Opinion Mining on YouTube

Aliaksei Severyn[1], **Alessandro Moschitti**[3,1],
Olga Uryupina[1], **Barbara Plank**[2], **Katja Filippova**[4]
[1]DISI - University of Trento, [2]CLT - University of Copenhagen,
[3]Qatar Computing Research Institute, [4]Google Inc.
severyn@disi.unitn.it, amoschitti@qf.org.qa,
uryupina@gmail.com, bplank@cst.dk, katjaf@google.com

Abstract

This paper defines a systematic approach to Opinion Mining (OM) on YouTube comments by (i) modeling classifiers for predicting the opinion polarity and the type of comment and (ii) proposing robust shallow syntactic structures for improving model adaptability. We rely on the tree kernel technology to automatically extract and learn features with better generalization power than bag-of-words. An extensive empirical evaluation on our manually annotated YouTube comments corpus shows a high classification accuracy and highlights the benefits of structural models in a cross-domain setting.

1 Introduction

Social media such as Twitter, Facebook or YouTube contain rapidly changing information generated by millions of users that can dramatically affect the reputation of a person or an organization. This raises the importance of automatic extraction of sentiments and opinions expressed in social media.

YouTube is a unique environment, just like Twitter, but probably even richer: multi-modal, with a social graph, and discussions between people sharing an interest. Hence, doing sentiment research in such an environment is highly relevant for the community. While the linguistic conventions used on Twitter and YouTube indeed show similarities (Baldwin et al., 2013), focusing on YouTube allows to exploit context information, possibly also multi-modal information, not available in isolated tweets, thus rendering it a valuable resource for the future research.

Nevertheless, there is almost no work showing effective OM on YouTube comments. To the best of our knowledge, the only exception is given by the classification system of YouTube comments proposed by Siersdorfer et al. (2010).

While previous state-of-the-art models for opinion classification have been successfully applied to traditional corpora (Pang and Lee, 2008), YouTube comments pose additional challenges: (i) polarity words can refer to either video or product while expressing contrasting sentiments; (ii) many comments are unrelated or contain spam; and (iii) learning supervised models requires training data for each different YouTube domain, e.g., *tablets*, *automobiles*, etc. For example, consider a typical comment on a YouTube review video about a *Motorola Xoom* tablet:

> *this guy really puts a negative spin on this , and I 'm not sure why , this seems crazy fast , and I 'm not entirely sure why his pinch to zoom his laggy all the other* ***xoom*** *reviews*

The comment contains a product name *xoom* and some negative expressions, thus, a bag-of-words model would derive a negative polarity for this product. In contrast, the opinion towards the product is neutral as the negative sentiment is expressed towards the video. Similarly, the following comment:

> *iPad 2 is better. the superior apps just destroy the* ***xoom****.*

contains two positive and one negative word, yet the sentiment towards the product is negative (the negative word *destroy* refers to *Xoom*). Clearly, the bag-of-words lacks the structural information linking the sentiment with the target product.

In this paper, we carry out a systematic study on OM targeting YouTube comments; its contribution is three-fold: firstly, to solve the problems outlined above, we define a classification schema, which separates spam and not related comments from the informative ones, which are, in turn, further categorized into video- or product-related comments

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1252–1261,
Baltimore, Maryland, USA, June 23-25 2014. ©2014 Association for Computational Linguistics

(type classification). At the final stage, different classifiers assign polarity (positive, negative or neutral) to each type of a meaningful comment. This allows us to filter out irrelevant comments, providing accurate OM distinguishing comments about the video and the target product.

The second contribution of the paper is the creation and annotation (by an expert coder) of a comment corpus containing 35k manually labeled comments for two product YouTube domains: *tablets* and *automobiles*.[1] It is the first manually annotated corpus that enables researchers to use supervised methods on YouTube for comment classification and opinion analysis. The comments from different product domains exhibit different properties (cf. Sec. 5.2), which give the possibility to study the domain adaptability of the supervised models by training on one category and testing on the other (and vice versa).

The third contribution of the paper is a novel structural representation, based on shallow syntactic trees enriched with conceptual information, i.e., tags generalizing the specific topic of the video, e.g., *iPad*, *Kindle*, *Toyota Camry*. Given the complexity and the novelty of the task, we exploit structural kernels to automatically engineer novel features. In particular, we define an efficient tree kernel derived from the Partial Tree Kernel, (Moschitti, 2006a), suitable for encoding structural representation of comments into Support Vector Machines (SVMs). Finally, our results show that our models are adaptable, especially when the structural information is used. Structural models generally improve on both tasks – polarity and type classification – yielding up to 30% of relative improvement, when little data is available. Hence, the impractical task of annotating data for each YouTube category can be mitigated by the use of models that adapt better across domains.

