Grammatical structures for word-level sentiment detection

Asad B. Sayeed
MMCI Cluster of Excellence
Saarland University
66123 Saarbrücken, Germany
asayeed@coli.uni-sb.de

Jordan Boyd-Graber, Bryan Rusk, Amy Weinberg
{iSchool / UMIACS, Dept. of CS, CASL}
University of Maryland
College Park, MD 20742 USA
{jbg@umiacs,brusk@,
aweinberg@casl}.umd.edu

Abstract

Existing work in fine-grained sentiment analysis focuses on sentences and phrases but ignores the contribution of individual words and their grammatical connections. This is because of a lack of both (1) annotated data at the word level and (2) algorithms that can leverage syntactic information in a principled way. We address the first need by annotating articles from the information technology business press via crowdsourcing to provide training and testing data. To address the second need, we propose a suffix-tree data structure to represent syntactic relationships between opinion targets and words in a sentence that are opinion-bearing. We show that a factor graph derived from this data structure acquires these relationships with a small number of word-level features. We demonstrate that our supervised model performs better than baselines that ignore syntactic features and constraints.

1 Introduction

The terms "sentiment analysis" and "opinion mining" cover a wide body of research on and development of systems that can automatically infer emotional states from text (after Pang and Lee (2008) we use the two names interchangeably). Sentiment analysis plays a large role in business, politics, and is itself a vibrant research area (Bollen et al., 2010).

Effective sentiment analysis for texts such as newswire depends on the ability to extract who (source) is saying what (target). Fine-grained sentiment analysis requires identifying the sources and targets directly relevant to sentiment bearing expressions (Ruppenhofer et al., 2008). For example, consider the following sentence from a major information technology (IT) business journal:

> Lloyd Hession, chief security officer at BT Radianz in New York, said that virtualization also opens up a slew of potential network access control issues.

There are three entities in the sentence that have the capacity to express an opinion: Lloyd Hession, BT Radianz, and New York. These are potential opinion *sources*. There are also a number of mentioned concepts that could serve as the topic of an opinion in the sentence, or *target*. These include all the sources, but also "virtualization", "network access control", "network", and so on.

The challenging task is to discriminate between these mentions and choose the ones that are relevant to the user. Furthermore, such a system must also indicate the content of the *opinion* itself. This means that we are actually searching for all triples {*source, target, opinion*} in this sentence (Kim and Hovy, 2006) and throughout each document in the corpus. In this case, we want to identify that Lloyd Hession is the *source* of an *opinion*, "slew of network issues," about a *target*, virtualization. Providing such fine-grained annotations would enrich information extraction, question answering, and corpus exploration applications by letting users see who is saying what with what opinion (Wilson et al., 2005; Stoyanov and Cardie, 2006).

We motivate the need for a grammatically-focused approach to fine-grained opinion mining and situate it

2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 667–676,
Montréal, Canada, June 3-8, 2012.

within the context of existing work in Section 2. We propose a supervised technique for learning opinion-target relations from dependency graphs in a way that preserves syntactic coherence and semantic compositionality. In addition to being theoretically sound — a lacuna identified in many sentiment systems[1] — such approaches improve downstream sentiment tasks (Moilanen and Pulman, 2007).

There are multiple types of downstream tasks that potentially require the retrieval of {*source, target, opinion*} relations on a sentence-by-sentence basis. An increasingly significant application area is in the use of large corpora in social science. This area of research requires the exploration and aggregation of data about the relationships between discourses, organizations, and people. For example, the IT business press data that we use in this work belongs to a larger research program (Tsui et al., 2009; Sayeed et al., 2010) of exploring industry opinion leadership. IT business press text is one type of text in which many entities and opinions can appear intermingled with one another in a small amount of text.

Another application for fine-grained sentiment relation retrieval of this type is paraphrasing, where attribution of which opinion belongs to which entities may be important for producing useful and accurate output, since source and target identification errors can change the entire meaning of an output text.

Unlike previous approaches that ignore syntax, we use a sentence's syntactic structure to build a probabilistic model that encodes whether a word is opinion bearing as a latent variable. We build a data structure we call a "syntactic relatedness trie" (Section 3) that serves as the skeleton for a graphical model over the sentiment relevance of words (Section 4). This approach allows us to learn features that predict opinion bearing constructions from grammatical structures. Because of a dearth of resources for this fine-grained task, we also develop new crowdsourcing techniques for labeling word-level, syntactically informed sentiment (Section 5). We use inference techniques to uncover grammatical patterns that connect opinion-expressing words and target entities (Section 6) performing better than using syntactically uninformed methods.

