Sentiment Classification using Automatically Extracted Subgraph Features

Shilpa Arora, Elijah Mayfield, Carolyn Penstein-Rosé and Eric Nyberg
Language Technologies Institute
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213
{shilpaa, emayfiel, cprose, ehn}@cs.cmu.edu

Abstract

In this work, we propose a novel representation of text based on patterns derived from linguistic annotation graphs. We use a subgraph mining algorithm to automatically derive features as frequent subgraphs from the annotation graph. This process generates a very large number of features, many of which are highly correlated. We propose a genetic programming based approach to feature construction which creates a fixed number of strong classification predictors from these subgraphs. We evaluate the benefit gained from evolved structured features, when used in addition to the bag-of-words features, for a sentiment classification task.

1 Introduction

In recent years, the topic of sentiment analysis has been one of the more popular directions in the field of language technologies. Recent work in supervised sentiment analysis has focused on innovative approaches to feature creation, with the greatest improvements in performance with features that insightfully capture the essence of the linguistic constructions used to express sentiment, e.g. (Wilson et al., 2004), (Joshi and Rosé, 2009)

In this spirit, we present a novel approach that leverages subgraphs automatically extracted from linguistic annotation graphs using efficient subgraph mining algorithms (Yan and Han, 2002). The difficulty with automatically deriving complex features comes with the increased feature space size. Many of these features are highly correlated and do not provide any new information to the model. For example, a feature of type *unigram_POS* (e.g. *"camera_NN"*) doesn't provide any additional information beyond the *unigram* feature (e.g. *"camera"*), for words that are often used with the same part of speech. However, alongside several redundant features, there are also features that provide new information. It is these features that we aim to capture.

In this work, we propose an evolutionary approach that constructs complex features from subgraphs extracted from an annotation graph. A constant number of these features are added to the unigram feature space, adding much of the representational benefits without the computational cost of a drastic increase in feature space size.

In the remainder of the paper, we review prior work on features commonly used for sentiment analysis. We then describe the annotation graph representation proposed by Arora and Nyberg (2009). Following this, we describe the frequent subgraph mining algorithm proposed in Yan and Han (2002), and used in this work to extract frequent subgraphs from the annotation graphs. We then introduce our novel feature evolution approach, and discuss our experimental setup and results. Subgraph features combined with the feature evolution approach gives promising results, with an improvement in performance over the baseline.

2 Related Work

Some of the recent work in sentiment analysis has shown that structured features (features that capture syntactic patterns in text), such as n-grams, dependency relations, etc., improve performance beyond

Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text, pages 131–139,
Los Angeles, California, June 2010.

the bag of words approach. Arora et al. (2009) show that deep syntactic scope features constructed from transitive closure of dependency relations give significant improvement for identifying types of claims in product reviews. Gamon (2004) found that using deep linguistic features derived from phrase structure trees and part of speech annotations yields significant improvements on the task of predicting satisfaction ratings in customer feedback data. Wilson et al. (2004) use syntactic clues derived from dependency parse tree as features for predicting the intensity of opinion phrases[1].

Structured features that capture linguistic patterns are often hand crafted by domain experts (Wilson et al., 2005) after careful examination of the data. Thus, they do not always generalize well across datasets and domains. This also requires a significant amount of time and resources. By automatically deriving structured features, we might be able to learn new annotations faster.

Matsumoto et al. (2005) propose an approach that uses frequent sub-sequence and sub-tree mining approaches (Asai et al., 2002; Pei et al., 2004) to derive structured features such as word sub-sequences and dependency sub-trees. They show that these features outperform bag-of-words features for a sentiment classification task and achieve the best performance to date on a commonly-used movie review dataset. Their approach presents an automatic procedure for deriving features that capture long distance dependencies without much expert intervention.

However, their approach is limited to sequences or tree annotations. Often, features that combine several annotations capture interesting characteristics of text. For example, Wilson et al. (2004), Gamon (2004) and Joshi and Rosé (2009) show that a combination of dependency relations and part of speech annotations boosts performance. The annotation graph representation proposed by Arora and Nyberg (2009) is a formalism for representing several linguistic annotations together on text. With an annotation graph representation, instances are represented as graphs from which frequent subgraph patterns may be extracted and used as features for learning new annotations.