2 Related work

Most prior work on more general OM has been carried out on more standardized forms of text, such as consumer reviews or newswire. The most commonly used datasets include: the MPQA corpus of news documents (Wilson et al., 2005), web customer review data (Hu and Liu, 2004), Amazon review data (Blitzer et al., 2007), the JDPA corpus of blogs (Kessler et al., 2010), etc. The aforementioned corpora are, however, only partially suitable for developing models on social media, since the informal text poses additional challenges for Information Extraction and Natural Language Processing. Similar to Twitter, most YouTube comments are very short, the language is informal with numerous accidental and deliberate errors and grammatical inconsistencies, which makes previous corpora less suitable to train models for OM on YouTube. A recent study focuses on sentiment analysis for Twitter (Pak and Paroubek, 2010), however, their corpus was compiled automatically by searching for emoticons expressing positive and negative sentiment only.

Siersdorfer et al. (2010) focus on exploiting user ratings (counts of ‘thumbs up/down’ as flagged by other users) of YouTube video comments to train classifiers to predict the community acceptance of new comments. Hence, their goal is different: predicting comment ratings, rather than predicting the sentiment expressed in a YouTube comment or its information content. Exploiting the information from user ratings is a feature that we have not exploited thus far, but we believe that it is a valuable feature to use in future work.

Most of the previous work on supervised sentiment analysis use feature vectors to encode documents. While a few successful attempts have been made to use more involved linguistic analysis for opinion mining, such as dependency trees with latent nodes (Täckström and McDonald, 2011) and syntactic parse trees with vectorized nodes (Socher et al., 2011), recently, a comprehensive study by Wang and Manning (2012) showed that a simple model using bigrams and SVMs performs on par with more complex models.

In contrast, we show that adding structural features from syntactic trees is particularly useful for the cross-domain setting. They help to build a system that is more robust across domains. Therefore, rather than trying to build a specialized system for every new target domain, as it has been done in most prior work on domain adaptation (Blitzer et al., 2007; Daumé, 2007), the domain adaptation problem boils down to finding a more robust system (Søgaard and Johannsen, 2012; Plank and Moschitti, 2013). This is in line with recent advances in parsing the web (Petrov and McDonald, 2012), where participants where asked to build a single system able to cope with different yet re-

[1] The corpus and the annotation guidelines are publicly available at: http://projects.disi.unitn.it/iKernels/projects/sentube/

lated domains.

Our approach relies on robust syntactic structures to automatically generate patterns that adapt better. These representations have been inspired by the semantic models developed for Question Answering (Moschitti, 2008; Severyn and Moschitti, 2012; Severyn and Moschitti, 2013) and Semantic Textual Similarity (Severyn et al., 2013). Moreover, we introduce additional tags, e.g., video concepts, polarity and negation words, to achieve better generalization across different domains where the word distribution and vocabulary changes.

3 Representations and models

Our approach to OM on YouTube relies on the design of classifiers to predict comment type and opinion polarity. Such classifiers are traditionally based on bag-of-words and more advanced features. In the next sections, we define a baseline feature vector model and a novel structural model based on kernel methods.

3.1 Feature Set

We enrich the traditional bag-of-word representation with features from a sentiment lexicon and features quantifying the negation present in the comment. Our model (FVEC) encodes each document using the following feature groups:

- **word n-grams**: we compute unigrams and bigrams over lower-cased word lemmas where binary values are used to indicate the presence/absence of a given item.
- **lexicon**: a sentiment lexicon is a collection of words associated with a positive or negative sentiment. We use two manually constructed sentiment lexicons that are freely available: the MPQA Lexicon (Wilson et al., 2005) and the lexicon of Hu and Liu (2004). For each of the lexicons, we use the number of words found in the comment that have *positive* and *negative* sentiment as a feature.
- **negation**: the count of negation words, e.g., {*don't, never, not, etc.*}, found in a comment.[2] Our structural representation (defined next) enables a more involved treatment of negation.
- **video concept**: cosine similarity between a comment and the title/description of the video. Most of the videos come with a title and a short description, which can be used to encode the topicality of each comment by looking at their overlap.

3.2 Structural model

We go beyond traditional feature vectors by employing structural models (STRUCT), which encode each comment into a shallow syntactic tree. These trees are input to tree kernel functions for generating structural features. Our structures are specifically adapted to the noisy user-generated texts and encode important aspects of the comments, e.g., words from the sentiment lexicons, product concepts and negation words, which specifically targets the sentiment and comment type classification tasks.

In particular, our shallow tree structure is a two-level syntactic hierarchy built from word lemmas (leaves) and part-of-speech tags that are further grouped into chunks (Fig. 1). As full syntactic parsers such as constituency or dependency tree parsers would significantly degrade in performance on noisy texts, e.g., Twitter or YouTube comments, we opted for shallow structures, which rely on simpler and more robust components: a part-of-speech tagger and a chunker. Moreover, such taggers have been recently updated with models (Ritter et al., 2011; Gimpel et al., 2011) trained specifically to process noisy texts showing significant reductions in the error rate on user-generated texts, e.g., Twitter. Hence, we use the CMU Twitter pos-tagger (Gimpel et al., 2011; Owoputi et al., 2013) to obtain the part-of-speech tags. Our second component – chunker – is taken from (Ritter et al., 2011), which also comes with a model trained on Twitter data[3] and shown to perform better on noisy data such as user comments.

