Multi-lingual 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]CST - 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

Multi-lingual 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]CST - 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

In order to successfully apply opinion mining (OM) to the large amounts of user-generated content produced every day, we need robust models that can handle the noisy input well yet can easily be adapted to a new domain or language. We here focus on Opinion Mining for YouTube by (i) modeling classifiers that predict the type of a comment and its polarity, while distinguishing whether the polarity is directed towards the product or video; (ii) proposing a robust shallow syntactic structure (STRUCT) that adapts well when tested across domains; and (iii) evaluating the effectiveness on the proposed structure on two languages, English and Italian. We rely on tree kernels to automatically extract and learn features with better generalization power than traditionally used bag-of-word models. Our extensive empirical evaluation shows that (i) STRUCT outperforms the bag-of-words model both within the same domain (up to 2.6% and 3% of absolute improvement for Italian and English, respectively); (ii) it is particularly useful when tested across domains (up to more than 4% absolute improvement for both languages), especially when little training data is available (up to 10% absolute improvement) and (iii) the proposed structure is also effective in a lower-resource language scenario, where only less accurate linguistic processing tools are available.

Key words: Natural Language Processing, Opinion Mining, Social Media

1. Introduction

The increasing prevalence of social media like Twitter, Facebook and YouTube, which enable millions of users to share information and opinions quickly, has urged the need for new tools that robustly and automatically process the sheer amounts of user-generated content produced every day. Of particular importance is the fact that such information units, e.g., *tweets* in the case of Twitter or *comments* in case of YouTube, often carry *opinions* (or *sentiment*), i.e., they express subjective opinions of a particular user. In particular, we estimated that roughly 60-80% of the YouTube comments do actually contain opinions. Therefore, social media provide a key source that raises the importance of automatic extraction of opinions affecting the reputation of a person, and organization, or a specific product.

In this study we focus on YouTube. It is a platform that hosts videos uploaded by users (companies, private persons, etc.). YouTube is a unique environment with many facets: it is multi-modal, multi-lingual, multi-domain and multi-cultural, since people from different regions of the world can upload videos including textual information about different topics, they can rate videos, and comments on videos in different languages. Therefore, work promoting the success of sentiment analysis systems in such an environment is of high interest for both the industry and the research community.

Most prior research on opinion mining has been carried out on well-edited texts [1], and there has been some recent effort for sentiment analysis on Twitter [2]. In contrast, YouTube comprises several new challenges, which need to be tackled: (i) words expressing polarity can refer to either the video content itself ("the girl is cute") or the advertised product ("I hate the G2"), or may even contain contrasting sentiment; (ii) many comments are unrelated and are spam ("go to http to win an iPad"); (iii) YouTube is a large resource covering a variety of domains, thus it is not clear how well a supervised system trained on, say, the *tablets* domain, fares on a different domain, e.g., videos about *automobiles*; and (iv) it covers a large variety of languages, both in terms of the video and the comments for a certain video, thus approaches that handle the multilinguality aspect are of particular interest.

Still, the majority of systems for sentiment analysis rely on the simple bag-of-words (BOW) representation. That is, the input text is split into n-grams of words (or characters). These are used in machine learning algorithms, e.g., Support Vector Machines (SVM) or logistic regression, to induce a model that can classify new instances. In fact, the winning system [3] of the SemEval 2013 shared task [2] used a BOW representation together with a sentiment lexicon in SVMs. However, opinions usually involve complex interactions between lexical items (e.g., variations in sentiment scope and target, modality and negations, etc.). The standard BOW representation cannot take those into consideration, since by definition it abstracts away from many important clues. For example, consider a comment from a YouTube video, where a person reviews a specific product, namely the *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 the product name (***xoom***) and a list of 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 Italian comment on an **iPad** video expresses positive sentiment about another product (*galaxy note*), but is neutral with respect to the topical product.

> *Questo video fa un presentazione interessante dell' iPad, ma ho preso il galaxy note :) ha uno schermo fantastico. veramente bello e fluido. te lo consiglio.* (This video gives a nice introduction of iPad but I took the galaxy note :) it has a fantastic screen. very nice and fluent. I recommend it).

The following short English comment illustrates the main problem even better:

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

It contains two positive and one negative word, yet the sentiment towards the product is negative (the negative token *destroy* refers to *Xoom*). Thus, it is important to distinguish if the sentiment on YouTube is directed either towards the source video itself, or the product described in that video or another product. This cannot be captured by a bag-of-words model, which lacks the needed structural information for linking the sentiment with the target product.

In this paper, we present the results of the first research effort on the systematic analysis of opinion mining (OM) for YouTube comments capitalizing on our previous work [4, 5]. The contributions of our research are:

1. *User comment type and polarity classification*: to solve the issues outlined above, we devise a classification scheme that separates unrelated and spam comments from informative ones, which are, in turn, further categorized into product- or video-related (type classification). Moreover, we learn classifiers to assign polarity (positive, negative, neutral) to each type of informative comment. This allows us to filter out irrelevant comments, providing accurate OM distinguishing comments about the video and the target product.
2. *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., *Fiat Panda*, *xoom*, *Toyota Aygo*. In particular, we define an efficient tree kernel derived from the Partial Tree Kernel [6], suitable for encoding structural representation of noisy user-generated comments into Support Vector Machines (SVMs).