[1] Although, in this work we are classifying sentences and not phrases, similar clues may be used for sentiment classification in sentences as well

In this work, we use an efficient frequent subgraph mining algorithm (gSpan) (Yan and Han, 2002) to extract frequent subgraphs from a linguistic annotation graph (Arora and Nyberg, 2009). An annotation graph is a general representation for arbitrary linguistic annotations. The annotation graph and subgraph mining algorithm provide us a quick way to test several alternative linguistic representations of text. In the next section, we present a formal definition of the annotation graph and a motivating example for subgraph features.

3 Annotation Graph Representation and Feature Subgraphs

Arora and Nyberg (2009) define the annotation graph as a quadruple: $G = (N, E, \Sigma, \lambda)$, where N is the set of nodes, E is the set of edges, s.t. $E \subset N \times N$, and $\Sigma = \Sigma_N \cup \Sigma_E$ is the set of labels for nodes and edges. $\lambda : N \cup E \rightarrow \Sigma$ is the labeling function for nodes and edges. Examples of node labels (Σ_N) are *tokens (unigrams)* and annotations such as *part of speech*, *polarity* etc. Examples of edge labels (Σ_E) are *leftOf*, *dependency type* etc. The *leftOf* relation is defined between two adjacent nodes. The *dependency type* relation is defined between a head word and its modifier.

Annotations may be represented in an annotation graph in several ways. For example, a dependency triple annotation 'good_amod_movie', may be represented as a *d_amod* relation between the head word 'movie' and its modifier 'good', or as a node *d_amod* with edges *ParentOfGov* and *ParentOfDep* to the head and the modifier words. An example of an annotation graph is shown in Figure 1.

The instance in Figure 1 describes a movie review comment, *'interesting, but not compelling.'*. The words 'interesting' and 'compelling' both have positive prior polarity, however, the phrase expresses negative sentiment towards the movie. Heuristics for special handling of *negation* have been proposed in the literature. For example, Pang et al. (2002) append every word following a negation, until a punctuation, with a 'NOT' . Applying a similar technique to our example gives us two sentiment bearing features, one positive (*'interesting'*) and one negative (*'NOT-compelling'*), and the model may not be as sure about the predicted label, since there is both

positive and negative sentiment present.

In Figure 2, we show three discriminating subgraph features derived from the annotation graph in Figure 1. These subgraph features capture the negative sentiment in our example phrase. The first feature in 2(a) captures the pattern using dependency relations between words. A different review comment may use the same linguistic construction but with a different pair of words, for example *"a pretty good, but not excellent story."* This is the same linguistic pattern but with different words the model may not have seen before, and hence may not classify this instance correctly. This suggests that the feature in 2(a) may be too specific.

In order to mine general features that capture the rhetorical structure of language, we may add prior polarity annotations to the annotation graph, using a lexicon such as Wilson et al. (2005). Figure 2(b) shows the subgraph in 2(a) with polarity annotations. If we want to generalize the pattern in 2(a) to any positive words, we may use the feature subgraph in Figure 2(c) with X wild cards on words that are polar or negating. This feature subgraph captures the negative sentiment in both phrases *'interesting, but not compelling.'* and *"a pretty good, but not excellent story."*. Similar generalization using wild cards on words may be applied with other annotations such as part of speech annotations as well. By choosing where to put the wild card, we can get features similar to, but more powerful than, the dependency back-off features in Joshi and Rosé (2009).

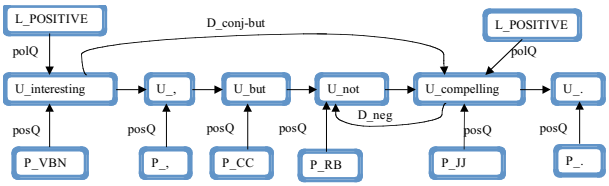

Figure 1: Annotation graph for sentence *'interesting, but not compelling.'* . Prefixes: 'U' for unigrams (tokens), 'L' for polarity, 'D' for dependency relation and 'P' for part of speech. Edges with no label encode the 'leftOf' relation between words.