[1] Alm (2011) recently argued that work on sentiment analysis needs to de-emphasize the goal of building systems that are "high-performing" by traditional measures, because the field risks sacrificing "opportunities that may lead to a more thorough understanding of language uses and users" in relation to subjective phenomena. The work we present in this paper therefore focuses on extracting meaningful features as an investment in future work that directly improves retrieval performance.

2 Background and existing work

We call opinion mining "fine-grained" when it retrieves many different {*source, target, opinion*} triples per document. This is particularly challenging when there are multiple triples even within a sentence. There is considerable work on identifying the source of an opinion. However, it is much harder to find obvious features that tell us whether "virtualization" is the target of an opinion. The most recent target identification techniques use machine learning to determine the presence of a target from known opinionated language (Jakob and Gurevych, 2010).

Even when targets are identified we must decide if an opinion is expressed, since not all target mentions will necessarily be accompanied by opinion expressions. Returning to the first example sentence, we could say that the negative opinion about virtualization is expressed by the words "slew" and "issues".

A system that could automatically make this discovery must draw on grammatical relationships between targets and the opinion bearing words. Parsers reveal these relationships, but the relationships are often indirect. The variability of language prevents a complete enumeration of all intervening items that make the relationships indirect, but examples include negation and intensifiers, which change opinion, and sentiment-neutral words, which fill syntactic or stylistic needs. In this paper, we cope with the variability of expression by using supervised machine learning to generalize across observations and learn which features best enable us to identify opinionated language.

Existing work in this area often uses semantic frames and role labeling (Kim and Hovy, 2006; Choi et al., 2006), but resources typically used in these tasks (e.g. FrameNet) are not exhaustive. More general approaches (Ruppenhofer et al., 2008) describe semantic and discourse contexts of opinion sources and targets cannot recognize them.

When techniques do identify targets via syntax, they often only use grammar as a feature in an otherwise syntax-agnostic model. Some work of this

nature merely identifies targets without providing the syntactic evidence necessary to find domain-relevant opinionated language (Jakob and Gurevych, 2010), relying on lists of opinion keywords. There is also work (Qiu et al., 2011) that uses predefined heuristics over dependency parses to identify both targets and opinion keywords but does not acquire new syntactic heuristics. Other work (Nakagawa et al., 2010) is similar to ours in that it uses factor graph modeling over a dependency parse formalism, but it assumes that opinionated language is known *a priori* and focuses on polarity classification, while our work tackles the more fundamental problem of identifying the opinionated language itself.

Little work has been done to perform target and opinion-expression extraction jointly, especially in a way that extracts features for downstream processing. This dearth persists despite evidence that such information improves sentiment analysis (Moilanen and Pulman, 2007).

An advantage of our proposed approach is that we can use dependency paths in order to capture situations where the relations are non-compositional or semantically motivated. In Section 5, we describe a data set that has the additional property that opinion is expressed in ways that require external pragmatic knowledge of the domain. An advantage of arbitrary, non-local dependencies is that we can treat this knowledge as part of the model we learn via long-distance chains, which can capture pragmatics.

3 Syntactic relatedness tries

We now describe how we build the syntactic relatedness trie (SRT) that forms the scaffolding for the probabilistic models needed to identify sentiment-bearing words via syntactic constraints extracted from a dependency parse (Kübler et al., 2009).

We use the Stanford Parser (de Marneffe and Manning, 2008) to produce a dependency graph and consider the resulting undirected graph structure over words. We construct a trie for each possible target word in a sentence (it is possible for a sentence to induce multiple tries if the sentence contains multiple potential targets). Each trie encodes paths from the possible target word to other words, and each path represents a sequence of words connected by undirected edges in the parse.

3.1 Encoding Dependencies in an SRT

SRTs enable us to encode the connections between a single linguistic object of interest—in this application, a possible target word—and a set of related objects. SRTs are data structures consisting of nodes and edges.

This description is very similar to the definition of a dependency parse. The key difference is that while a token only appears once as a node in a dependency parse, an SRT can contain multiple nodes that originate from the same token. This encodes the possible connections between an opinion target and opinion-conveying words.

