Detecting Opinion Polarities using Kernel Methods

Rasoul Kaljahi
ADAPT Centre
School of Computing
Dublin City University, Ireland
rasoul.kaljahi@adaptcentre.ie

Jennifer Foster
ADAPT Centre
School of Computing
Dublin City University, Ireland
jfoster@computing.dcu.ie

Abstract

We investigate the application of kernel methods to representing both structural and lexical knowledge for predicting polarity of opinions in consumer product review. We introduce *any-gram kernels* which model lexical information in a significantly faster way than the traditional n-gram features, while capturing all possible orders of n-grams (n) in a sequence without the need to explicitly present a pre-specified set of such orders. We also modify the traditional tree kernel function to compute the similarity based on word embedding vectors instead of exact string match and present experiments using the new models.

1 Introduction

The automatic identification and analysis of opinion and sentiment in text (Pang and Lee, 2008) has emerged as a major natural language processing task in recent years, due in part to the abundance of opinions now available online. Initially, much of the focus of sentiment analysis research was on detecting the overall sentiment of documents and sentences (Pang et al., 2002). This kind of analysis is insufficient when the sentence or document contains multiple opinions directed towards multiple targets and the goal is to identify each of them individually. For example, a review of a laptop may discuss various features of the product such as battery life, speed and memory. While the review may carry a positive assessment of the laptop in general, the sentiment towards some of these aspects may be negative. *Aspect-based* sentiment analysis (ABSA) (Hu and Liu, 2004) aims to tackle this problem. In this work, we address the problem of aspect-based sentiment analysis and follow recent SemEval shared tasks in using consumer reviews of laptops and restaurants as our test domains.

The majority of machine learning approaches to sentiment analysis have relied on bag-of-n-grams (Pang and Lee, 2008). However, n-grams lead to large sparse feature sets which are not computationally efficient. Moreover, only a limited number of orders of n-grams (i.e. n) can be used and the choice of n requires tuning. To tackle this issue, we introduce *any-gram kernels* which 1) capture all orders of n-grams and 2) are faster while 3) performing at the same level as traditional n-gram features.

Recent research has shown that structural features extracted from syntactic analysis of text can boost the performance of surface-oriented models, by capturing information that these models cannot (Karlgren et al., 2010; Kiritchenko et al., 2014). For example, in *If you like spicy food get the chicken vindaloo.*, a lexicon-based model assigns a positive sentiment to the aspect term *spicy food* due to the nearby presence of *like* , whereas a syntax-based model has the potential to recognise that *like* does not convey an opinion when it is modified by *if*. We use tree kernels to model both the constituency and dependency structure of the sentences. This approach is more efficient than hand-crafted syntactic features, as it requires less engineering effort and is faster to develop.

Traditional tree kernel function computes the similarity of trees based on the exact string match of the node labels including words. This method overlooks the similarity between words which can be used interchangeably in the context. Plank and Moschitti (2013) address this problem by generalizing words

Proceedings of the Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media, pages 60–69, Osaka, Japan, December 12 2016.

using word clusters and latent semantic analysis (LSA). Word embeddings (Bengio et al., 2003), which have been used successfully in tasks involving similarity between words (Collobert and Weston, 2008; Mikolov et al., 2013; Baroni et al., 2014), are an alternative approach to obtain this generalization. We modify the tree kernel function so that the similarity of trees are computed based on the similarity of pre-trained word embedding vectors. We conduct experiments using the new kernel function and find that these similarities are not conclusively more useful than mere string matches.

2 Related Work

Although tree kernels have been previously used in sentiment analysis, no work has directly employed them in ABSA. The closest work has been carried out by Nguyen and Shirai (2015), who use tree kernels to first identify opinion words related to a given aspect term, which are then used in calculating the sentiment score for that aspect term. In sentence-level polarity prediction, Trindade et al. (2013) augment constituency tree kernels by inserting WordNet senses and contextual polarity of words as new nodes under terminal nodes. Agarwal et al. (2011) apply tree kernels with a customized tree format for tweets, instead of using parse trees, where tokens are gathered under a root node together with POS tags and a set of special tags used to represent the types of tokens (e.g. `STOP` for stop words). In document-level sentiment classification, Tu et al. (2012) combine constituency or dependency tree kernels with bag-of-word features. To represent documents, they use several minimal subtrees each of which contains at least one subjective word based on a sentiment lexicon.