4 Subgraph Mining Algorithms

In the previous section, we demonstrated that subgraphs from an annotation graph can be used to identify the rhetorical structure used to express sentiment. The subgraph patterns that represent general linguistic structure will be more frequent than surface level patterns. Hence, we use a frequent subgraph mining algorithm to find frequent subgraph patterns, from which we construct features to use in the supervised learning algorithm.

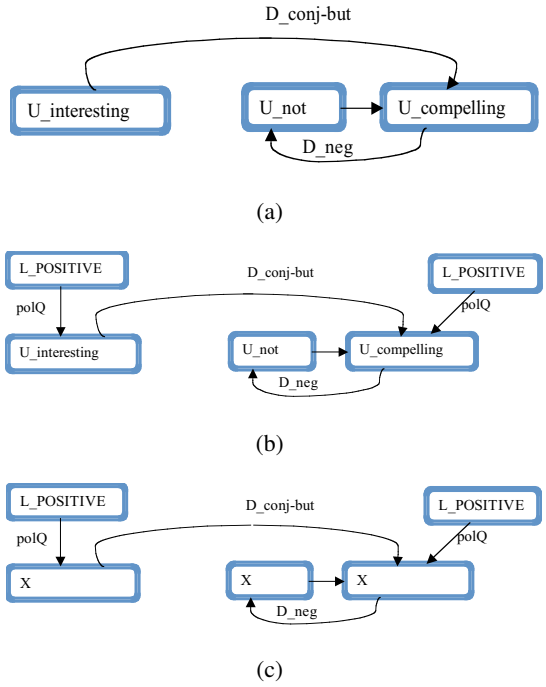

Figure 2: Subgraph features from the annotation graph in Figure 1

The goal in frequent subgraph mining is to find frequent subgraphs in a collection of graphs. A graph G' is a subgraph of another graph G if there exists a subgraph isomorphism[2] from G' to G, denoted by $G' \sqsubseteq G$.

Earlier approaches in frequent subgraph mining (Inokuchi et al., 2000; Kuramochi and Karypis, 2001) used a two-step approach of first generating the candidate subgraphs and then testing their frequency in the graph database. The second step involves a subgraph isomorphism test, which is NP-complete. Although efficient isomorphism testing algorithms have been developed making it practical to use, with lots of candidate subgraphs to test, it can

[2] http://en.wikipedia.org/wiki/Subgraph_isomorphism_problem

still be very expensive for real applications.

gSpan (Yan and Han, 2002) uses an alternative pattern growth based approach to frequent subgraph mining, which extends graphs from a single subgraph directly, without candidate generation. For each discovered subgraph G, new edges are added recursively until all frequent supergraphs of G have been discovered. gSpan uses a depth first search tree (DFS) and restricts edge extension to only vertices on the rightmost path. However, there can be multiple DFS trees for a graph. gSpan introduces a set of rules to select one of them as representative. Each graph is represented by its unique canonical DFS code, and the codes for two graphs are equivalent if the graphs are isomorphic. This reduces the computational cost of the subgraph mining algorithm substantially, making gSpan orders of magnitude faster than other subgraph mining algorithms. With several implementations available [3], gSpan has been commonly used for mining frequent subgraph patterns (Kudo et al., 2004; Deshpande et al., 2005). In this work, we use gSpan to mine frequent subgraphs from the annotation graph.

5 Feature Construction using Genetic Programming

A challenge to overcome when adding expressiveness to the feature space for any text classification problem is the rapid increase in the feature space size. Among this large set of new features, most are not predictive or are very weak predictors, and only a few carry novel information that improves classification performance. Because of this, adding more complex features often gives no improvement or even worsens performance as the feature space's signal is drowned out by noise.

Riloff et al. (2006) propose a feature subsumption approach to address this issue. They define a hierarchy for features based on the information they represent. A complex feature is only added if its discriminative power is a delta above the discriminative power of all its simpler forms. In this work, we use a Genetic Programming (Koza, 1992) based approach which evaluates interactions between features and evolves complex features from them. The advantage of the genetic programing based approach over feature subsumption is that it allows us to evaluate a feature using multiple criteria. We show that this approach performs better than feature subsumption.