The object of interest is the opinion target, defined as the SRT root node (e.g. in Figure 1 "policy" is a known target, so it becomes the root of an SRT). Each SRT edge corresponds to a grammatical relationship between words and is labeled with that relationship. We use the notation $a \xrightarrow{\mathrm{R}} b$ to signify that node a has the relationship ("role") R with b. We say in this case that node b is a descendent of node a with the role R. The directed edges constitute a trie or suffix tree that represents the fact that multiple paths may share elements that all provide evidence for the relevance of multiple leaves. [2]

In the remainder of this section we describe the necessary steps to create a training corpus for fine-grained sentiment analysis. We provide an example of how to create an SRT from a dependency parse and then to attach latent variable assignments to an SRT based on human annotations in a way that respects syntactic constraints.

3.2 Using sentiment flow to label an SRT

Our goal is to discriminate between parts of the structure that are relevant to target-opinion word relations and those that are not. We use the term sentiment flow (shortened to "flow" when space is an issue) for relevant sentiment-bearing words in the SRT and inert for the remainder of the sentence. We use the term "flow" because our invariant (section 3.3) constrains a sentiment flow in a SRT to be a contiguous subgraph; this corresponds to linguistic intuitions that, for example, in the sentence "Linux with Wine

[2]The SRT will be used to create an undirected graphical model; the notion of directedness refers to the traversal of paths used to construct the SRT.

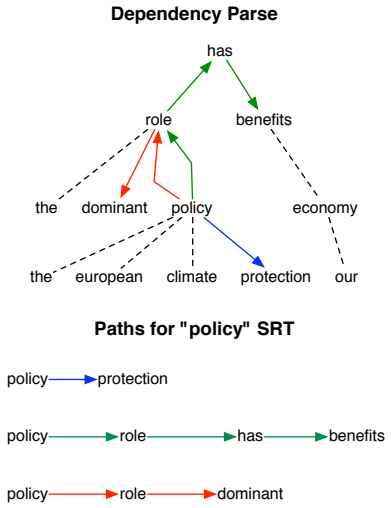

Figure 1: Dependency parse example. A dependency parse (top) is used to generate a syntactic relatedness trie for all possible targets of a sentiment-bearing expression. For the target word "policy", there are a number of paths (colors are consistent in paths to be added to the SRT and in the dependency parse) that connect it to other words; once extracted, these paths will be inserted into a target-specific SRT.

is very usable", {"Linux", "is", "very"} could not be part of a sentiment flow without also including {"usable"}.

Now that we have the structure of the model, we need training data: sentences where sentiment bearing words have been labeled. We describe how to go from sentiment-labeled words to valid flows using this sentence from the MPQA:

> The *dominant* role of the European climate *protection* **policy** has *benefits* for our economy.

In this sentence, the target word "policy" is connected to multiple sentiment-bearing words via paths in the dependency parse (Figure 1). We can represent these relationships using paths through the graph as in Figure 2(a). (For clarity, we do not show some paths.)

Suppose that an annotator decides that "protection" and "benefits" are directly expressing an opinion about the policy, but "dominant" is ambiguous (it has some negative connotations). The nodes "protection" and "benefits" are a flow, and the "dominant"

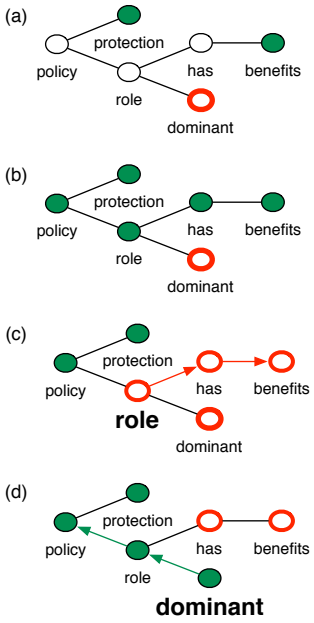

Figure 2: Labeled SRTs rooted on the target word "policy"; green-filled nodes represent words that are part of a sentiment flow and nodes with a red outline represent inert nodes. (a) Initial labels for SRT (e.g. as provided by annotators) (b) propagating labels to yield a valid sentiment flow (c) a change of "role" to inert also renders its children inert (d) a change of "dominant" to be part of a sentiment flow also causes its parents to be part of a flow.

node is inert. However, there is considerable overlap between the "dominant" path and the "benefits" path. That is the motivation for combining them into a trie structure and labeling them in such a way that the path remains a flow until there is *no* path element that leads to a flow leaf (Figure 2).

