernel that use the shortest path in a dependency tree, or perhaps other semantic trees, could potentially be improved. The algorithms described in [10], for example, could be further improved by implementing the MEDT algorithm. This would potentially improve the performance of one member of their ensemble, adding to the overall performance.

We were able to combine the flat feature kernel and the dependency tree kernel in a more creative way than the standard voting ensemble and did so with some success. This different style of combining kernels could lead to many other beneficial combinations. Rather than running a selection of classifiers over the same data, this method divides the data into similar segments for reduced entropy in each set of data. Because of this difference in data composition, it is likely that several classifiers would perform quite differently, requiring a new round of evaluations and rankings.

Method	Precision	Recall	F-Score
Hybrid Linguistic Approach	48.7	25.7	33.6
Shallow Linguistic Kernel	51.0	72.8	60.0
Ensemble Kernels with CBR	60.5	71.9	65.7
Composite Kernel	43.4	52.8	47.6
PRISE with MEDT and Preclustering	42.7	27.1	33.0

Figure 4: The Hybrid Linguistic Approach and Shallow Linguistic Kernel were the baseline algorithms for contest entries [?]. The Ensemble Kernels with Case Based Reasoning(CBR) [10] was the top performer in the contest. The composite kernel from [2] was the most similar to our system. PRISE with MEDT and Preclustering was our top performing algorithm.

8. FUTURE WORK

The likely next step here would be to implement these findings in more sophisticated classifiers using ensemble techniques to combine results of multiple classifiers. Each individual classifier could be tested for improvement by extening the dependency path or preclustering the data so that different classifier models are used on each cluster.

We have focused down the problem to a very specific point so that any future work could build off of our findings and incorperate them into a larger system. We have added improvements to one specific classifier without using any ensemble techniques so that any ensemble that includes a classifier similar to ours can benifit from better performance from one of its members.

9. CONCLUSIONS

After looking at the results from both our system and the top performing entries in the DDI contest, it is easy to conclude that DDI interaction extraction is a difficult problem and modern solutions have a lot of room for improvement.

The best scores were achieved by using multiple classifiers together in a way that they can make up for the weaknesses of one another. With this powerful combonation of classifiers that has been discovered as the optimal configuration, each individual classifier can be improved by giving direct focus to each one as we have done with dependency tree kernels in this project.

10. REFERENCES

[1] R. Bod. Beyond grammar: An experience-based theory of language. *CSLI Publications*, 1998.

[2] F. M. Chowdhury, A. B. Abacha, A. Lavelli, and P. Zweigenbaum. Two different machine learning techniques for drug-drug interaction extraction. DDIExtraction 2011, 2011.

[3] F. M. Chowdhury, A. Lavelli, and A. Moschitti. A study on dependency tree kernels for automatic extraction of protein-protein interaction. pages 124–133. Association for Computational Linguistics, BioNLP 2011 Workshop, 2011.

[4] T. Fayruzov. *Mining and Modeling Interaction Networks for System Biology.* PhD thesis, Ghent University, 2010.

[5] K. Fundel, R. KÃijffner, and R. Zimmer. Relex–relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371, 2007.

[6] M. Lease and E. Charniak. Parsing biomedical literature. pages 58–69. IJCNLP'05, 2005.

[7] A. Moschitti. A study on convolution kernels for shallow semantic parsing. Barcelona, Spain, 2004. 42nd Annual Meeting on Association for Computational Linguistics.

[8] A. Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. *European Conference on Machine Learning*, pages 318–329, 2006.

[9] I. Segura-Bedmar, P. Martinez, and D. Sanchez-Cisneros. The 1st ddiextraction-2011 challenge task: Extraction of drug-drug interactions from biomedical texts. Madrid, Spain, 2011. Universidad Carlos III de Madrid, Computer Science Department.

[10] P. Thomas, M. Neves, I. Solt, D. Tikk, and U. Leser. Relation extraction for drug-drug interactions using ensemble learning. Madrid, Spain, 2011. Humboldt-Universitat zu Berlin, Knowledge Management in Bioinformatics, DDIExtraction 2011.

[11] D. Tikk, P. Thomas, P. Palaga, J. Hakenberg, and U. Leser. A comprehensive benchmark of kernel methods to extract protein-protein interactions from literature.

[12] D. Zelenko, A. Chinatsu, and A. Richardello. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 2003.