A lot of work has considered this genetic programming problem (Smith and Bull, 2005). The most similar approaches to ours are taken by Krawiec (2002) and Otero et al. (2002), both of which use genetic programming to build tree feature representations. None of this work was applied to a language processing task, though there has been some similar work to ours in that community, most notably (Hirsch et al., 2007), which built search queries for topic classification of documents. Our prior work (Mayfield and Rosé, 2010) introduced a new feature construction method and was effective when using unigram features; here we extend our approach to feature spaces which are even larger and thus more problematic.

The Genetic Programming (GP) paradigm is most advantageous when applied to problems where there is not a correct answer to a problem, but instead there is a gradient of partial solutions which incrementally improve in quality. Potential solutions are represented as trees consisting of functions (non-leaf nodes in the tree, which perform an action given their child nodes as input) and terminals (leaf nodes in the tree, often variables or constants in an equation). The tree (an individual) can then be interpreted as a program to be executed, and the output of that program can be measured for fitness (a measurement of the program's quality). High-fitness individuals are selected for reproduction into a new generation of candidate individuals through a breeding process, where parts of each parent are combined to form a new individual.

We apply this design to a language processing task at the stage of feature construction - given many weakly predictive features, we would like to combine them in a way which produces a better feature. For our functions we use boolean statements AND and XOR, while our terminals are selected randomly from the set of all unigrams and our new, extracted subgraph features. Each leaf's value, when applied to a single sentence, is equal to 1 if that subgraph is present in the sentence, and 0 if the subgraph is not

[3] `http://www.cs.ucsb.edu/~xyan/software/gSpan.htm`, `http://www.kyb.mpg.de/bs/people/nowozin/gboost/`

present.

The tree in Figure 3 is a simplified example of our evolved features. It combines three features, a unigram feature 'too' (centre node) and two subgraph features: 1) the subgraph in the leftmost node occurs in collocations containing *"more than"* (e.g., *"nothing more than"* or *"little more than"*), 2) the subgraph in the rightmost node occurs in negative phrases such as *"opportunism at its most glaring"* (JJS is a superlative adjective and PRP$ is a possessive pronoun). A single feature combining these weak indicators can be more predictive than any part alone.

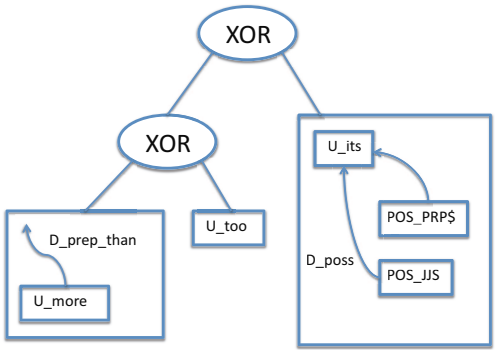

Figure 3: A tree constructed using subgraph features and GP (Simplified for illustrative purposes)

In the rest of this section, we first describe the feature construction process using genetic programming. We then discuss how fitness of an individual is measured for our classification task.

5.1 Feature Construction Process

We divide our data into two sets, training and test. We again divide our training data in half, and train our GP features on only one half of this data[4] This is to avoid overfitting the final SVM model to the GP features. In a single GP run, we produce one feature to match each class value. For a sentiment classification task, a feature is evolved to be predictive of the positive instances, and another feature is evolved to be predictive of the negative documents. We repeat this procedure a total of 15 times (using different seeds for random selection of features), producing a total of 30 new features to be added to the feature space.

[4]For genetic programming we used the ECJ toolkit (`http://cs.gmu.edu/~eclab/projects/ecj/`).