In other words, we want the path elements common to a flow path and an inert path to reinforce sentiment flow. The transition from flow to inert is learned by the classifier.

We enforce this requirement through the procedure shown in Figure 2, which is equivalent to finding the depth first search tree of the dependency graph and applying the node-labeling scheme as above.

3.3 Invariant

Anything that follows a node with an inert label is by definition not reachable from the root of the tree.

Consequently, any node that is part of a sentiment flow that follows an inert node is not reachable along a path and is actually inert itself. We specify this directly as an invariant on the data structure:

> **Invariant:** no node descending from a node labeled inert can be labeled as a part of a sentiment flow.

This specifies that flow labels spread out from the root of the SRT. Our inference algorithm requires that we be able to change the labels of nodes for test data, thus we need to define invariant-respecting operations for switching labels from flow to flow and vice-versa. A flow label switched to inert will require all the descendents of that particular node to switch to inert as well as in figure 2(c). Similarly, an inert label switched to flow will require all of the ancestors of that node to switch to flow as in 2(d).

4 Encoding SRTs as a factor graph

In this section, we develop supervised machine learning tools to produce a labeled SRT from unlabeled, held-out data in a single, unified model, without permitting the sorts of inconsistencies that may be admitted by using a local classifier at each node.

4.1 Sampling labels

A factor graph (Kschischang et al., 1998) is a representation of a joint probability distribution in the form of a graph with two types of vertices: variable vertices and factor vertices. Given a set of variables $Z = \{z_1 \dots z_n\}$, we connect them via factors $F = \{f_1 \dots f_m\}$. Factors are functions that represent relationships, i.e. probabilistic dependencies, among the variables; the product of all factors gives the complete joint distribution p. Each factor f_i can take as input some corresponding subset of variables Y_i from Z. We can then write the relationship as follows:

$$p(Z) \propto \prod_{k=1}^{m} f_k(Y_k)$$

Our goal is to discover the values for the variables that best explain a dataset. While there are many approaches for inference in statistical models, we turn to MCMC methods (Neal, 1993) to discover the underlying structure of the model. More specifically, we seek a posterior distribution over latent variables that partition words in a sentence into flow and inert groups; we estimate this posterior using Gibbs sampling (Finkel et al., 2005).

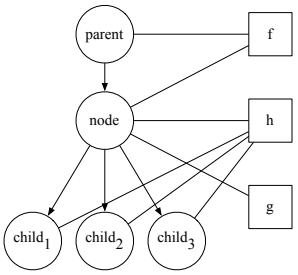

Figure 3: Graphical model of SRT factors

The sampler requires an initial state that respects the invariant. Our initial setting is produced by iterating through all labels in the SRT forest and randomly setting them as either flow or inert with uniform probability.

A Gibbs sampler samples new variable assignments from the conditional distribution, treating the variable assignments for all other variables fixed. However, the assignment of a single node is highly coupled with its neighbors, so a block sampler is used to propose changes to groups nodes that respect the flow labeling of the overall assignments. This was implemented by changing the proposal distribution used by the FACTORIE framework (McCallum et al., 2009).

We can thus represent a node and its contribution to the overall score using the graph in Figure 3. This graph contains the given node, its parent, and a variable number of children. The factors that go into the labeling decision for each node are thus constrained to a small, computationally tractable space around the given node. This graph contains three factors:

- g represents a function over features of the given node itself, or "node features."
- f represents a function over a bigram of features taken from the parent node and the given node, or "parent-node" features.
- h represents a function over a combination features on the node and features of *all* its children, or "node-child" features.

We provide further details about these factors in the next section.

In addition to the latent value associated with each word, we associate each node with features derived from the dependency parse: the word from the sentence itself, the part-of-speech (POS) tag assigned by the Stanford parser, and the label of the incoming dependency edge. We treat the edge labels from the original dependency parse as a feature of the node.