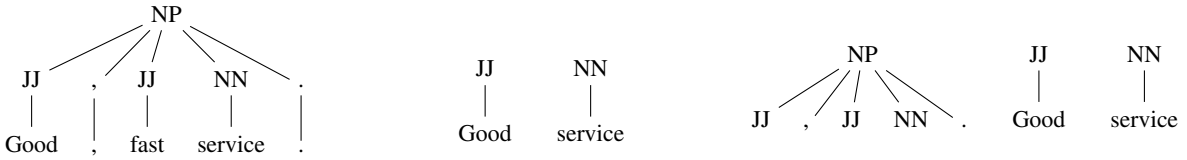
Wiegand and Klakow (2010) employ tree kernels to represent constituency and predicate-argument structure (PAS) in finding opinion holders. They enrich these trees by inserting nodes with generalized concept labels such as location, opinion and person. Their results show that while augmenting the constituency trees is useful, PAS trees do not benefit from extra information. Their best tree kernel setting outperforms their hand-crafted features and their combination leads to a higher performance.

Syntax has also been used in sentiment analysis using hand-crafted features. Johansson and Moschitti (2013) build a set of classifiers and re-rankers for identification of opinion holders, opinion expressions and their polarity in the MPQA corpus (Wiebe et al., 2005). These systems exploit dependency paths within opinion expressions and between opinion holders and opinion expressions. They evaluate these systems extrinsically using a product attribute (aspect) polarity classifier on a product review dataset, which is similar to the task addressed here. This classifier uses syntactic path features from candidate attributes to sentiment words and identified opinion expressions. Dong et al. (2015) introduce a context-free grammar for sentiment in which positive and negative polarity symbols replace syntactic labels in non-terminals. They build a parser which learns this grammar using only sentences annotated with their polarity without any information about their syntactic structure. Socher et al. (2013) build a dataset of movie reviews automatically parsed and manually annotated for the polarity of each constituent. This dataset is then used to compute compositional vector representations of phrases in a neural network framework, which are then used as features in training a model to predict the polarity of each phrase.

Pre-trained word embeddings have previously been used by (Liu et al., 2015) in a similar task of aspect term extraction on the same dataset used here, as initial weights in neural network models and also as features in conditional random field models. Their results show that word embeddings can improve over the baselines of both of these models. To the best of our knowledge, our study is the first to use word embeddings in tree kernel computation.

3 Kernel Methods

Kernel methods provide a means to define custom similarity functions, called *kernel functions*, which can be used by some machine learning algorithms such as support vector machines (SVM) to calculate the similarity between two data points which are not represented as vectors of numbers. Tree kernels (Collins and Duffy, 2002; Moschitti, 2006) are examples of these functions that compute the similarity between two data points represented as trees, based on the number of common fragments between them. Therefore, the need for explicitly encoding an instance in terms of manually designed and extracted features is eliminated, while benefiting from a very high-dimensional feature space. This approach has



(a) Original tree (b) Subtree fragments (c) Subset tree fragments

Figure 1: A sample tree and examples of its *subtree* and *subset* tree fragments

shown to be effective in many NLP tasks including parsing and named entity recognition (Collins and Duffy, 2002), semantic role labelling (Moschitti, 2006), sentiment analysis (see §2) and machine translation quality estimation (Hardmeier et al., 2012; Kaljahi et al., 2014). In the following sections, we describe tree kernels and introduce *any-gram kernels* which are built on top of tree kernels. We also introduce a model which incorporates pre-trained word embedding vectors in tree kernel calculation.

3.1 Tree Kernels

The kernel function applied to tree pair T_1 and T_2 is defined as follows:

$$K(T_1, T_2) = \sum_{n_1 \in \{T_1\ nodes\}} \sum_{n_2 \in \{T_2\ nodes\}} \Delta(n_1, n_2) \tag{1}$$

where Δ calculates the similarity between every two nodes in the tree as follows (Moschitti, 2006):