5.2 Defining Fitness

Our definition of fitness is based on the concepts of precision and recall, borrowed from information retrieval. We define our set of documents as being comprised of a set of positive documents $P_0, P_1, P_2, ...P_u$ and a set of negative documents $N_0, N_1, N_2, ...N_v$. For a given individual I and document D, we define $hit(I, D)$ to equal 1 if the statement I is true of that document and 0 otherwise. Precision and recall of an individual feature for predicting positive documents[5] is then defined as follows:

$$Prec(I) = \frac{\sum_{i=0}^{u} hit(I, P_i)}{\sum_{i=0}^{u} hit(I, P_i) + \sum_{i=0}^{v} hit(I, N_i)} \quad (1)$$

$$Rec(I) = \frac{\sum_{i=0}^{u} hit(I, P_i)}{u} \quad (2)$$

We then weight these values to give significantly more importance to precision, using the F_β measure, which gives the harmonic mean between precision and recall:

$$F_\beta(I) = \frac{(1 + \beta^2) \times (Prec(I) \times Rec(I))}{(\beta^2 \times Prec(I)) + Rec(I)} \quad (3)$$

In addition to this fitness function, we add two penalties to the equation. The first penalty applies to prevent trees from becoming overly complex. One option to ensure that features remain moderately simple is to simply have a maximum depth beyond which trees cannot grow. Following the work of Otero et al. (2002), we penalize trees based on the number of nodes they contain. This discourages bloat, i.e. sections of trees which do not contribute to overall accuracy. This penalty, known as parsimony pressure, is labeled *PP* in our fitness function.

The second penalty is based on the correlation between the feature being constructed, and the subgraphs and unigrams which appear as nodes within that individual. Without this penalty, a feature may

[5]Negative precision and recall are defined identically, with obvious adjustments to test for negative documents instead of positive.

often be redundant, taking much more complexity to represent the same information that is captured with a simple unigram. We measure correlation using Pearson's product moment, defined for two vectors X, Y as:

$$\rho_{x,y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y} \quad (4)$$

This results in a value from 1 (for perfect alignment) to -1 (for inverse alignment). We assign a penalty for any correlation past a cutoff. This function is labeled *CC* (correlation constraint) in our fitness function.

Our fitness function therefore is:

$$\text{Fitness} = F_{\frac{1}{8}} + PP + CC \quad (5)$$

6 Experiments and Results

We evaluate our approach on a sentiment classification task, where the goal is to classify a movie review sentence as expressing positive or negative sentiment towards the movie.

6.1 Data and Experimental Setup

Data: The dataset consists of snippets from Rotten Tomatoes (Pang and Lee, 2005) [6]. It consists of 10662 snippets/sentences total with equal number positive and negative sentences (5331 each). This dataset was created and used by Pang and Lee (2005) to train a classifier for identifying positive sentences in a full length review. We use the first 8000 (4000 positive, 4000 negative) sentences as training data and evaluate on remaining 2662 (1331 positive, 1331 negative) sentences. We added part of speech and dependency triple annotations to this data using the Stanford parser (Klein and Manning, 2003).

Annotation Graph: For the annotation graph representation, we used *Unigrams (U)*, *Part of Speech (P)* and *Dependency Relation Type (D)* as labels for the nodes, and *ParentOfGov* and *ParentOfDep* as labels for the edges. For a dependency triple such as "amod_good_movie", five nodes are added to the annotation graph as shown in Figure 4(a). *ParentOfGov* and *ParentOfDep* edges are added from the dependency relation node *D_amod* to the unigram nodes *U_good* and *U_movie*. These edges are also added for the part of speech nodes that correspond to the two unigrams in the dependency relation, as shown in Figure 4(a). This allows the algorithm to find general patterns, based on a dependency relation between two part of speech nodes, two unigram nodes or a combination of the two. For example, a subgraph in Figure 4(b) captures a general pattern where *good* modifies a noun. This feature exists in "amod_good_movie", "amod_good_camera" and other similar dependency triples. This feature is similar to the the dependency back-off features proposed in Joshi and Rosé (2009).