To address the specifics of OM tasks on YouTube comments, we enrich syntactic trees with semantic tags to encode: (i) central concepts of the video, (ii) sentiment-bearing words expressing *positive* or *negative* sentiment and (iii) negation words. To automatically identify concept words of the video we use context words (tokens detected as nouns by the part-of-speech tagger) from the video title and video description and match them in the tree. For the matched words, we enrich labels of their parent nodes (part-of-speech and chunk) with the PRODUCT tag. Similarly, the nodes associated with words found in

[2] The list of negation words is adopted from http://sentiment.christopherpotts.net/lingstruc.html

[3] The chunker from (Ritter et al., 2011) relies on its own POS tagger, however, in our structural representations we favor the POS tags from the CMU Twitter tagger and take only the chunk tags from the chunker.

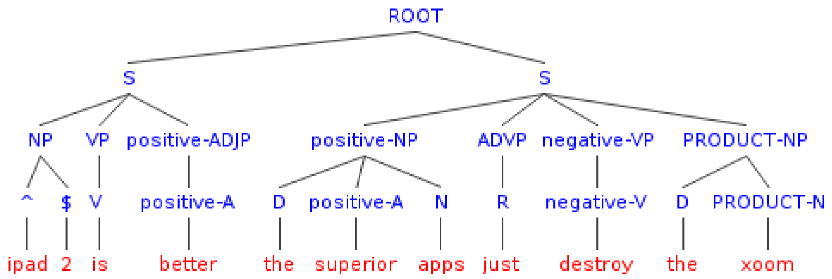


Figure 1: Shallow tree representation of the example comment (labeled with `product` type and `negative` sentiment): *"iPad 2 is better. the superior apps just destroy the xoom."* (lemmas are replaced with words for readability) taken from the video "Motorola Xoom Review". We introduce additional tags in the tree nodes to encode the central concept of the video (motorola *xoom*) and sentiment-bearing words *(better, superior, destroy)* directly in the tree nodes. For the former we add a `PRODUCT` tag on the chunk and part-of-speech nodes of the word *xoom*) and polarity tags (*positive* and *negative*) for the latter. Two sentences are split into separate root nodes `S`.

the sentiment lexicon are enriched with a polarity tag (either *positive* or *negative*), while negation words are labeled with the `NEG` tag. It should be noted that vector-based (`FVEC`) model relies only on feature counts whereas the proposed tree encodes powerful contextual syntactic features in terms of tree fragments. The latter are automatically generated and learned by SVMs with expressive tree kernels.

For example, the comment in Figure 1 shows two *positive* and one *negative* word from the sentiment lexicon. This would strongly bias the `FVEC` sentiment classifier to assign a `positive` label to the comment. In contrast, the `STRUCT` model relies on the fact that the negative word, *destroy*, refers to the PRODUCT (*xoom*) since they form a verbal phase (VP). In other words, the tree fragment: `[S [negative-VP [negative-V [destroy]] [PRODUCT-NP [PRODUCT-N [xoom]]]]` is a strong feature (induced by tree kernels) to help the classifier to discriminate such hard cases. Moreover, tree kernels generate all possible subtrees, thus producing generalized (back-off) features, e.g., `[S [negative-VP [negative-V [destroy]] [PRODUCT-NP]]]]` or `[S [negative-VP [PRODUCT-NP]]]]`.

3.3 Learning

We perform OM on YouTube using supervised methods, e.g., SVM. Our goal is to learn a model to automatically detect the sentiment and type of each comment. For this purpose, we build a multiclass classifier using the one-vs-all scheme. A binary classifier is trained for each of the classes and the predicted class is obtained by taking a class from the classifier with a maximum prediction score. Our back-end binary classifier is SVM-light-TK[4], which encodes structural kernels in the SVM-light (Joachims, 2002) solver. We define a novel and efficient tree kernel function, namely, **Sh**allow syntactic **T**ree **K**ernel (SHTK), which is as expressive as the Partial Tree Kernel (PTK) (Moschitti, 2006a) to handle feature engineering over the structural representations of the `STRUCT` model. A polynomial kernel of degree 3 is applied to feature vectors (`FVEC`).

Combining structural and vector models. A typical kernel machine, e.g., SVM, classifies a test input $\boldsymbol{x}$ using the following prediction function: $h(\boldsymbol{x}) = \sum_i \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i)$, where α_i are the model parameters estimated from the training data, y_i are target variables, $\boldsymbol{x}_i$ are support vectors, and $K(\cdot,\cdot)$ is a kernel function. The latter computes the *similarity* between two comments. The `STRUCT` model treats each comment as a tuple $\boldsymbol{x} = \langle \boldsymbol{T}, \boldsymbol{v} \rangle$ composed of a shallow syntactic tree $\boldsymbol{T}$ and a feature vector $\boldsymbol{v}$. Hence, for each pair of comments $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, we define the following comment similarity kernel:

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = K_{\text{TK}}(\boldsymbol{T}_1, \boldsymbol{T}_2) + K_{\text{v}}(\boldsymbol{v}_1, \boldsymbol{v}_2), \quad (1)$$

where K_{TK} computes SHTK (defined next), and K_{v} is a kernel over feature vectors, e.g., linear, polynomial, Gaussian, etc.