3. *Creation and annotation of a corpus of YouTube comments*: it contains 50k manually labeled (by an expert coder) comments for two product 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).
4. *Multi-lingual experiments*: in contrast to our and other prior work focused exclusively on one language (mainly, English), we show that our structural representation also works well for a less-resourced language, namely, Italian. This is of particular interest since it tests the proposed representation under limiting conditions: the performance of linguistic preprocessors is inferior to tools that were developed for English, or we might not even have access to all required analyzers. This allows us to gauge how well our structural models fare in a setup that can lead to inaccurate and noisy representations.

Our evaluation shows that our models are adaptable and thus robust across domains. Structural models generally improve on both tasks – polarity and type classification – yielding up to 30% 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. Moreover, our evaluation on the Italian data confirms that the proposed representation works well also on another rather different and less-resourced language.

The remainder of the paper is structured as follows. Section 2 discusses related work, Section 3 introduces our baseline models and structured representation. Section 4 introduces our corpus, Section 5 describes our experiments. Section 6 concludes and provides directions for future research.

2. Related work

In the past decade, automatic sentiment analysis of texts has attracted attention from both industry and academia. Such interest has produced a large body of research work, mainly focusing on the use of machine learning algorithms for opinion classification [1, 7].

Most prior work on opinion mining has been performed on more standardized forms of text, such as consumer reviews or newswire. The most commonly used

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

datasets include: the MPQA corpus of news documents [8], web customer review data [9], Amazon review data [10], the JDPA corpus of blogs [11], etc. However, these corpora are only partially suitable for developing models on social media, since the informal text poses additional challenges for Information Extraction and Natural Language Processing.

Opinion mining on Social Media has started to receive a lot of attention from the scientific community only very recently [12, 2]. Several annotation projects have been proposed to support the development of sentiment analysis models for social media, focusing mainly on Twitter—one of the biggest initiatives being the SemEval 2013 task on the sentiment analysis [2]. While most of these datasets contain only English documents, several corpora cover other languages. In particular, the SentiTUT project focuses on annotating a Twitter-based corpus for sentiment analysis in Italian [13].

Similar to Twitter, most YouTube comments are very short, the language is informal with numerous accidental and deliberate errors and grammatical inconsistencies [14], which make previous corpora less suitable to train models for OM on YouTube. Nevertheless, YouTube is a much less explored social media, with almost no work on sentiment analysis published so far. Siersdorfer et al. [15] focus on exploiting user ratings (the 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. Their goal is thus 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.

Early studies on opinion mining focused on the *document polarity* classification problem: for a given document, the algorithm assigns a label determining its general attitude (positive, negative or neutral). This formulation, however, is often too simplistic and thus the most recent studies address more fine-grained tasks, including identifying subjective vs. objective parts of a document [16, 17], opinion holders [18] or more complex sentiments and emotions [12], in particular, as irony or sarcasm [19, 20]. In our work, we refine the traditional polarity classification formulation, distinguishing between different sentiment targets (video vs. product). This allows us to provide a better understanding of user-generated comments that may address several topics, expressing different emotions.

Most of the previous work on supervised sentiment analysis use feature vectors to encode documents. Several studies provide in-depth analysis of lexical features for opinion mining [21, 22]. Such features can be effectively combined with external information, for example, with personalized co-occurence statistics [23]. Our feature-based baseline model (cf. Section 3) is very similar to the best performing system from the SemEval 2013 shared task on Twitter [3].

While a few successful attempts have been made to use more involved linguistic analysis for opinion mining, such as dependency trees [24, 25] and constituency trees with vectorized nodes [26], recently, a comprehensive study by Wang and Manning [27] showed that a simple model using bigrams and SVMs performs on par with more complex algorithms.

In contrast, we show that adding structural features (cf. Section 3) from syntactic trees is particularly useful for the cross-domain setting (cf. Section 5.4) and our findings carry over to our experiments on Italian (cf. Section 5.5). The structural features 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 [10, 28], the domain adaptation problem boils down to finding a more robust system [29, 30]. This is in line with recent advances in parsing The Web [31], where participants were asked to build a single system able to cope with different yet related domains.

Our approach, which we will describe in detail in the next section, relies on robust syntactic structures to automatically generate patterns that, given our empirical findings, have shown to adapt better. These representations were inspired by the semantic models developed for Question Answering [32, 33, 34] and Semantic Textual Similarity [35]. 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.

Most studies on opinion mining, especially for Social Media data, focus on English. Thus, several algorithms have been proposed for detecting opinions in English tweets within the recent SemEval evaluation campaign [2]. To our knowledge, only one study has addressed the task for Italian so far [36]. There are several important differences between this work and our research. We have created a manually annotated corpus (cf. Section 4) that can be used by the scientific community for experiments on supervised opinion mining. The previous work was based on automatically extracted data and a small manually tagged test collection. We propose structural models for our data representation and show that they yield superior performance. The previous work adopted a more baseline methodology, relying on term-based opinion scores.