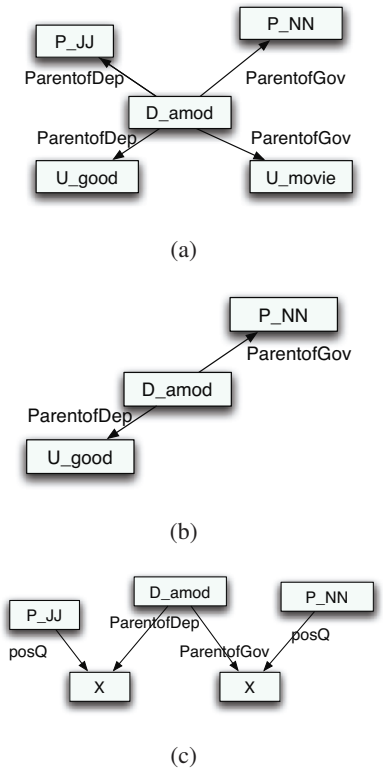

Figure 4: Annotation graph and a feature subgraph for dependency triple annotation "amod_good_camera". (c) shows an alternative representation with wild cards

The extra edges are an alternative to putting wild cards on words, as proposed in section 3. On the other hand, putting a wild card on every word in the annotation graph for our example (Figure 4(c)), will only give features based on dependency relations between part of speech annotations. Thus, the wild card based approach is more restrictive than

[6] http://www.cs.cornell.edu/people/pabo/movie-review-data/rt-polaritydata.tar.gz

adding more edges. However, with lots of edges, the complexity of the subgraph mining algorithm and the number of subgraph features increases tremendously.

Classifier: For our experiments we use Support Vector Machines (SVM) with a linear kernel. We use the SVM-light[7] implementation of SVM with default settings.

Parameters: The *gSpan* algorithm requires setting the minimum support threshold ($minsup$) for the subgraph patterns to extract. Support for a subgraph is the number of graphs in the dataset that contain the subgraph. We experimented with several values for minimum support and $minsup = 2$ gave us the best performance.

For Genetic Programming, we used the same parameter settings as described in Mayfield and Rosé (2010), which were tuned on a different dataset[8] than one used in this work, but it is from the same movie review domain. We also consider one alteration to these settings. As we are introducing many new and highly correlated features to our feature space through subgraphs, we believe that a stricter constraint must be placed on correlation between features. To accomplish this, we can set our correlation penalty cutoff to 0.3, lower than the 0.5 cutoff used in prior work. Results for both settings are reported.

Baselines: To the best of our knowledge, there is no supervised machine learning result published on this dataset. We compare our results with the following baselines:

- *Unigram-only Baseline:* In sentiment analysis, unigram-only features have been a strong baseline (Pang et al., 2002; Pang and Lee, 2004). We only use unigrams that occur in at least two sentences of the training data same as Matsumoto et al. (2005). We also filter out stop words using a small stop word list[9].

- χ^2 *Baseline:* For our training data, after filtering infrequent unigrams and stop words, we get 8424 features. Adding subgraph features increases the total number of features to $44,161$, a factor of 5 increase in size. Feature selection can be used to reduce this size by selecting the most discriminative features. χ^2 feature selection (Manning et al., 2008) is commonly used in the literature. We compare two methods of feature selection with χ^2, one which rejects features if their χ^2 score is not significant at the 0.05 level, and one that reduces the number of features to match the size of our feature space with GP.

- *Feature Subsumption (FS):* Following the idea in Riloff et al. (2006), a complex feature C is discarded if $IG(S) \geq IG(C) - \delta$, where IG is Information Gain and S is a simple feature that *representationally subsumes* C, i.e. the text spans that match S are a superset of the text spans that match C. In our work, complex features are subgraph features and simple features are unigram features contained in them. For example, $(D\_amod)\_Edge\_ParentOfDep\_(U\_bad)$ is a complex feature for which U_bad is a simple feature. We tried same values for $\delta \in \{0.002, 0.001, 0.0005\}$, as suggested in Riloff et al. (2006). Since all values gave us same number of features, we only report a single result for feature subsumption.

- *Correlation (Corr):* As mentioned earlier, some of the subgraph features are highly correlated with unigram features and do not provide new knowledge. A correlation based filter for subgraph features can be used to discard a complex feature C if its absolute correlation with its simpler feature (unigram feature) is more than a certain threshold. We use the same threshold as used in the GP criterion, but as a hard filter instead of a penalty.