We can represent the set of possible observed linguistic feature classes as the set of features Φ. Figure 3 induces a scoring function with contributions of each node to the score(label|node) =

$$\prod_{\phi \in \Phi} \Big(f(\text{parent}_\phi, \text{node}_\phi | \text{label}) g(\text{node}_\phi | \text{label}) \\ h(\text{node}_\phi, \text{child}_{1\phi}, \ldots, \text{child}_{n\phi} | \text{label}) \Big) .$$

After assignments for the latent variables are sampled, the weights for the factors (which when combined create individual factors f that define the joint) must be learned. This is accomplished via the *sample-rank* algorithm (Wick et al., 2009).

5 Data source

Our goal is to identify opinion-bearing words and targets using supervised machine learning techniques. Sentiment corpora with sub-sentential annotations, such as the Multi-Perspective Question-Answering (MPQA) corpus (Wilson and Wiebe, 2005) and the J. D. Power and Associates (JDPA) blog post corpus (Kessler et al., 2010), exist, but most of these annotations are at a phrase level. Within a phrase, however, some words may contribute more than others to the statement of an opinion. We developed our own annotations to discover such distinctions[3]. We describe these briefly here; more information about the development of the data source can be found in Sayeed et al. (2011).

5.1 Information technology business press

Our work is part of a larger collaboration with social scientists to study the diffusion of information technology (IT) innovations through society by identifying opinion leaders and IT-relevant opinionated language Rogers (2003). Thus, we focus on a collection of articles from the IT professional magazine, *Information Week*, from the years 1991 to 2008. This consists of 33K articles including news bulletins and opinion columns. Our IT concept target list (59 terms) comes from our application. Thus, we construct a trie for each appearance of any of these possible target terms. We consider this list of target terms to be complete, which allows us to focus on discovering opinion-bearing text associated with these targets.

[3]To download the corpus, visit `http://www.umiacs.umd.edu/~asayeed/naacl12data/`.

5.2 Crowdsourced annotation process

Our process for obtaining gold standard data involves multiple levels of human annotation including on crowdsourcing platforms Hsueh et al. (2009).

There are 75K sentences with IT concept mentions, only a minority of which express relevant opinions. Hired undergraduate students searched a random selection of these sentences and found 219 that contain these opinions. We used cosine-similarity to rank the remaining sentences against the 219.

We then needed to identify which of the words contained an opinion. We excluded all words that were common function words (e.g.,"the", "in") but left negations. We engineered tasks so that only a randomly-selected five or six words appear highlighted for classification in order to limit annotator boredom. We called this group a "highlight group". The virtualization example would look like this:

> Lloyd Hession, chief *security* officer at BT Radianz in New York, said that **virtualization** also *opens* up a slew of potential *network access* control *issues*.

In the virtualization example, the worker would see that **virtualization** is highlighted as the IT concept target. Other words are highlighted as candidates that the worker must classify as being opinion-relevant to "virtualization". Each highlight group corresponds to a syntactic relatedness trie (Section 3).

A task was presented to a worker in the form of a highlight group and some list boxes that represent classes for the highlighted words: "positive", "negative", "not opinion-relevant", and "ambiguous". The worker was required to drag each highlighted candidate word to exactly one of the boxes. As we are not doing opinion polarity classification, the "positive" and "negative" boxes were intended as a form of misdirection intended to avoid having the worker consider what an opinion is; we treated this input as a single "opinion-relevant" category.

Three or more users annotated each highlight group, and an aggregation scheme was applied afterwards: "ambiguous" answers were rolled into "not opinion-relevant" and ties were dropped. Our quality control process involved filtering out workers who performed poorly on a small subset of gold-standard answers We annotated 30 evaluation units to determine that our process retrieved opinion-relevant words at 85% precision and 74% recall.

Annotators labeled 700 highlight groups for the results in this paper. The total cost of this exercise was approximately 250 USD, which includes the fees charged by Amazon and CrowdFlower. These last highlight groups were converted to SRTs and divided into training and testing groups, 465 and 196 SRTs respectively, with a small number lost to fatal errors in the Stanford parser.

6 Experiments and discussion

During the training phase, we evaluate the quality of a candidate labeling based on label accuracy. We need to identify both flow nodes and inert nodes in order to distinguish between relevant and irrelevant subcomponents. We thus also employ precision and recall as performance metrics.