3. Representations for Opinion Mining

In this section, we describe our learning systems exploiting innovative computational representations. First, we will introduce and discuss standardly used representation based on simple bag-of-words features. Then, we will introduce our novel structured representation, its advantages and requirements and the suitable learning machinery based on tree kernels.

3.1. Feature Set

As mentioned in Section 2, traditional opinion mining systems mainly use the bag-of-word representation in conjunction with sentiment lexicons and simple negation handling [1, 2, 3]. Following prior work, we use the features below in our baseline model (`FVEC`):

- **word n-grams**: unigrams and bigrams over lower-cased word lemmas, i.e., a binary feature that indicates the presence/absence of a given item.
- **lexicon**: a sentiment lexicon is a list of words associated with positive or negative sentiment. The lexicon features represent the count of the number of *positive* and *negative* tokens in a user comment, respectively.
- **negation**: this feature counts the number of negation words, e.g., {*don't, never, not, etc.*} in a given comment.[2] As we will discuss in the next Section, our structural representation enables a more powerful treatment of negation.
- **video concept**: the cosine similarity using word features between a user comment and the title/description of the video. Since many of the videos come with a title and a short description, we use this feature as a proxy to encode the topicality of each comment in relation to the video.

Our experiment section will show that models using the above feature representation are already very powerful. Yet, the goal of this study is to gauge the effectiveness of a more informed model that takes structure into consideration, as introduced next.

3.2. Structural model

In order to go beyond traditional feature vectors we use structural models (`STRUCT`), which encode each comment into a shallow syntactic tree. Each user comment is transformed into a tree structure, which constitutes the input of a tree kernel function. This, in turn, can generate structural features from such tree. Social media text, which is less well-edited and in general very noisy data, requires

[2]The list of English negation words is adopted from `http://sentiment.christopherpotts.net/lingstruc.html`. The list of Italian negation words has been compiled by the authors and consists of: "no", "non", "mai", "nessuno", "nessuna", "nessun", "niente", "nulla", "neppure", "neppur", "neanche", "mica", "né", "nemmeno", "manco", "giammai".

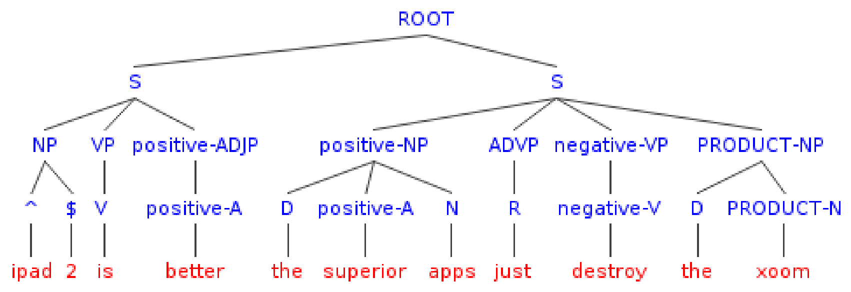


Figure 1: Shallow tree representation of the comment: *"iPad 2 is better. the superior apps just destroy the xoom."* taken from the video "Motorola Xoom Review". We (i) replace words with lemmas to increase generalization and (ii) introduce additional tags in the tree nodes to encode the central concept of the video (motorola *xoom*) and sentiment-bearing words *(better, superior, destroy)* directly in the tree nodes. For the former concept type, we add a `PRODUCT` tag on the chunk and part-of-speech nodes of the word *xoom* whereas, for the latter, we add the polarity tags, *positive* and *negative*. Multiple-sentence comments are split into separate trees `S`, linked by a root node.

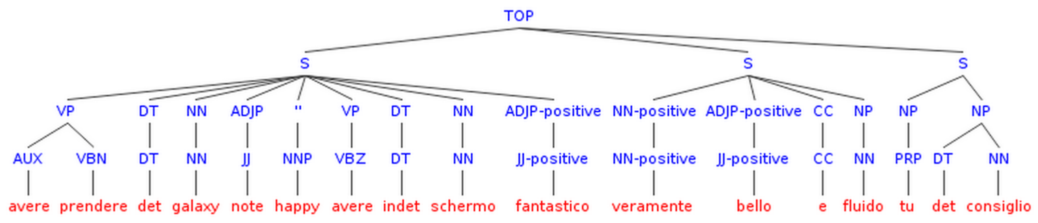


Figure 2: Shallow tree representation of an Italian example comment (labeled with `product` type and `neutral` sentiment). The comment belongs to a video about the iPad, therefore, no `PRODUCT` tag is given, thereby indicating that the many positive expressions are not related to the product itself.

special attention in handling. Therefore, we propose a structure that is specifically adapted to the noisy user-generated text:

1. The structural representation should encode structural dependencies, yet be robust to the noisy text input.
2. It is important that the structure encodes important concepts of the comments.

Therefore, we here propose to use a shallow syntactic tree representation with enriched tags. Our structure (cf. figures 1 and 2, for English and Italian, respectively) encodes not only words from the sentiment lexicons, but also important concepts about the product and negation words, which specifically target the sentiment and comment type classification tasks.