6.2 Results and Discussion

In Table 1, we present our results. As can be seen, subgraph features when added to the unigrams, without any feature selection, decrease the performance. χ^2 feature selection with fixed feature space size provides a very small gain over unigrams. All other feature selection approaches perform worse

[7] http://svmlight.joachims.org/

[8] Full movie review data by Pang et al. (2002)

[9] http://nlp.stanford.edu/IR-book/html/htmledition/dropping-common-terms-stop-words-1.html (with one modification: removed 'will', added 'this')

Settings	#Features	Acc.	Δ
Uni	8424	75.66	-
Uni + Sub	44161	75.28	-0.38
Uni + Sub, χ^2 sig.	3407	74.68	-0.98
Uni + Sub, χ^2 size	8454	75.77	+0.11
Uni + Sub, (FS)	18234	75.47	-0.19
Uni + Sub, (Corr)	18980	75.24	-0.42
Uni + GP (U) †	8454	76.18	+0.52
Uni + GP (U+S) ‡	8454	76.48	+0.82
Uni + GP (U+S) †	8454	**76.93**	+1.27

Table 1: Experimental results for feature spaces with unigrams, with and without subgraph features. Feature selection with 1) fixed significance level (χ^2 sig.), 2) fixed feature space size (χ^2 size), 3) Feature Subsumption (FS) and 4) Correlation based feature filtering (Corr)). GP features for unigrams only {GP(U)}, or both unigrams and subgraph features {GP(U+S)}. Both the settings from Mayfield and Rosé (2010) (‡) and more stringent correlation constraint (†) are reported. $\#Features$ is the number of features in the training data. Acc is the accuracy and Δ is the difference from unigram only baseline. Best performing feature configuration is highlighted in bold.

than the unigram-only approach. With GP, we observe a marginally significant gain ($p < 0.1$) in performance over unigrams, calculated using one-way ANOVA. Benefit from GP is more when subgraph features are used in addition to the unigram features, for constructing more complex pattern features. Additionally, our performance is improved when we constrain the correlation more severely than in previously published research, supporting our hypothesis that this is a helpful way to respond to the problem of redundancy in subgraph features.

A problem that we see with χ^2 feature selection is that several top ranked features may be highly correlated. For example, the top 5 features based on χ^2 score are shown in Table 2; it is immediately obvious that the features are highly redundant.

With GP based feature construction, we can consider this relationship between features, and construct new features as a combination of selected unigram and subgraph features. With the correlation criterion in the evolution process, we are able to build combined features that provide new information compared to unigrams.

The results we present are for the best performing parameter configuration that we tested, after a series of experiments. We realize that this places us in danger of overfitting to the particulars of this data set, however, the data set is large enough to partially mitigate this concern.

(D_advmod)_Edge_ParentOfDep _(U_too)
U_too
U_bad
U_movie
(D_amod)_Edge_ParentOfDep _(U_bad)

Table 2: Top features based on χ^2 score

7 Conclusion and Future Work

We have shown that there is additional information to be gained from text beyond words, and demonstrated two methods for increasing this information - a subgraph mining approach that finds common syntactic patterns that capture sentiment-bearing rhetorical structure in text, and a feature construction technique that uses genetic programming to combine these more complex features without the redundancy, increasing the size of the feature space only by a fixed amount. The increase in performance that we see is small but consistent.

In the future, we would like to extend this work to other datasets and other problems within the field of sentiment analysis. With the availability of several off-the-shelf linguistic annotators, we may add more linguistic annotations to the annotation graph and richer subgraph features may be discovered. There is also additional refinement that can be performed on our genetic programming fitness function, which is expected to improve the quality of our features.

Acknowledgments

This work was funded in part by the DARPA Machine Reading program under contract FA8750-09-C-0172, and in part by NSF grant DRL-0835426. We would like to thank Dr. Xifeng Yan and Marisa Thoma for the gSpan code.

References

Shilpa Arora, Mahesh Joshi and Carolyn P. Rosé. 2009. *Identifying Types of Claims in Online Customer Re-*

views. Proceedings of the HLT/NAACL.

Shilpa Arora and Eric Nyberg. 2009. *Interactive Annotation Learning with Indirect Feature Voting*. Proceedings of the HLT/NAACL (Student Research Workshop).