An example of how this works can be seen by comparing figure 2(b) to figure 2(d), viewing the former as the gold standard and the latter as a hypothetical system output. If we run the evaluation over that single SRT and treat flow as the positive class, we find that 3 true positives, 1 false positive, 2 false negatives, and no true negatives. There are 6 labels in total. That yields 0.50 accuracy, 0.75 precision, 0.60 recall, and 0.67 F-measure.

We run every experiment (training a model and testing on held-out data) 10 times and take the mean average and range of all measures. F-measure is calculated for each run and averaged *post hoc*.

6.1 Experiments

Our baseline system is the initial setting of the labels for the sampler: uniform random assignment of flow labels, respecting the invariant. This leads to a large class imbalance in favor of inert as any switch to inert converts all nodes downstream from the root to convert to inert, while a switch to flow causes only one ancestor branch to convert to flow.

Our next systems involve combinations of our SRT factors with the observed linguistic features. All our experiments include the factor g that pertains only to the features of the node. Then we add factor f—the parent-node "bigram" features—and finally factor h, the variable-length node-child features. We also experiment with including and excluding combinations of POS, role, and word features. We also explored models that only made local decisions, ignoring the consistency constraints over sentiment flows. Although such models cannot be used in techniques such as Nakagawa et al.'s polarity classifier, they function as a baseline and inform whether syntactic constraints help performance.

We ran the inferencer for 200 iterations to train a model with a particular factor-feature combination. We use the learned model to predict the labels on the held-out testing data by running the inference algorithm (sampling labels only) for 50 iterations.

6.2 Discussion

We present a sampling of possible feature-factor combinations in table 1 in order to show trends in the performance of the system.

Unsurprisingly, the invariant-respecting baseline had very high precision but low recall. Simply including the node-only g factor with all features increases the recall while hurting precision. On removing word features, recall increases without changing precision. This suggests that some words in some SRTs are associated with flow labels in the training data, but not as much in the testing data.

Including parent-node f features with the g features yields higher precision and lower recall, suggesting that parent-node word features support precision. Including all features on all factors (f, g, and h) preserves most of the precision but improves recall. Excluding h features increases recall slightly more than it hurts precision. Excluding both word features for all factors and role h features hurts all measures.

The accuracy measure, however, does show overall improvement with the inclusion of more feature-factor combinations. In particular, the node-child h factor does appear to have an effect on the performance. The presence of some combinations of child word, POS tags, and roles appear to provide some indication of the flow labeling of some of the nodes. The best models in terms of accuracy include all or

Experiment	Features	Invariant?	Precision	Recall	F	Accuracy
Baseline	N/A	Yes	0.78 ± 0.05	0.06 ± 0.01	0.11 ± 0.02	0.51 ± 0.01
		No	0.50 ± 0.00	0.49 ± 0.00	0.50 ± 0.00	0.50 ± 0.00
Node only	All	Yes	0.63 ± 0.10	0.34 ± 0.10	0.42 ± 0.07	0.54 ± 0.03
		No	0.51 ± 0.00	0.88 ± 0.03	0.65 ± 0.01	0.51 ± 0.01
	All but word	Yes	0.63 ± 0.16	0.40 ± 0.22	0.42 ± 0.19	0.53 ± 0.03
		No	0.57 ± 0.04	0.56 ± 0.17	0.55 ± 0.07	0.55 ± 0.03
Parent, node	Parent: all but word Node: all	Yes	0.71 ± 0.06	0.21 ± 0.04	0.31 ± 0.05	0.55 ± 0.01
	All	Yes	0.84 ± 0.07	0.11 ± 0.04	0.19 ± 0.06	0.53 ± 0.01
Full graph	Parent: all but word Node: all but word Children: POS only	Yes	0.59 ± 0.06	0.39 ± 0.11	0.46 ± 0.07	0.54 ± 0.03
	Parent: all Node: all Children: all but word	Yes	0.67 ± 0.05	0.39 ± 0.08	0.47 ± 0.06	0.59 ± 0.02
	All	Yes	0.70 ± 0.05	0.35 ± 0.08	0.46 ± 0.07	0.59 ± 0.02
		No	0.70 ± 0.03	0.20 ± 0.05	0.36 ± 0.06	0.56 ± 0.01

Table 1: Performance using different feature combinations, including some without enforcing the invariant. Mean averages and standard deviation for 10 runs.

almost all of the features.