In particular, our proposed shallow syntactic tree is a two-level tree 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.

We address the second point above by enriching the syntactic trees with semantic tags, which encode: (i) central concepts of the video (e.g. the product name itself), (ii) sentiment-bearing words expressing *positive* or *negative* sentiment and (iii) negation words. To automatically identify concept words relevant for the video we use nouns detected by the part-of-speech tagger in the video title or 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 (cf. the parent nodes for *xoom* in Figure 1). In the same way, nodes associated with words found in the sentiment lexicon are enriched with their respective polarity (either *positive* or *negative*).

The advantage of the structured representation over the features vector-based (`FVEC`) model is the fact that it encodes powerful contextual syntactic features in terms of tree fragments. The latter are automatically generated and learned by SVMs with expressive tree kernels. On the downside, it needs linguistic processors such as a part-of-speech tagger and chunker – however, as we will see in the experimental setup, our proposed shallow representation is robust and handles noisy data well, works also on another language and in both cases outperforms the `FVEC` model. In fact, the `FVEC` model relies only on feature counts of simple features that do not take structure into consideration.

To illustrate the advantage of the structured representation, consider the English comment shown in Figure 1. It contains two *positive* and one *negative* token as found in the sentiment lexicon. This would strongly bias the `FVEC` sentiment classifier to assign a `positive` label to the comment. However, the polarity of this comment is negative with respect to the target product. In contrast, the `STRUCT` model relies on the fact that the negative word, *destroy*, refers to the product *xoom*, where the latter is in object relation to the verb *destroy*. Consider the excerpt of tree fragments generated by the tree kernel shown in Figure 3 (the kernel function generates many more subtrees, as described in more detail in the next section). The tree fragment on the left is a strong feature that helps the classifier to discriminate in such difficult cases. In general, tree kernels generate all possible subtrees, thereby producing generalized (back-off) features. For instance, the middle and right subtree shown in Figure 3 generalize the one on the left. The following section will depict in more detail how the tree kernel machinery works.

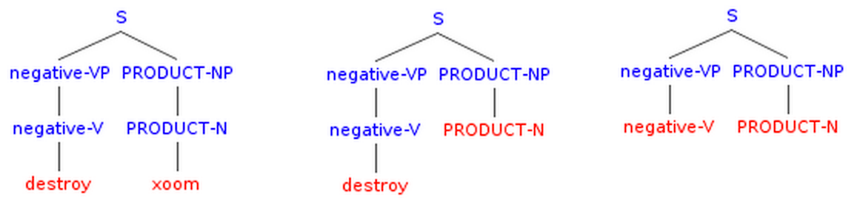


Figure 3: Excerpt of tree fragment features automatically generated by the kernel function for the example shown in Figure 1.

3.3. Machine Learning approach

We rely on supervised machine learning, i.e., SVMs, for performing opinion mining on YouTube. The goal is to learn a model for automatically detecting the sentiment and type of each comment. The tasks will be described in detail in Section 5.1, while here we focus on introducing the learning algorithm.

We build a multi-classifier using the one-vs-all scheme for detecting the sentiment or comment type: a binary classifier is trained for each class, and, at testing time, the class obtaining the maximum prediction score is chosen. Our back-end binary classifier is SVM- light-TK[3], which encodes structural kernels in the SVM-light [37] solver. We define a novel and efficient tree kernel function, namely, the **Sh**allow syntactic **T**ree **K**ernel (SHTK), which is as much expressive as the Partial Tree Kernel (PTK) [6] to handle feature engineering over the structural representations of the `STRUCT` model.

3.3.1. Kernel learning and vector model

A typical kernel machine, e.g., SVMs, 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.

In our `FVEC` model, the similarity between two comments is measured with a polynomial kernel of degree 3 between the two comment feature vectors $\boldsymbol{v}_i$:

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = K_{\text{v}}(\boldsymbol{v}_1, \boldsymbol{v}_2), \tag{1}$$

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

3.3.2. *Combining structural and vector models*

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:

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = K_{\text{TK}}(\boldsymbol{T}_1, \boldsymbol{T}_2) + K_{\text{v}}(\boldsymbol{v}_1, \boldsymbol{v}_2), \tag{2}$$

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

3.3.3. *Shallow syntactic tree kernel (SHTK)*

Following the convolution kernel framework, we define the new SHTK function from Eq. 2 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 equation for Convolution Tree Kernels is:

$$TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2), \tag{3}$$

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. Such number can vary according to several possible definitions 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[4]:

$$SHTK(T_1, T_2) = \sum_{h=1}^{H} \sum_{n_1 \in N_{T_1}^h} \sum_{n_2 \in N_{T_2}^h} \Delta(n_1, n_2), \tag{4}$$

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

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

[4]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.

where $\lambda, \mu \in [0, 1]$ are decay factors; the large sum is adopted from a definition of the subsequence kernel [38] 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 penalize the 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 [6], 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 requires to be only applied to node pairs from the same level (see Eq. 4), where the node labels can match – chunk, POS or lexicals. This reduces the time for selecting the matching-node pairs carried out in PTK [6]. 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. 4 and the use of subsequence (with gaps), SHTK can derive useful dependencies between its elements.