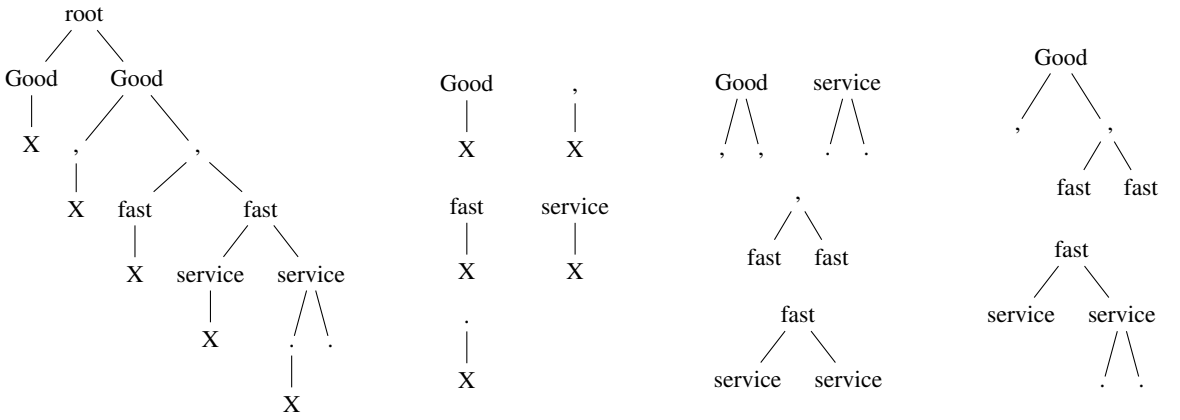
$$\Delta(n_1, n_2) = \begin{cases} 0 \ : \ if\ pr_1 \ \neq \ pr_2 \\ 1 \ : \ if\ pr_1 \ = \ pr_2 \ \ \& \ \ n_1,\ n_2\ are\ pre\text{-}terminals \\ \prod_{j=1}^{nc}(\sigma + \Delta(c_{n_1}^j, c_{n_2}^j)) \ : \ otherwise \end{cases} \tag{2}$$

where pr_1 and pr_2 are the production rules rewriting n_1 and n_2 respectively, nc is the number of children of either node, as they have the same number of children because they have the same production rules, and $c_{n_1}^j$ is the j^{th} child of node n_1. σ controls the type of tree fragments to be used: 0 is for *subtree* fragments which include a node in the tree with the whole sub-tree under it (Figure 1b) and 1 is for *subset* fragments which loosen this constraint by allowing the internal nodes as terminals, resulting in more substructures (Figure 1c). Moschitti (2006) introduces an efficient implementation within a SVM framework, where instead of all node pairs, the Δ function is applied only on similar node pairs.

3.2 Any-gram Kernels

N-gram features have been predominantly used in sentiment analysis and have shown to be very effective (Pang et al., 2002). While unigrams capture the individual sentiment-bearing words, higher orders of n-gram can also capture contextual information. However, there is no clear consensus as to what order of n-gram is the most effective in this task (Pang et al., 2002; Dave et al., 2003). Consequently, various orders should be empirically compared or combined together. Considering the sparsity of these features, especially in the higher orders, these approaches impose high computational costs. We propose *any-gram kernels*, which not only avoids this expense, but also models all orders of n-grams in a sequence. In this model, sentence tokens are arranged in a binary tree in such a way that all possible n-grams are captured by valid tree fragments, which are those extracted by subset tree kernels. Figure 2 shows an example of an n-gram tree and some of the tree fragments extracted from it acting as n-grams.

Adding a dummy `X` node under the left child helps extract unigrams (Figure 2b), as no valid tree fragment can consist of only one node. Bigrams are represented by tree fragments rooted at a two-children node with both of its children (Figure 2c). Trigrams are formed by tree fragments rooted at a two-children node with both of its children, and the children of the right child (Figure 2d). The n-grams for n>3 are represented in a similar way, which are not shown in Figure 2. As can be seen in Figure 2, a dummy *root* node is used to help extract the unigram for the first word of the sentence (*Good*). Note that



(a) N-gram tree (b) Unigram fragments (c) Bigram fragments (d) Trigram fragments

Figure 2: N-gram tree for *Good, fast service.* and examples of its unigram, bigram and trigram fragments

these fragments are valid n-grams despite the duplicate nodes in them because no two different n-grams can be represented by a single such fragment. However, there are different fragments which represent a single n-gram. For example, the *Good-,* bigram is extracted with the fragment shown in Figure 2c as well as another one including the same fragment plus an `X` node under *Good* (not shown in the figure). These duplicates are inevitable but do not noticeably affect the performance. In total, the any-gram tree for a sentence of length N contains $3N + 1$ nodes. Another advantage of any-gram kernels is that other information (e.g. location of aspect term) can be plugged into the tree.

Any-gram kernels can be compared to string kernels (Lodhi et al., 2002) where the n-grams are not explicitly extracted for the learning algorithm but calculated using the string kernel function. The difference is that string kernels require the order of n-grams (n) to be fixed. Also, the tree kernel implementation of the any-gram kernels is faster as the tree kernels are computed in $O(K+M)$ average time, where K and M are the number of nodes of the two trees. When translated to the sentence length, the complexity of the any-gram kernel is $O(|s_1|+|s_2|)$ which is still linear to the sentence lengths $|s_1|$ and $|s_2|$, as the number of nodes in the any-gram tree of sentence s is $3|s|+1$ which is linear to the sentence length. However, the complexity of the string kernels is $O(n|s_1||s_2|)$.