Tatsuya Asai, Kenji Abe, Shinji Kawasoe, Hiroshi Sakamoto and Setsuo Arikawa. 2002. *Efficient substructure discovery from large semi-structured data*. Proceedings of SIAM Int. Conf. on Data Mining (SDM).

Mukund Deshpande , Michihiro Kuramochi , Nikil Wale and George Karypis. 2005. *Frequent Substructure-Based Approaches for Classifying Chemical Compounds*. IEEE Transactions on Knowledge and Data Engineering.

Michael Gamon. 2004. *Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis*, Proceedings of COLING.

Laurence Hirsch, Robin Hirsch and Masoud Saeedi. 2007. *Evolving Lucene Search Queries for Text Classification*. Proceedings of the Genetic and Evolutionary Computation Conference.

Mahesh Joshi and Carolyn P. Rosé. 2009. *Generalizing Dependency Features for Opinion Mining*. Proceedings of the ACL-IJCNLP Conference (Short Papers).

Akihiro Inokuchi, Takashi Washio and Hiroshi Motoda. 2000. *An Apriori-based Algorithm for Mining Frequent Substructures from Graph Data*. Proceedings of PKDD.

Dan Klein and Christopher D. Manning. 2003. *Accurate unlexicalized parsing*. Proceedings of the main conference of the ACL.

John Koza. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

Krzysztof Krawiec. 2002. *Genetic programming-based construction of features for machine learning and knowledge discovery tasks*. Genetic Programming and Evolvable Machines.

Taku Kudo, Eisaku Maeda and Yuji Matsumoto. 2004. *An Application of Boosting to Graph Classification*. Proceedings of NIPS.

Michihiro Kuramochi and George Karypis. 2002. *Frequent Subgraph Discovery*. Proceedings of ICDM.

Christopher D. Manning, Prabhakar Raghavan and Hinrich Schtze. 2008. *Introduction to Information Retrieval*. Proceedings of PAKDD.

Shotaro Matsumoto, Hiroya Takamura and Manabu Okumura. 2005. *Sentiment Classification Using Word Sub-sequences and Dependency Sub-trees*. Proceedings of PAKDD.

Elijah Mayfield and Carolyn Penstein-Rosé. 2010. *Using Feature Construction to Avoid Large Feature Spaces in Text Classification*. Proceedings of the Genetic and Evolutionary Computation Conference.

Fernando Otero, Monique Silva, Alex Freitas and Julio Nievola. 2002. *Genetic Programming for Attribute Construction in Data Mining.* Proceedings of the Genetic and Evolutionary Computation Conference.

Bo Pang, Lillian Lee and Shivakumar Vaithyanathan. 2002. *Thumbs up? Sentiment Classication using Machine Learning Techniques*. Proceedings of EMNLP.

Bo Pang and Lillian Lee. 2004. *A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts*. Proceedings of the main conference of ACL.

Bo Pang and Lillian Lee. 2005. *Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales*. Proceedings of the main conference of ACL.

Jian Pei, Jiawei Han, Behzad Mortazavi-asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal and Mei-chun Hsu. 2004. *Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach*. Proceedings of IEEE Transactions on Knowledge and Data Engineering.

Ellen Riloff, Siddharth Patwardhan and Janyce Wiebe. 2006. *Feature Subsumption for Opinion Analysis*. Proceedings of the EMNLP.

Matthew Smith and Larry Bull. 2005. *Genetic Programming with a Genetic Algorithm for Feature Construction and Selection*. Genetic Programming and Evolvable Machines.

Theresa Wilson, Janyce Wiebe and Rebecca Hwa. 2004. *Just How Mad Are You? Finding Strong and Weak Opinion Clauses*. Proceedings of AAAI.

Theresa Wilson, Janyce Wiebe and Paul Hoffmann. 2005. *Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis*. Proceedings of HLT/EMNLP.

Xifeng Yan and Jiawei Han. 2002. *gSpan: Graph-Based Substructure Pattern Mining*. UIUC Technical Report, UIUCDCS-R-2002-2296 (shorter version in ICDM'02).