As discussed above, the SHTK function defines a very general type of the tree fragments imposing very little constraints on their shape, i.e., a fragment can start at any non-terminal node, and nodes in the fragment can contain children with gaps. According to the recursive definition of the Δ function for the SHTK, it will generate various tree fragments as shown in Fig. 3. Note that each of the tree fragment defined by SHTK encodes both syntactic structure and information about the sentiment (via additional node tags).

To sum up, to take advantage of automatic feature engineering offered by kernels, we define SHTK function to mine all possible tree fragments from a `STRUCT` model which encodes comments into a shallow syntactic tree. The mined tree fragments (implicitly generated the tree kernel) represent features used by an SVM to learn a predictive model. Hence, the tree kernel learning framework applied to our `STRUCT` representation results in a more expressive feature representation than, for example, typical bag-of-words as it integrates both syntactic and sentiment information in a structured way.

4. SenTube: the YouTube comments corpus

In order to construct the YouTube comments corpus, we compiled a list of products (Apple iPad, Motorola xoom, Fiat 500, etc.). Our target videos are either commercials or review videos of the products, therefore we queried the YouTube API (gdata)[5] to retrieve the initial set of links to YouTube videos.[6] Since we aimed at collecting a multi-lingual corpus, i.e., English and Italian videos, we use the same list of products for the two languages, which allows future cross-lingual experiments. The initial set of links was then manually inspected in order to ensure that the videos were indeed of the target language and that they contained at least several comments. We focused on two particular product domains: automobiles (`AUTO`) and tablets (`TABLETS`).

For each video, we downloaded all available comments (limited to a maximum of 1k comments per video). They were then manually annotated with comment type and polarity. We distinguish between the following types:

- **product**: about products in general or some of their aspects;
  - positive/negative/neutral (toward the product)
- **video**: about videos or some of their details;
  - positive/negative/neutral (toward the video)
- **spam**: advertisements and/or malicious links; and
- **off-topic**: comments that have almost no content ("lmao") or content that is not related to the video ("Thank you!").

With respect to comment polarity, we follow prior work and use the standard categories, i.e., expressing {*positive, negative, neutral*} sentiment, but distinguishing whether a comment refers to the product or the video. Moreover, if a comment contains several statements of different polarities, it is annotated as both *positive* and *negative*. For example, the following is positive towards the video but negative with regard to the target product: "Awesome clip but waiting for the Kindle paperwhite".

In total we have 217 annotated English videos with around 43k comments (139 videos on `TABLETS` and 78 videos for the `AUTO` domain). Moreover, we have 198 annotated Italian videos on the same products with around 10k comments (100 videos on `TABLETS` and 98 for `AUTO`). For several products, there was no corresponding Italian commercial or review video. Table 1 highlights different corpus

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

[6]By appending either "commercial" or "review" to the query term which is product name. For the Italian data the links were collected manually through the web interface.

	English			Italian		
	AUTO	TABLETS	ALL	AUTO	TABLETS	ALL
comments in total	21051	22202	43253	4752	5639	10391
comments per video	269.9	159.7	199.4	48.5	56.4	52.5
tokens in total	442123	463066	905189	119724	118663	238387
tokens per comment	21.0	20.9	20.9	25.2	21.0	22.9
vocabulary size	28020	22558	42180	17489	13629	26339
unique tokens	17111	13823	26295	11591	8524	17131
sentiment tokens	53193	51557	104750	4159	4184	8343
sentiment tokens per comment	2.5	2.3	2.4	0.9	0.7	0.8

Table 1: Corpus statistics for the SenTube dataset.

statistics for both languages and both domains. There are several notable differences between the two languages.

First, the Italian videos contain fewer comments in total (around 50 comments per video in Italian, as opposed to 200 comments per video for English). This reflects the fact that the English-speaking YouTube audience is much larger and therefore Italian videos do not receive the same number of comments. Thus, our Italian corpus is approximately 1/3 in size of the English corpus.

Second, a much smaller percentage of tokens in Italian can be found in our sentiment dictionary. This can be explained by the lower coverage of the Italian sentiment lexicon (see Section 5 for more details on the sentiment dictionaries we have used in our experiments). It should also be stressed that for both languages, the percentage of sentiment tokens is low (0.7-2.5 tokens per comment, around 3-11% of all the tokens). This highlights the importance of a more general approach, that is able to learn task-relevant sentiment patterns going beyond specific lexical.

Finally, for all the domains, we get a very large number of unique tokens, mainly due to a large amount of noise in user-generated comments. This limits the applicability of bag-of-words approaches.

For our experiments, we have split the entire SenTube corpus into several parts, as described in Section 5 below. Summary statistics over our train and test partitions (in number of comments and the distribution over the different labels) are given in Table 2 and Table 3 for English and Italian, respectively.