3.3 Tree Kernels with Word Embeddings

The Δ presented in equation 2 computes the similarity of two production rules based on exact string match between the peer nodes in the rules. Consequently, similar but not identical tree fragments such as `JJ->amazing` and `JJ->wonderful` will be ruled out even though they can contribute to the similarity of two trees. Therefore, a mechanism which accounts for the similarity of nodes with different surface forms in this computation may be useful. To this end, we modify the Δ to compute the production rule similarity based on the similarity of the word embedding vectors of their peer node pairs.[1] Formally:

$$\Delta(n_1, n_2) = \begin{cases} 0 \ : \ if \ |pr_1| != |pr_2| \ \ or \ \ \prod_{i=1}^{|pr_1|} t(n_{(pr_1)i}, n_{(pr_2)i}) = 0 \\ 1 \ : \ if \ n_1, \ n_2 \ are \ pre\text{-}terminals \ \ and \ \ \prod_{i=1}^{|pr_1|} t(n_{(pr_1)i}, n_{(pr_2)i}) = 1 \\ \prod_{j=1}^{nc}(\sigma + \Delta(c^j_{n_1}, c^j_{n_2})) \ : \ otherwise \end{cases} \quad (3)$$

where pr_1 and pr_2 are the production rules rewriting n_1 and n_2 respectively, $|pr_1|$ and $|pr_2|$ are the number of nodes in the production rules, $n_{(pr_1)i}$ and $n_{(pr_2)i}$ are the i^{th} peer nodes of the two production rules (e.g. *amazing* and *wonderful* as the 2nd nodes in the example production rules provided above) and t is a threshold function defined as follows:

[1]Obviously, exact match is used for the syntactic labels on the syntactic trees. For simplicity, we do not include this in the formal notation, but it can be easily addressed in the implementation.

	Laptop		Restaurant	
	Train	Test	Train	Test
# sentences	3045	800	3041	800
# aspect terms	2358	654	3693	1134
% positive	42%	52%	59%	65%
% negative	37%	20%	22%	17%
% neutral	19%	26%	17%	17%
% conflict[2]	2%	2%	2%	1%

Table 1: Number of sentences, aspect terms and their polarity distributions in the data sets

$$t(n_1, n_2) = \begin{cases} 0 \ : \ if \ sim(V_{n_1}, V_{n_2}) < \theta \\ 1 \ : \ if \ sim(V_{n_1}, V_{n_2}) \geq \theta \end{cases} \quad (4)$$

where V_{n_1} and V_{n_2} are the word embedding vectors of two input nodes, sim is a vector similarity function and θ is the similarity threshold above which the two nodes are considered equal for the kernel computation. With $\theta = 1$, the kernel value will be equal to the value of the traditional tree kernel.

Since all possible peer node pairs in the production rule pair need to be compared, unlike the traditional tree kernel, the worst-case complexity is increased to $O(N \times M)$, where N and M are the number of nodes in T_1 and T_2 respectively. To partially remedy this situation, we use dynamic programming where we store newly calculated production rule similarity as well as node similarity in a table for later use.

4 Experiments

Data We use the data released for Task 4 of SemEval 2014 (Pontiki et al., 2014) (called SE14 hereafter), which is concerned with ABSA. The data is in the form of consumer reviews from two domains: laptops and restaurants. Table 1 shows various characteristics of the data including the number of sentences and aspect terms and the percentage of each polarity type. As seen in Table 1, the restaurant dataset contains more aspect terms than the laptop ones, as most of its sentences have more than one aspect term. In terms of the polarity class distribution, the *conflict* polarity accounts for only a tiny portion of the aspect terms, whereas the *positive* polarity dominates the datasets except for the laptop training set where it has a similar share as the *negative* polarity. The proportion of *neutral* and *negative* polarities tend to be similar, which is also consistent across the four datasets.