Shallow syntactic tree kernel. Following the convolution kernel framework, we define the new

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

SHTK function from Eq. 1 to compute the similarity between tree structures. It counts the number of common substructures between two trees T_1 and T_2 without explicitly considering the whole fragment space. The general equations for Convolution Tree Kernels is:

$$TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2), \quad (2)$$

where N_{T_1} and N_{T_2} are the sets of the T_1's and T_2's nodes, respectively and $\Delta(n_1, n_2)$ is equal to the number of common fragments rooted in the n_1 and n_2 nodes, according to several possible definition of the atomic fragments.

To improve the speed computation of TK, we consider pairs of nodes (n_1, n_2) belonging to the same tree level. Thus, given H, the height of the `STRUCT` trees, where each level h contains nodes of the same type, i.e., chunk, POS, and lexical nodes, we define SHTK as the following[5]:

$$SHTK(T_1, T_2) = \sum_{h=1}^{H} \sum_{n_1 \in N_{T_1}^h} \sum_{n_2 \in N_{T_2}^h} \Delta(n_1, n_2), \quad (3)$$

where $N_{T_1}^h$ and $N_{T_2}^h$ are sets of nodes at height h.

The above equation can be applied with any Δ function. To have a more general and expressive kernel, we use Δ previously defined for PTK. More formally: if n_1 and n_2 are leaves then $\Delta(n_1, n_2) = \mu\lambda(n_1, n_2)$; else $\Delta(n_1, n_2) =$

$$\mu\Big(\lambda^2 + \sum_{\vec{I}_1, \vec{I}_2, |\vec{I}_1| = |\vec{I}_2|} \lambda^{d(\vec{I}_1) + d(\vec{I}_2)} \prod_{j=1}^{|\vec{I}_1|} \Delta(c_{n_1}(\vec{I}_{1j}), c_{n_2}(\vec{I}_{2j}))\Big),$$

where $\lambda, \mu \in [0, 1]$ are decay factors; the large sum is adopted from a definition of the subsequence kernel (Shawe-Taylor and Cristianini, 2004) to generate children subsets with gaps, which are then used in a recursive call to Δ. Here, $c_{n_1}(i)$ is the i^{th} child of the node n_1; $\vec{I}_1$ and $\vec{I}_2$ are two sequences of indexes that enumerate subsets of children with gaps, i.e., $\vec{I} = (i_1, i_2, .., |I|)$, with $1 \leq i_1 < i_2 < .. < i_{|I|}$; and $d(\vec{I}_1) = \vec{I}_{1l(\vec{I}_1)} - \vec{I}_{11} + 1$ and $d(\vec{I}_2) = \vec{I}_{2l(\vec{I}_2)} - \vec{I}_{21} + 1$, which penalizes subsequences with larger gaps.

It should be noted that: firstly, the use of a subsequence kernel makes it possible to generate child subsets of the two nodes, i.e., it allows for gaps, which makes matching of syntactic patterns less rigid. Secondly, the resulting SHTK is essentially a special case of PTK (Moschitti, 2006a), adapted to the shallow structural representation `STRUCT` (see Sec. 3.2). When applied to `STRUCT` trees, SHTK exactly computes the same feature space as PTK, but in faster time (on average). Indeed, SHTK required to be only applied to node pairs from the same level (see Eq. 3), where the node labels can match – chunk, POS or lexicals. This reduces the time for selecting the matching-node pairs carried out in PTK (Moschitti, 2006a; Moschitti, 2006b). The fragment space is obviously the same, as the node labels of different levels in `STRUCT` are different and will not be matched by PTK either.

Finally, given its recursive definition in Eq. 3 and the use of subsequence (with gaps), SHTK can derive useful dependencies between its elements. For example, it will generate the following subtree fragments: `[positive-NP [positive-A N]]`, `[S [negative-VP [negative-V [destroy]] [PRODUCT-NP]]]]` and so on.

4 YouTube comments corpus

To build a corpus of YouTube comments, we focus on a particular set of videos (technical reviews and advertisings) featuring commercial products. In particular, we chose two product categories: automobiles (`AUTO`) and tablets (`TABLETS`). To collect the videos, we compiled a list of products and queried the YouTube gData API[6] to retrieve the videos. We then manually excluded irrelevant videos. For each video, we extracted all available comments (limited to maximum 1k comments per video) and manually annotated each comment with its type and polarity. We distinguish between the following types:

product: discuss the topic product in general or some features of the product;

video: discuss the video or some of its details;

spam: provide advertising and malicious links; and

off-topic: comments that have almost no content (“lmao”) or content that is not related to the video (“Thank you!”).