Our non-invariant-respecting baseline unsurprisingly was nearly 50% on all measures. Including the node-only features dramatically increases recall, less if we exclude word features. The word features appear to have an effect on recall just as in the invariant-respecting case with node-only features. With all features, precision is dramatically improved, but with a large cost to recall. However, it underperforms the equivalent invariant-respecting model in recall, F-measure, and accuracy.

Though these invariant-violating models are unconstrained in the way they label the graph, our invariant-respecting models still outperform them. A coherent path contains more information than an incoherent one; it is important to find negating and intensifying elements *in context*. Our SRT invariant allows us to achieve better performance and will be more useful to downstream tasks.

Finally, it appears that using more factors and linguistic features promotes stability in performance and decreases sensitivity to the initial setting.

6.3 Manual inspection

One pattern that prominently stood out in the testing data with the full-graph model was the misclassification of flow labels as inert in the vicinity of Stanford dependency labels such as `conj_and`. These kinds of labels have high "fertility"; the labels immediately following them in the SRT could be a variety of types, creating potential data sparsity issues.

This problem could be resolved by making some features transparent to the learner. For example, if node q has an incoming `conj_and` dependency edge label, then q's parent could also be directly connected to q's children, as a conjunction should be linguistically transparent to the status of the children in the sentiment flow.

There are many fewer incidents of inert labels being classified as flow. There are paths through an SRT where a flow candidate word is the ancestor of an inert candidate word from the set of crowdsourced candidates. The model sometimes appears to "overshoot" the flow candidate. Considering that recall is already fairly low, attempts to address this problem risks making the model too conservative. One potential solution is to prune or separate paths that contain multiple flow candidates.

6.3.1 Paths found

We examined the labeling on the held-out testing data of the best-performing model of the full graph system with all linguistic features. For example, consider the following highlight group:

> But Microsoft's informal approach may not be enough as the number of **blogs** at the company grows, especially since the line between "personal" Weblogs and those done as part of the job can be hard to distinguish.

In this case, the Turkers decided that "distinguish" expressed a negative opinion about blogs, in the sense

that something that was difficult to distinguish was a problem: the modifier "hard" is what makes it negative. The system found an entirely flow path that connected these attributes into a single unit:

Blog:flow $\xrightarrow{\text{prep}_{\text{o}}\text{f}}$ number:flow $\xrightarrow{\text{nsubj}}$
grows:flow $\xrightarrow{\text{ccomp}}$ hard:flow $\xrightarrow{\text{xcomp}}$
distinguish:flow

In this path, "blog" and "distinguish" are both connected to one another by "hard", giving "distinguish" its negative spin. There are two non-local dependencies in this example: xcomp, ccomp. Very often, more than one unique path connects the concept to the opinion candidate word.

7 Conclusions and future work

In this work, we have applied machine learning to produce a robust modeling of syntactic structure for an information extraction application. A solution to the problem of modeling these structures requires the development of new techniques that model complex linguistic relationships in an application-dependent way. We have shown that we can mine these relationships without being overcome by the data-sparsity issues that typically stymie learning over complex linguistic structure.

The limitations on these techniques ultimately find their root in the difficulty in modeling complex syntactic structures that simultaneously exclude irrelevant portions of the structure while maintaining connected relations. Our technique uses a structure-labelling scheme that enforces connectedness. Enforcing connected structure is not only necessary to produce useful results but also to improve accuracy.

Further performance gains might be possible by enriching the feature set. For example, the POS tagset used by the Stanford parser contains multiple verb tags that represent different English tenses and numbers. For the purpose of sentiment relations, it is possible that the differences between verb tags are too small to matter and are causing data sparsity issues. Thus, we could additional features that "back off" to general verb tags.

Acknowledgements

This paper is based upon work supported by the US National Science Foundation under Grant IIS-0729459. Additional support came from the Cluster of Excellence "Multimodal Computing and Innovation", Germany. Jordan Boyd-Graber is also supported by US National Science Foundation Grant NSF grant #1018625 and the Army Research Laboratory through ARL Cooperative Agreement W911NF-09-2-0072. Any opinions, findings, conclusions, or recommendations expressed are the authors' and do not necessarily reflect those of the sponsors.

References

Alm, C. O. (2011). Subjective natural language problems: Motivations, applications, characterizations, and implications. In *ACL (Short Papers)*.