Experiment Details To obtain the syntactic analysis of the data, we parse them into their constituency structures using a PCFG-LA parser (Petrov et al., 2006). The parser is trained on the entire Wall Street Journal section of the *Penn Treebank* (Marcus et al., 1993). We then obtain dependency parses by converting these constituency parses using the `Stanford` converter (de Marneffe and Manning, 2008). To apply tree kernels, we use the `SVMLight-TK` implementation (Moschitti, 2006)[3]. Based on a set of preliminary experiments, we use subset tree kernels and the *one-versus-one* (OVO) method to convert the binary output of the SVM to multi-class (positive, negative, neutral and conflict). The error/margin trade-off of the SVM (C) is tuned using development sets randomly extracted from the official training sets. For tree kernels with word embeddings, we use cosine similarity for sim in equation 4. The θ parameter is tuned on the development set, where the optimum value is selected from $\{0.7, 0.8, 0.9\}$. The pre-trained word vectors used are the publicly available ones trained using *GloVe* (Pennington et al., 2014) trained on 42B-token corpus of *Common Crawl* (1.9M vocabulary) with 300 dimensions.[4]

4.1 Word Any-gram Kernels

We start by modelling the word any-grams using traditional tree kernels, where for each aspect term, we take the any-gram tree formed using all tokens in the sentence in which the aspect term appears, as input

[2]The conflict polarity is used when both positive and negative sentiments are expressed towards the aspect term as in *Waiters are slow but sweet.*

[3]`http://disi.unitn.it/moschitti/Tree-Kernel.htm`

[4]`http://nlp.stanford.edu/projects/glove/`. We chose these word embeddings as they cover a wide range of domains and performed better than the other commonly used ones trained on 100GB Google News corpus using *word2vec*.

	String Match				Word Similarity			
	Laptop		Restaurant		Laptop		Restaurant	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test
Majority	42.67	52.12	60.45	64.19	-	-	-	-
$\texttt{HC}_{ng_{1+2}}$	67.61	60.24	67.59	71.16	-	-	-	-
$\texttt{NGTK}_{w}$	65.04	60.24	67.26	70.72	64.78 (-)	62.08 (+)	68.40 (+)	70.99 (+)
$\texttt{NGTK}_{s}$	62.47	58.41	68.07	74.16	-	-	-	-
$\texttt{NGTK}_{w.s}$	**71.21**	**67.43**	70.99	**75.93**	**69.67 (-)**	**68.50 (+)**	**71.31 (+)**	**75.93 (=)**
$\texttt{SyTK}_{c}$	61.44	58.41	65.96	71.16	62.47 (+)	58.26 (-)	65.64 (-)	71.08 (-)
$\texttt{SyTK}_{c.s}$	68.64	65.44	70.02	**76.54**	67.87 (-)	66.06 (+)	69.69 (-)	75.84 (-)
$\texttt{SyTK}_{d}$	61.70	61.62	65.80	72.13	61.18 (-)	58.41 (-)	66.61 (+)	72.13 (=)
$\texttt{SyTK}_{d.s}$	65.81	65.44	69.37	75.22	65.81 (=)	64.98 (-)	69.37 (=)	74.96 (-)
$\texttt{SyTK}_{cs.ds}$	**68.89**	**67.13**	**71.96**	76.10	**68.38 (-)**	**68.35 (+)**	**70.99 (-)**	**76.46 (+)**
SE14 Best	-	70.48	-	80.95	-	-	-	-

Table 2: Accuracy of majority baseline, hand-crafted (HC) unigram+bigram features, any-gram kernel (NGTK) and syntax tree kernel systems (SyTK) and best SemEval 2014 system, evaluated on the laptop and restaurant development and test sets, based on exact string match and word embedding similarity

to our tree kernel algorithm. An example input tree is similar to the one shown in Figure 2 but with an AT node replacing the X node under the right child corresponding to an aspect term token (*service* in this example). The results are given in Table 2 ($\texttt{NGTK}_{w}$ under *String Match* column. The table also includes the performance of a majority baseline for comparison.

The second row of the table ($\texttt{HC}_{ng_{1+2}}$) contains the performance of an alternative system using hand-crafted unigrams and bigram features, containing 20K and 23K features for the laptop and restaurant datasets respectively. The performance of word any-gram kernels is on a par with the hand-crafted n-gram features (except on the laptop development set), although they are computationally cheaper and require much less engineering effort to select most useful orders and combination of orders of n-grams. We measured the time spent for the classification of both test sets using each model on the same machine. The average of three runs for $\texttt{NGTK}_{w}$ and $\texttt{HC}_{ng_{1+2}}$ were 50 and 490 ms respectively.[5] The comparison also suggest that higher orders of n-gram contained in the any-gram kernels are not useful in this task.