To gauge the quality of the labels, in an initial experiment we asked five annotators to label a sample set of one hundred comments and measured the agreement. The resulting annotator agreement α value [39, 40] scores are 60.6 (`AUTO`), 72.1 (`TABLETS`) for the *sentiment polarity* task and 64.1 (`AUTO`), 79.3 (`TABLETS`) for the *type* classification task. For the remaining comments, the entire annotation task was assigned to a single coder. Further details on the corpus, including detailed annotation guidelines, can be found in [4].

5. Experiments

In this section, we first introduce our tasks, data and experimental setup. Then, we report on: (i) experiments on the individual subtasks of opinion and type classification; (ii) the full task of predicting both type and sentiment. Moreover, we test the models (iii) to study their adaptability across domains, i.e., train on one domain and test on another. We here provide also learning curves that give an indication on the required amount and type of data and the scalability to other domains. Finally, we report on (iv) the applicability of the proposed structured model to another language, namely Italian, where we see that even in such a less-resourced setup our models outperform the commonly used feature-based methods.

5.1. Task description

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

Type classification.. Since the challenge of sentiment analysis on YouTube data lies in the fact that a comment may express the sentiment not only towards the `product` shown in the video, but also the `video` itself, it is of important to distinguish between the opinion target, whether it is the video or product. For instance, users may post positive comments to the video while being generally negative about the product and vice versa. Hence, type classification is important.

Additionally, many comments are irrelevant for both the product and the video (`off-topic`) or may even contain `spam`. However, given that the main goal of sentiment analysis is to select sentiment-bearing comments and identify their polarity, we here do not consider distinguishing between `off-topic` and `spam` and conflate them into a single category (`uninformative`). Therefore, the comment type classification task is again a three-way decision: `video`, `product` and `uninformative`.

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.

Therefore, we cast the problem to a single multi-class classification task[7] with seven classes: the Cartesian product between {`product, video`} type labels

[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`).

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

and {`positive, neutral, negative`} sentiment labels, and the additional class for `uninformative`.

5.2. Data

Tables 2 and 3 show the datasets for English and Italian, respectively. For both languages, we split the videos into 50% training (TRAIN) and 50% test set (TEST), such that each video contains all its comments. This ensures that all comments from the same video appear in either TRAIN or 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 2 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., annotated as both `positive` and `negative`.

For both languages, in the **sentiment** task the majority of comments have `neutral` polarity, while the `negative` class is the least frequent. Approximately 50% of the comments are neutral for the English dataset, and 42% in the

Task	class	AUTO		TABLETS	
		Train	Test	Train	Test
Sentiment	positive	573 (31%)	367 (36%)	572 (25%)	423 (22%)
	neutral	771 (42%)	433 (42%)	1240 (55%)	1065 (56%)
	negative	503 (27%)	219 (21%)	452 (20%)	417 (22%)
	total	1847	1019	2264	1905
Type	product	1183 (44%)	638 (48%)	1860 (62%)	1722 (69%)
	video	775 (29%)	448 (33%)	548 (18%)	319 (13%)
	off-topic	760 (28%)	257 (19%)	594 (20%)	435 (18%)
	spam	0 (0%)	0 (0%)	21 (<1%)	7 (<1%)
	total	2718	1343	3023	2483
Full	product-pos.	322 (12%)	187 (14%)	415 (14%)	353 (14%)
	product-neu.	425 (16%)	220 (16%)	932 (31%)	869 (35%)
	product-neg.	351 (13%)	173 (13%)	386 (13%)	369 (15%)
	video-pos.	251 (9%)	180 (13%)	157 (5%)	70 (3%)
	video-neu.	346 (13%)	213 (16%)	308 (10%)	196 (8%)
	video-neg.	152 (6%)	46 (3%)	66 (2%)	48 (2%)
	off-topic	760 (28%)	257 (19%)	594 (20%)	435 (18%)
	spam	0 (0%)	0 (0%)	21 (<1%)	7 (<1%)
	total	2607	1276	2879	2347

Table 3: Summary of YouTube comments data for Italian.

Italian one. Interestingly, the ratios between polarities expressed in comments from `AUTO` and `TABLETS` are very similar across both TRAIN and TEST. This actually holds for both languages.

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. This holds also for the Italian dataset, although it is slightly less balanced for `AUTO` (a bit more product related comments). However, also there we observe the majority of comments (69%) about the product. It is likely due to the nature of the `TABLETS` videos, which are more geek-oriented, where users are more prone to share their opinions and enter involved discussions about a product. In contrast, 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 least frequent classes, which makes them the most difficult to learn and predict.

5.3. Processing tools, lexicons and evaluation measures

For English, several models were developed recently [41, 42], specifically tailored to process noisy inputs. They yield significant reductions in the error rate on user-generated texts, e.g., Twitter. Hence, we use the CMU Twitter pos-tagger [42, 43] in our setup for English to obtain the part-of-speech tags. Our second component – the chunker – is the one developed by Ritter et al. [41], which also comes with a model trained on Twitter data[8]. It has been shown to perform better on noisy data such as user comments. Since the linguistic conventions used on Twitter and YouTube show similarities [14], we here exploit the existence of such tools to process YouTube data in contrast to tools trained on more well-edited text.