Regarding the polarity, we distinguish between {*positive, negative, neutral*} sentiments with respect to the product and the video. If the comment contains several statements of different polarities, it is annotated as both *positive* and *negative*: “Love the video but waiting for iPad 4”. In total we have

[5] To have a similarity score between 0 and 1, a normalization in the kernel space, i.e. $\frac{SHTK(T_1,T_2)}{\sqrt{SHTK(T_1,T_1) \times SHTK(T_2,T_2)}}$ is applied.

[6] https://developers.google.com/youtube/v3/

annotated 208 videos with around 35k comments (128 videos TABLETS and 80 for AUTO).

To evaluate the quality of the produced labels, we asked 5 annotators to label a sample set of one hundred comments and measured the agreement. The resulting annotator agreement α value (Krippendorf, 2004; Artstein and Poesio, 2008) scores are 60.6 (AUTO), 72.1 (TABLETS) for the sentiment task and 64.1 (AUTO), 79.3 (TABLETS) for the **type** classification task. For the rest of the comments, we assigned the entire annotation task to a single coder. Further details on the corpus can be found in Uryupina et al. (2014).

5 Experiments

This section reports: (i) experiments on individual subtasks of opinion and type classification; (ii) the full task of predicting type and sentiment; (iii) study on the adaptability of our system by learning on one domain and testing on the other; (iv) learning curves that provide an indication on the required amount and type of data and the scalability to other domains.

5.1 Task description

Sentiment classification. We treat each comment as expressing positive, negative or neutral sentiment. Hence, the task is a three-way classification.

Type classification. One of the challenging aspects of sentiment analysis of YouTube data is that the comments may express the sentiment not only towards the product shown in the video, but also the video itself, i.e., users may post positive comments to the video while being generally negative about the product and vice versa. Hence, it is of crucial importance to distinguish between these two types of comments. Additionally, many comments are irrelevant for both the product and the video (off-topic) or may even contain spam. Given that the main goal of sentiment analysis is to select sentiment-bearing comments and identify their polarity, distinguishing between off-topic and spam categories is not critical. Thus, we merge the spam and off-topic into a single uninformative category. Similar to the opinion classification task, comment type classification is a multi-class classification with three classes: video, product and uninform.

Full task. While the previously discussed sentiment and type identification tasks are useful to model and study in their own right, our end goal is: given a stream of comments, to jointly predict both the type and the sentiment of each comment. We cast this problem as a single multi-class classification task with seven classes: the Cartesian product between {product, video} type labels and {positive, neutral, negative} sentiment labels plus the uninformative category (*spam* and *off-topic*). Considering a real-life application, it is important not only to detect the polarity of the comment, but to also identify if it is expressed towards the product or the video.[7]

Task	class	AUTO		TABLETS	
		TRAIN	TEST	TRAIN	TEST
Sentiment	positive	2005 (36%)	807 (27%)	2393 (27%)	1872 (27%)
	neutral	2649 (48%)	1413 (47%)	4683 (53%)	3617 (52%)
	negative	878 (16%)	760 (26%)	1698 (19%)	1471 (21%)
	total	5532	2980	8774	6960
Type	product	2733 (33%)	1761 (34%)	7180 (59%)	5731 (61%)
	video	3008 (36%)	1369 (26%)	2088 (17%)	1674 (18%)
	off-topic	2638 (31%)	2045 (39%)	2334 (19%)	1606 (17%)
	spam	26 (>1%)	17 (>1%)	658 (5%)	361 (4%)
	total	8405	5192	12260	9372
Full	product-pos.	1096 (13%)	517 (10%)	1648 (14%)	1278 (14%)
	product-neu.	908 (11%)	729 (14%)	3681 (31%)	2844 (32%)
	product-neg.	554 (7%)	370 (7%)	1404 (12%)	1209 (14%)
	video-pos.	909 (11%)	290 (6%)	745 (6%)	594 (7%)
	video-neu.	1741 (21%)	683 (14%)	1002 (9%)	773 (9%)
	video-neg.	324 (4%)	390 (8%)	294 (2%)	262 (3%)
	off-topic	2638 (32%)	2045 (41%)	2334 (20%)	1606 (18%)
	spam	26 (>1%)	17 (>1%)	658 (6%)	361 (4%)
	total	8196	5041	11766	8927

Table 1: Summary of YouTube comments data used in the sentiment, type and full classification tasks. The comments come from two product categories: AUTO and TABLETS. Numbers in parenthesis show proportion w.r.t. to the total number of comments used in a task.

5.2 Data

We split all the videos 50% between training set (TRAIN) and test set (TEST), where each video contains all its comments. This ensures that all comments from the same video appear either in TRAIN or in TEST. Since the number of comments per video varies, the resulting sizes of each set are different (we use the larger split for TRAIN). Table 1 shows the data distribution across the task-specific classes – **sentiment** and **type** classification. For the **sentiment** task we exclude off-topic and spam comments as well as comments with ambiguous sentiment, i.e., an-

[7] We exclude comments annotated as both video and product. This enables the use of a simple flat multi-classifiers with seven categories for the full task, instead of a hierarchical multi-label classifiers (i.e., type classification first and then opinion polarity). The number of comments assigned to both product and video is relatively small (8% for TABLETS and 4% for AUTO).

notated as both positive and negative.