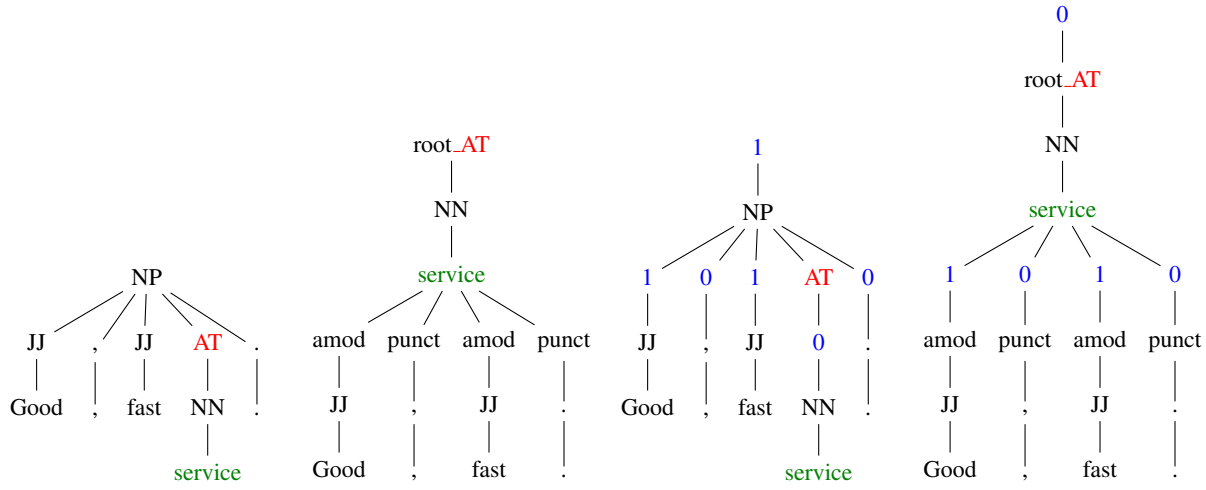
4.2 Constituency Tree Kernels

To use tree kernels for ABSA, the aspect terms need to be marked in the parse tree (Hovy et al., 2013), mainly to differentiate between multiple aspect terms in a sentence and also as information supplied to the algorithm. We tried a set of various formats for this purpose and decided to use one in which a node indicating aspect term (AT) is inserted above the pre-terminal node in the span of aspect term. The results for this format are presented in Table 2 under String Match column ($\texttt{SyTK}_{c}$) and an example tree is shown in Figure 3a. As can be seen, the constituency structure alone tends to be less effective than word n-grams.

4.3 Dependency Tree Kernels

While constituency trees can be readily used as input to tree kernels, dependency trees need to be restructured for this purpose, by moving the dependency labels from the arcs to nodes. We follow the format of Kaljahi et al. (2014), in which the nodes in the resulting trees are word forms and dependency relations, and we also included POS tags as they proved to be useful. In the resulting tree, a word is a child of its POS tag, which is in turn a child of its dependency relation to its head. The dependency relation is in turn the child of the head word. This continues until the root node. Aspect terms are represented by attaching an AT label to the dependency relations. Figure 3b depicts an example tree in this format and Table 2 shows its accuracy ($\texttt{SyTK}_{d}$) under String Match column. Dependency tree kernels tend to outperform the constituency ones. This may be an indication that relationships between words are more important than the hierarchical structure of in which they are arranged for this task.

[5] In fact, training and tuning time are also considerably lower. With the SVM that we use here, tree kernels have only the C parameter to be tuned, but the RBF kernels for hand-crafted features have the C and gamma parameters to be tuned.



(a) Constituency tree (b) Dependency tree (c) a with sentiment scores (d) b with sentiment scores

Figure 3: Sample plain constituency and dependency tree kernel representation for *Good, fast service.* (a and b) and with sentiment scores added (c and d)

4.4 Adding Sentiment Scores

Sentiment lexica assign a score to each word representing the polarity of its sentiment and are often constructed automatically or semi-automatically. We follow Wagner et al. (2014) in constructing a sentiment lexicon which is a combination of four commonly used lexica including *MPQA* (Wilson et al., 2005), *SentiWordNet* (Baccianella et al., 2010), *General Inquirer* [6] and *Opinion Lexicon* (Hu and Liu, 2004). The combined polarity score using their method is in the range [-4,4], where the sign of the score represents the polarity and its value expresses the strength of the sentiment it bears. However, to avoid sparsity, we use a coarse-grained set of three scores $\{-1, 0, 1\}$, for negative, neutral/unknown, positive polarities in the same order, which also turns out to perform better in our experiments.