For Italian, however, no specific tools are available at the moment to process social media data. We therefore opted for general-purpose modules. This way, we can gauge how well our model still works in such a suboptimal setup. We obtain tokens, lemmas and part-of-speech tags through TextPro [44]—a state-of-the-art NLP suite for Italian. To generate our shallow structural representations, we use the Berkeley parser trained on the Torino Treebank [45]. Given that we reconstruct chunks from a full syntactic parse tree, we opted for a simple flattening heuristic to obtain the chunk nodes. In particular, we remove all the intermediate nodes from a parse tree that are at the height greater than two. An example tree (where chunk nodes are obtained by flattening a full syntactic parse) is shown in Figure 2.

Regarding sentiment lexicons: for English, we merge two manually constructed sentiment lexicons that are freely available, the MPQA Lexicon [8] and the lexicon of Hu and Liu [9]. For Italian, we use the SentiStrength lexicon [46].

For system evaluation, we measure accuracy (the proportion of correctly predicted labels) as well as the per-label F1, the balanced mean between precision (the proportion of correct predictions for that label) and recall (the proportion of correct predictions for that label compared to the gold standard).

5.4. English Results

We first present the results for the traditional in-domain setup, where both TRAIN and TEST come from the same domain, e.g., `AUTO` or `TABLETS`, respectively. Next, we show the learning curves to analyze the behavior of the `FVEC` and `STRUCT` models when the training size increases. Fnally, we perform a set of cross-domain experiments that describe the enhanced adaptability of the patterns generated by the `STRUCT` model.

[8]The chunker from [41] 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.

5.4.1. In-domain experiments

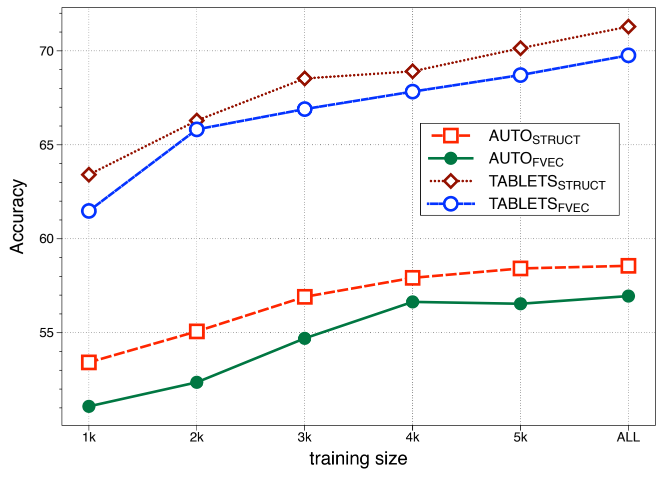
We compare the FVEC and STRUCT models on the three tasks described in Sec. 5.1: sentiment, type and full. Table 4 reports the per-class performance and the overall accuracy of the multi-class classifier. We note that: first of all, the

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

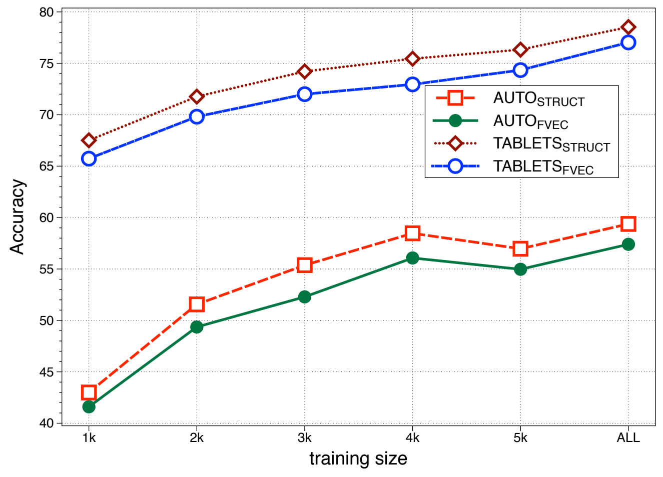
Table 4: In-domain experiments on AUTO and TABLETS using two models: FVEC and STRUCT. The results are reported for sentiment, type and full classification tasks. The metrics used are precision (P), recall (R) and F1 for each individual class and the general accuracy of the multi-class classifier (Acc). Experiments on English data.

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 the 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 and most importantly, we observe that the STRUCT model provides 1.5-3% absolute improvement in accuracy over the FVEC. This actually holds for all tasks. If we examine individual categories and the F1 scores, we see that F1 also improves with the STRUCT model, except for the *negative* classes for AUTO, where we see a small drop. We conjecture that the sentiment prediction for the 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 the TABLETS category tend to be more elaborated and well-argumented, thus, full benefiting from the extended expressiveness of the structural



(a) Sentiment classification



(b) Type classification

Figure 4: In-domain learning curves. ALL refers to the entire TRAIN set for a given product category, i.e., `AUTO` and `TABLETS` (see Table 2). Experiments on English data.

representation.

Finally, if we consider the per-class performance breakdown, we observe that correctly predicting `negative` sentiment is the most difficult task for both `AUTO` and `TABLETS`. This again is probably caused by the fact that there is a smaller proportion of the negative comments in the training set. For the **type** task, the video-related class is substantially more difficult than the product-related one for both domains. 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. Also there we saw that handling negative sentiment and detecting video-related comments turns out to be the most difficult.