For the **sentiment** task about 50% of the comments have neutral polarity, while the negative class is much less frequent. Interestingly, the ratios between polarities expressed in comments from AUTO and TABLETS are very similar across both TRAIN and TEST. Conversely, for the **type** task, we observe that comments from AUTO are uniformly distributed among the three classes, while for the TABLETS the majority of comments are product related. It is likely due to the nature of the TABLETS videos, that are more geek-oriented, where users are more prone to share their opinions and enter involved discussions about a product. Additionally, videos from the AUTO category (both commercials and user reviews) are more visually captivating and, being generally oriented towards a larger audience, generate more video-related comments. Regarding the **full** setting, where the goal is to have a joint prediction of the comment sentiment and type, we observe that video-negative and video-positive are the most scarce classes, which makes them the most difficult to predict.

5.3 Results

We start off by presenting the results for the traditional in-domain setting, where both TRAIN and TEST come from the same domain, e.g., AUTO or TABLETS. Next, we show the learning curves to analyze the behavior of FVEC and STRUCT models according to the training size. Finally, we perform a set of cross-domain experiments that describe the enhanced adaptability of the patterns generated by the STRUCT model.

5.3.1 In-domain experiments

We compare FVEC and STRUCT models on three tasks described in Sec. 5.1: sentiment, type and full. Table 2 reports the per-class performance and the overall accuracy of the multi-class classifier. Firstly, we note that the performance on TABLETS is much higher than on AUTO across all tasks. This can be explained by the following: (i) TABLETS contains more training data and (ii) videos from AUTO and TABLETS categories draw different types of audiences – well-informed users and geeks expressing better-motivated opinions about a product for the former vs. more general audience for the latter. This results in the different quality of comments with the AUTO being more challenging to analyze. Secondly, we observe that the STRUCT model provides 1-3% of absolute improvement in accuracy over FVEC for every task. For individual categories the F1 scores are also improved by the STRUCT model (except for the *negative* classes for AUTO, where we see a small drop). We conjecture that sentiment prediction for AUTO category is largely driven by one-shot phrases and statements where it is hard to improve upon the bag-of-words and sentiment lexicon features. In contrast, comments from TABLETS category tend to be more elaborated and well-argumented, thus, benefiting from the expressiveness of the structural representations.

Considering per-class performance, correctly predicting negative sentiment is most difficult for both AUTO and TABLETS, which is probably caused by the smaller proportion of the negative comments in the training set. For the **type** task, video-related class is substantially more difficult than product-related for both categories. For the **full** task, the class video-negative accounts for the largest error. This is confirmed by the results from the previous sentiment and type tasks, where we saw that handling negative sentiment and detecting video-related comments are most difficult.

5.3.2 Learning curves

The learning curves depict the behavior of FVEC and STRUCT models as we increase the size of the training set. Intuitively, the STRUCT model relies on more general syntactic patterns and may overcome the sparseness problems incurred by the FVEC model when little training data is available.

Nevertheless, as we see in Figure 2, the learning curves for sentiment and type classification tasks across both product categories do not confirm this intuition. The STRUCT model consistently outperforms the FVEC across all training sizes, but the gap in the performance does not increase when we move to smaller training sets. As we will see next, this picture changes when we perform the cross-domain study.

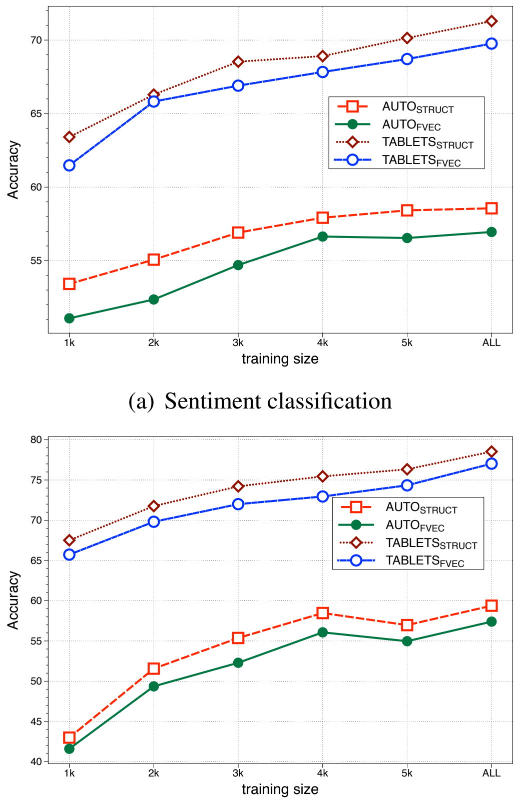
5.3.3 Cross-domain experiments

To understand the performance of our classifiers on other YouTube domains, we perform a set of cross-domain experiments by training on the data from one product category and testing on the other.