Starting with the any-gram kernels, we replace the words with their sentiment polarity scores and replicate the experiments. Table 2 shows the performance of the resulting system ($\texttt{NGTK}_s$). According to the results, the sentiment score n-grams are more useful than the word n-grams for the restaurant dataset. However, the opposite seems to be true for the laptop dataset. When this system is combined with the word n-grams, the resulting system ($\texttt{NGTK}_{w.s}$ in Table 2 under String Match) significantly outperforms both of its components. Interestingly, the gain is more significant for the laptop dataset this time.

We now attempt to include sentiment scores in the parse trees. We experiment with various formats, including inserting the scores as nodes or replacing terminals with their scores in the original tree and combining the two trees. We also propagate them upwards in the tree to parent nodes until the root node, by assigning each node the majority score among all its children's positive and negative scores (neutral excluded).[7] In case of ties, the propagated score is set to neutral. Of the formats experimented with, the one with a single tree in which the scores are propagated and inserted as new nodes above their corresponding nodes outperformed others. An example tree is shown in Figure 3c. The propagated score for node `NP` is 1, as there are only two positive scores (1) among all its children (*good* and *fast*). Table 2 shows the accuracy of this setting ($\texttt{SyTK}_{c.s}$ under String Match column). Adding polarity scores substantially increases the performance, although the model complexity does not change significantly.

As in constituency tree kernels, we add sentiment scores to several positions in the dependency trees and find the best performance when they are inserted as nodes above the dependency relation nodes. For instance, in the example tree in Figure 3d, the polarity score for *fast* (1) is inserted as a node above its dependency relation node (`amod`) to its head (*service*). The accuracy obtained using these augmented trees is shown in Table 2 ($\texttt{SyTK}_{d.s}$ under String Match column).

[6] http://www.wjh.harvard.edu/~inquirer/

[7] We also tried using the average children score, which performed marginally lower than the majority score.

Interestingly, constituency and dependency tree kernels tend to perform closely, despite their different structures. While the latter slightly outperforms the former with plain trees, the former benefits more from the polarity scores. One reason can be that more structure is added to the constituency trees than to the dependency trees due to additional nodes for propagated scores.

Finally, we investigate the extent to which the different structures of constituency and dependency tree kernels complement each other by combining $\texttt{SyTK}_{c.s}$ and $\texttt{SyTK}_{d.s}$. The results are shown in Table 2 ($\texttt{SyTK}_{cs.ds}$ under String Match column). The laptop test set and the restaurant development sets benefit the most from the combination, while the other datasets do not see significant changes. This suggests that the complementarity of these two representations is dependent on the data. Compared to the any-gram kernels, the syntactic tree kernels perform slightly better on the restaurant dataset, but are outperformed on the laptop dataset, despite the any-gram kernels carrying less information and being simpler.

4.5 Using Word Embeddings

As described in §3.3 and §4, we replace word forms with word embeddings in kernel computation and use cosine similarity between them instead of exact string match between word forms. The *Word Similarity* column in Table 2 shows the performance of the systems replicated using this method. As can be seen, the changes are inconsistent, but in general the word embedding similarities tend to be helpful for any-gram kernels, contributing from 0.27 to 1.84 percentages of accuracy. However, most of the changes for syntactic tree kernels are negative, the sharpest being 3.21 percent for the laptop test set with $\texttt{SyTK}_{d}$.

Our analysis shows that the optimized similarity threshold tends to be as high as 0.9, while there is only about 500 type token pairs in each domain's dataset which are as similar. Interestingly, an overwhelming number of these pairs involve numbers and stop words. Perhaps, it would be worthwhile to examine word embeddings trained on a corpus in the same domain as the target data set instead of a large but general corpus, or those tuned to better capture sentimental facets of words. For example, with the word embedding used here, the cosine similarity between *good* and *bad* is 77%, which is higher than the similarity score of *superb* and *brilliant* which is 72%.

5 Discussion

Despite their simplicity and efficiency, the models built here achieve reasonably good performance. In fact, our best settings can take the third and fifth place among 31 systems submitted to the SE14 shared task (subtask 2 of task 4) for the laptop and restaurant domains respectively, although our goal here has not been to outperform the state of the art using these systems individually. Table 2 show the performance of the best system (Wagner et al., 2014) submitted to this shared task (*SE14 Best*), which achieves on average 4 points higher accuracy than the best systems built here. This system is built using n-grams and sentiment lexicon features. It combines the output of a rule-based system as features with bag-of-n-gram features. The rule-based system sums the polarity scores of all words around the aspect term in terms of token, discourse chunk and dependency path distance. Their bag-of-n-gram features target the aspect term context and combine word forms with polarity scores and part-of-speech tags. A similar speed test done for any-gram kernels in §4.1 shows that their system is 5 times slower than the tree kernel systems built here. To improve the state of the art, a combination strategy can be sought which effectively exploits the merits of both kinds of approaches.