Bollen, J., Mao, H., and Zeng, X.-J. (2010). Twitter mood predicts the stock market. *CoRR*, abs/1010.3003.

Choi, Y., Breck, E., and Cardie, C. (2006). Joint extraction of entities and relations for opinion recognition. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

de Marneffe, M.-C. and Manning, C. D. (2008). The stanford typed dependencies representation. In *CrossParser '08: Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, Morristown, NJ, USA. Association for Computational Linguistics.

Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hsueh, P.-Y., Melville, P., and Sindhwani, V. (2009). Data quality from crowdsourcing: a study of annotation selection criteria. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, HLT '09, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jakob, N. and Gurevych, I. (2010). Extracting opinion targets in a single and cross-domain setting with conditional random fields. In *EMNLP*.

Kessler, J. S., Eckert, M., Clark, L., and Nicolov, N. (2010). The 2010 ICWSM JDPA sentment

corpus for the automotive domain. In *4th Int'l AAAI Conference on Weblogs and Social Media Data Workshop Challenge (ICWSM-DWC 2010)*.

Kim, S.-M. and Hovy, E. (2006). Extracting opinions, opinion holders, and topics expressed in online news media text. In *SST '06: Proceedings of the Workshop on Sentiment and Subjectivity in Text*, pages 1–8, Morristown, NJ, USA. Association for Computational Linguistics.

Kschischang, F. R., Frey, B. J., and andrea Loeliger, H. (1998). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47:498–519.

Kübler, S., McDonald, R., and Nivre, J. (2009). Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 2(1).

McCallum, A., Schultz, K., and Singh, S. (2009). Factorie: Probabilistic programming via imperatively defined factor graphs. In *Neural Information Processing Systems (NIPS)*.

Moilanen, K. and Pulman, S. (2007). Sentiment composition. In *Proceedings of the Recent Advances in Natural Language Processing International Conference (RANLP-2007)*, Borovets, Bulgaria.

Nakagawa, T., Inui, K., and Kurohashi, S. (2010). Dependency tree-based sentiment classification using crfs with hidden variables. In *HLT-NAACL*.

Neal, R. M. (1993). Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto.

Pang, B. and Lee, L. (2008). Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2).

Qiu, G., Liu, B., Bu, J., and Chen, C. (2011). Opinion word expansion and target extraction through double propagation. *Computational linguistics*, 37(1):9—27.

Rogers, E. M. (2003). *Diffusion of Innovations, 5th Edition*. Free Press.

Ruppenhofer, J., Somasundaran, S., and Wiebe, J. (2008). Finding the sources and targets of subjective expressions. In Calzolari, N., Choukri, K., Maegaard, B., Mariani, J., Odjik, J., Piperidis, S., and Tapias, D., editors, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Sayeed, A. B., Nguyen, H. C., Meyer, T. J., and Weinberg, A. (2010). Expresses-an-opinion-about: using corpus statistics in an information extraction approach to opinion mining. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10.

Sayeed, A. B., Rusk, B., Petrov, M., Nguyen, H. C., Meyer, T. J., and Weinberg, A. (2011). Crowdsourcing syntactic relatedness judgements for opinion mining in the study of information technology adoption. In *Proceedings of the Association for Computational Linguistics 2011 workshop on Language Technology for Cultural Heritage, Social Sciences, and the Humanities (LaTeCH)*. Association for Computational Linguistics.

Stoyanov, V. and Cardie, C. (2006). Partially supervised coreference resolution for opinion summarization through structured rule learning. In *EMNLP '06: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 336–344, Morristown, NJ, USA. Association for Computational Linguistics.

Tsui, C.-J., Wang, P., Fleischmann, K., Oard, D., and Sayeed, A. (2009). Understanding IT innovations by computational analysis of discourse. In *International conference on information systems*.

Wick, M., Rohanimanesh, K., Culotta, A., and Mccallum, A. (2009). SampleRank: Learning preference from atomic gradients. In *NIPS WS on Advances in Ranking*.

Wilson, T. and Wiebe, J. (2005). Annotating attributions and private states. In *CorpusAnno '05: Proceedings of the Workshop on Frontiers in Corpus Annotations II*, Morristown, NJ, USA. Association for Computational Linguistics.

Wilson, T., Wiebe, J., and Hoffmann, P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT/EMNLP*.