Table 3 reports the accuracy for three tasks when we use all comments (TRAIN + TEST) from AUTO to predict on the TEST from TABLETS

Task	class	AUTO						TABLETS					
		FVEC			STRUCT			FVEC			STRUCT		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1
Sent	positive	49.1	72.1	58.4	50.1	73.9	59.0	67.5	70.3	69.9	71.2	71.3	71.3
	neutral	68.2	55.0	61.4	70.1	57.6	63.1	81.3	71.4	76.9	81.1	73.1	77.8
	negative	42.0	36.9	39.6	41.3	35.8	38.8	48.3	60.0	54.8	50.2	62.6	56.5
	Acc			54.7			55.7			68.6			70.5
Type	product	66.8	73.3	69.4	68.8	75.5	71.7	78.2	95.3	86.4	80.1	95.5	87.6
	video	45.0	52.8	48.2	47.8	49.9	48.7	83.6	45.7	58.9	83.5	46.7	59.4
	uninform	59.3	48.2	53.1	60.6	53.0	56.4	70.2	52.5	60.7	72.9	58.6	65.0
	Acc			57.4			59.4			77.2			78.6
Full	product-pos	34.0	49.6	39.2	36.5	51.2	43.0	48.4	56.8	52.0	52.4	59.3	56.4
	product-neu	43.4	31.1	36.1	41.4	36.1	38.4	68.0	67.5	68.1	59.7	83.4	70.0
	product-neg	26.3	29.5	28.8	26.3	25.3	25.6	43.0	49.9	45.4	44.7	53.7	48.4
	video-pos	23.2	47.1	31.9	26.1	54.5	35.5	69.1	60.0	64.7	64.9	68.8	66.4
	video-neu	26.1	30.0	29.0	26.5	31.6	28.8	56.4	32.1	40.0	55.1	35.7	43.3
	video-neg	21.9	3.7	6.0	17.7	2.3	4.8	39.0	17.5	23.9	39.5	6.1	11.5
	uninform	56.5	52.4	54.9	60.0	53.3	56.3	60.0	65.5	62.2	63.3	68.4	66.9
	Acc			40.0			41.5			57.6			60.3

Table 2: In-domain experiments on AUTO and TABLETS using two models: FVEC and STRUCT. The results are reported for sentiment, type and full classification tasks. The metrics used are precision (P), recall (R) and F1 for each individual class and the general accuracy of the multi-class classifier (Acc).



(a) Sentiment classification

(b) Type classification

Figure 2: In-domain learning curves. ALL refers to the entire TRAIN set for a given product category, i.e., AUTO and TABLETS (see Table 1)

Source	Target	Task	FVEC	STRUCT
AUTO	TABLETS	Sent	66.1	66.6
		Type	59.9	64.1†
		Full	35.6	38.3†
TABLETS	AUTO	Sent	60.4	61.9†
		Type	54.2	55.6†
		Full	43.4	44.7†

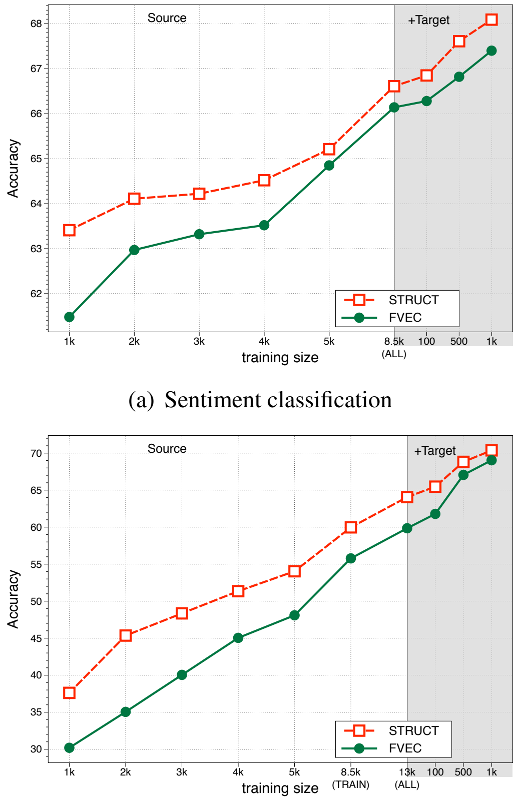
Table 3: Cross-domain experiment. Accuracy using FVEC and STRUCT models when trained/tested in both directions, i.e. AUTO→TABLETS and TABLETS→AUTO. † denotes results statistically significant at 95% level (via pairwise t-test).

and in the opposite direction (TABLETS→AUTO). When using AUTO as a source domain, STRUCT model provides additional 1-3% of absolute improvement, except for the sentiment task.

Similar to the in-domain experiments, we studied the effect of the source domain size on the target test performance. This is useful to assess the adaptability of features exploited by the FVEC and STRUCT models with the change in the number of labeled examples available for training. Additionally, we considered a setting including a small amount of training data from the target data (i.e., supervised domain adaptation).