6 Concluding Remarks

We have presented a series of experiments with tree kernels for aspect-based sentiment analysis and shown that a) tree kernels in the form of *any gram kernels* can be used as an efficient alternative to bag-of-ngrams, b) similarity based on word embeddings does not appear to be obviously superior to simple string match, c) constituency and dependency structure can be fruitfully combined, and d) it is always worth including information from sentiment lexica in the trees. A possible future work is to find methods to effectively combine tree kernels with state-of-the-art hand-crafted features.

Acknowledgements

This research is supported by Science Foundation Ireland in the ADAPT Centre (Grant 13/RC/2106) (`www.adaptcentre.ie`) at Dublin City University.

References

Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. 2011. Sentiment analysis of twitter data. In *Proceedings of the Workshop on Languages in Social Media*, pages 30–38.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. In *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC'10)*.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Michael Collins and Nigel Duffy. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. In *Proceedings of ACL*, pages 263–270.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167.

Kushal Dave, Steve Lawrence, and David M. Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th International Conference on World Wide Web*, pages 519–528.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *Proceedings of the COLING Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8.

Li Dong, Furu Wei, Shujie Liu, Ming Zhou, and Ke Xu. 2015. A statistical parsing framework for sentiment classification. *Computational Linguistics*, 41(2):293–336.

C. Hardmeier, J. Nivre, and J. Tiedemann. 2012. Tree Kernels for Machine Translation Quality Estimation. In *Proceedings of WMT*, pages 109–113.

Dirk Hovy, Shashank Shrivastava, Sujay Kumar Jauhar, Mrinmaya Sachan, Kartik Goyal, Huying Li, Whitney Sanders, and Eduard Hovy. 2013. Identifying metaphorical word use with tree kernels. In *Proceedings of the First Workshop on Metaphor in NLP*, pages 52–57, Atlanta, Georgia, June. Association for Computational Linguistics.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 168–177.

Richard Johansson and Alessandro Moschitti. 2013. Relational features in fine-grained opinion analysis. *Computational Linguistics*, 39(3):473–509.

Rasoul Samed Zadeh Kaljahi, Jennifer Foster, Raphael Rubino, and Johann Roturier. 2014. Quality Estimation of English-French Machine Translation: A Detailed Study of the Role of Syntax. In *Proceedings of COLING*, pages 2052–2063.

Jussi Karlgren, Gunnar Eriksson, Magnus Sahlgren, and Oscar Täckström. 2010. Between bags and trees–constructional patterns in text used for attitude identification. In *Advances in Information Retrieval*, pages 38–49.

Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif Mohammad. 2014. Nrc-canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 437–442.

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1433–1443.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at ICLR*.

Alessandro Moschitti. 2006. Making Tree Kernels practical for Natural Language Learning. In *Proceedings of EACL*, pages 113–120.

Thien Hai Nguyen and Kiyoaki Shirai. 2015. Aspect-based sentiment analysis using tree kernel based relation extraction. In *Computational Linguistics and Intelligent Text Processing - 16th International Conference, CICLing 2015, Proceedings, Part II*, pages 114–125.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, pages 79–86.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning Accurate, Compact and Interpretable Tree Annotation. In *Proceedings of COLING-ACL*.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1498–1507.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, October.

Luis A. Trindade, Hui Wang, William Blackburn, and Niall Rooney. 2013. An enhanced semantic tree kernel for sentiment polarity classification. In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing - Volume 2*, pages 50–62.

Zhaopeng Tu, Yifan He, Jennifer Foster, Josef van Genabith, Qun Liu, and Shouxun Lin. 2012. Identifying High-Impact Sub-Structures for Convolution Kernels in Document-level Sentiment Classification. In *Proceedings of ACL*, pages 338–343.

Joachim Wagner, Piyush Arora, Santiago Cortes, Utsab Barman, Dasha Bogdanova, Jennifer Foster, and Lamia Tounsi. 2014. Dcu: Aspect-based polarity classification for semeval task 4. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 223–229.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 1(2):0.

Michael Wiegand and Dietrich Klakow. 2010. Convolution Kernels for Opinion Holder Extraction. In *Proceedings of NAACL-HLT*, pages 795–803.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 347–354.

Yuanbin Wu, Qi Zhang, Xuangjing Huang, and Lide Wu. 2009. Phrase dependency parsing for opinion mining. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1541, August.