For this purpose, we drew the learning curves of the FVEC and STRUCT models applied to the **sentiment** and **type** tasks (Figure 3): AUTO is used as the source domain to train models, which are tested on TABLETS.[8] The plot shows that when

[8] The results for the other direction (TABLETS→AUTO) show similar behavior.



(a) Sentiment classification

(b) Type classification

Figure 3: Learning curves for the cross-domain setting (AUTO→TABLETS). Shaded area refers to adding a small portion of comments from the same domain as the target test data to the training.

little training data is available, the features generated by the STRUCT model exhibit better adaptability (up to 10% of improvement over FVEC). The bag-of-words model seems to be affected by the data sparsity problem which becomes a crucial issue when only a small training set is available. This difference becomes smaller as we add data from the same domain. This is an important advantage of our structural approach, since we cannot realistically expect to obtain manual annotations for 10k+ comments for each (of many thousands) product domains present on YouTube.

5.4 Discussion

Our STRUCT model is more accurate since it is able to induce structural patterns of sentiment. Consider the following comment: *optimus pad is better. this xoom is just to bulky but optimus pad offers better functionality*. The FVEC bag-of-words model misclassifies it to be positive, since it contains two positive expressions (*better*, *better functionality*) that outweigh a single negative expression (*bulky*). The structural model, in contrast, is able to identify the product of interest (*xoom*) and associate it with the negative expression through a structural feature and thus correctly classify the comment as negative.

Some issues remain problematic even for the structural model. The largest group of errors are implicit sentiments. Thus, some comments do not contain any explicit positive or negative opinions, but provide detailed and well-argumented criticism, for example, *this phone is heavy*. Such comments might also include irony. To account for these cases, a deep understanding of the product domain is necessary.

6 Conclusions and Future Work

We carried out a systematic study on OM from YouTube comments by training a set of supervised multi-class classifiers distinguishing between video and product related opinions. We use standard feature vectors augmented by shallow syntactic trees enriched with additional conceptual information.

This paper makes several contributions: (i) it shows that effective OM can be carried out with supervised models trained on high quality annotations; (ii) it introduces a novel annotated corpus of YouTube comments, which we make available for the research community; (iii) it defines novel structural models and kernels, which can improve on feature vectors, e.g., up to 30% of relative improvement in type classification, when little data is available, and demonstrates that the structural model scales well to other domains.

In the future, we plan to work on a joint model to classify all the comments of a given video, s.t. it is possible to exploit latent dependencies between entities and the sentiments of the comment thread. Additionally, we plan to experiment with hierarchical multi-label classifiers for the full task (in place of a flat multi-class learner).

Acknowledgments

The authors are supported by a Google Faculty Award 2011, the Google Europe Fellowship Award 2013 and the European Community's Seventh Framework Programme (FP7/2007-2013) under the grant #288024: LIMOSINE.

References

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596, December.

Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. 2013. How noisy social media text, how diffrnt social media sources? In *IJCNLP*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*.

Hal Daumé, III. 2007. Frustratingly easy domain adaptation. *ACL*.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for Twitter: annotation, features, and experiments. In *ACL*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *KDD*.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *KDD*.

Jason S. Kessler, Miriam Eckert, Lyndsie Clark, and Nicolas Nicolov. 2010. The 2010 ICWSM JDPA sentiment corpus for the automotive domain. In *ICWSM-DWC*.

Klaus Krippendorf, 2004. *Content Analysis: An Introduction to Its Methodology, second edition*, chapter 11. Sage, Thousand Oaks, CA.

Alessandro Moschitti. 2006a. Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML*.

Alessandro Moschitti. 2006b. Making tree kernels practical for natural language learning. In *EACL*, pages 113–120.

Alessandro Moschitti. 2008. Kernel methods, syntax and semantics for relational text categorization. In *CIKM*.

Olutobi Owoputi, Brendan OConnor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL-HLT*.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*.

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

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *ACL*.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: an experimental study. In *ACL*.

Aliaksei Severyn and Alessandro Moschitti. 2012. Structural relationships for large-scale learning of answer re-ranking. In *SIGIR*.

Aliaksei Severyn and Alessandro Moschitti. 2013. Automatic feature engineering for answer selection and extraction. In *EMNLP*.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013. Learning semantic textual similarity with structural representations. In *ACL*.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Stefan Siersdorfer, Sergiu Chelaru, Wolfgang Nejdl, and Jose San Pedro. 2010. How useful are your comments?: Analyzing and predicting YouTube comments and comment ratings. In *WWW*.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*.

Anders Søgaard and Anders Johannsen. 2012. Robust learning in random subspaces: Equipping nlp for oov effects. In *COLING*.

Oscar Täckström and Ryan McDonald. 2011. Semi-supervised latent variable models for sentence-level sentiment analysis. In *ACL*.

Olga Uryupina, Barbara Plank, Aliaksei Severyn, Agata Rotondi, and Alessandro Moschitti. 2014. SenTube: A corpus for sentiment analysis on YouTube social media. In *LREC*.

Sida Wang and Christopher Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL*.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *EMNLP*.