5.4.2. Learning curves

In this section we examine what happens if we have different amounts of training data at our disposal. We examine the learning curves for both the FVEC and STRUCT models for increasing training set sizes. 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.

As shown in Figure 4, the STRUCT model consistently outperforms the FVEC across all training sizes. This also holds for the case very little training data is available, although both learning curves for sentiment and type classification tasks across both product categories show that there is not that large of an advantage as expected for very little data. The STRUCT model outperforms the FVEC model by an almost constant margin. As we will see next, this picture considerably changes when we apply the model across domains.

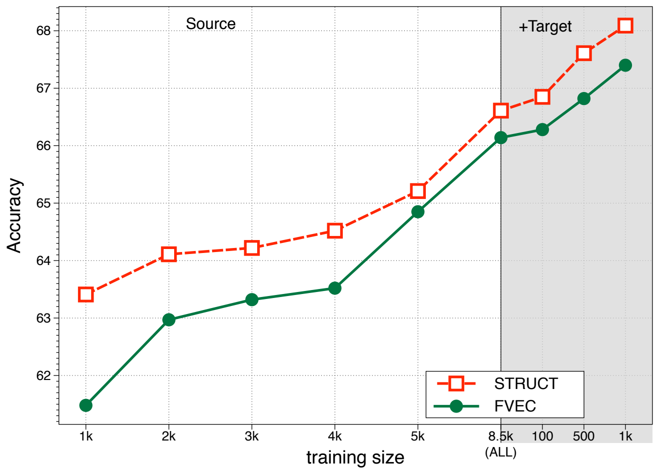
5.4.3. Cross-domain experiments

To examine the performance of our classifiers on other YouTube domains, we perform a set of cross-domain experiments. This means that we train a model on the data from one product category and test it on data from the other. This is important as it allows us to examine the adaptability of our models and estimate how much and whether we need training data for a new domain.

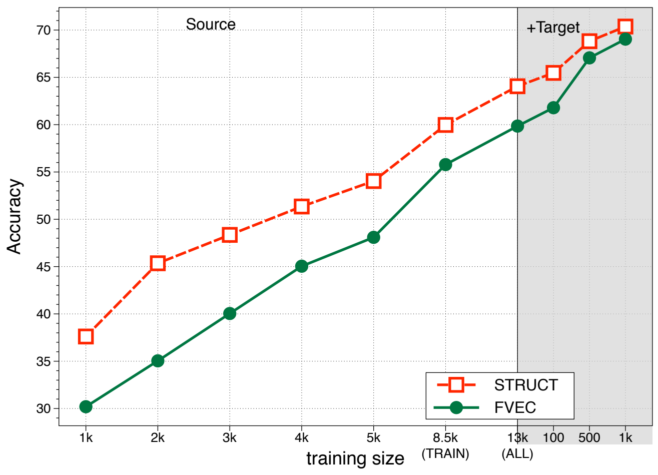
Table 5 reports the accuracy for the three tasks when we use all comments (TRAIN + TEST) from AUTO to predict on the TEST from TABLETS, and the other way around (TABLETS→AUTO). When we use AUTO as a source domain,

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 5: Cross-domain experiment for English. Accuracy using FVEC and STRUCT models when trained/tested in both directions, i.e. AUTO→TABLETS and TABLETS→AUTO. † denotes results statistically significant at $p < 0.05$ (via pairwise t-test).



(a) Sentiment classification



(b) Type classification

Figure 5: Experiments on English data: 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.

the STRUCT model provides an 1-3% absolute improvement in accuracy, except for the sentiment task where it reaches baseline performance. All improvements are significant, thus showing that our structure representation is way more robust across domains than the vector based model.

Similar to the in-domain experiments, we studied the effect of the source domain size on the target test performance. This is important 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).

The learning curves of the `FVEC` and `STRUCT` models applied to the **sentiment** and **type** tasks are shown in Figure 5: `AUTO` is used as the source domain to train models, which are tested on `TABLETS`.[9] The plot shows that when little training data is available, the features generated by the `STRUCT` model are much more robust and show better adaptability, up to 10% absolute improvement over FVEC (30% of relative improvement). 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 (see the shaded area in the Figure 5). This is an important advantage of our structural approach, since we cannot realistically expect to obtain manual annotations for 10k+ comments for each (of the many thousands) product domains present on YouTube.

5.5. Experiments on Opinion Mining for Italian

The Italian data poses several additional challenges for our approach: firstly, such data is much smaller, about 1/5 to 1/3 of the corresponding English one depending on the task (cf. Table 3). Note that we collected the same set of products for both languages. For some products, however, we were not able to find any corresponding Italian videos. Even for well-represented products, the average number of comments per video is much lower for Italian (50 comments per video) than for English (about 200 comments per video). Table 3 describes the Italian corpus.

Secondly, the Italian preprocessing is less robust. In particular, we do not have any social media-specific models for Italian. Our way to generate chunks is error-prone as well: we rely on a parser (trained on a relatively small corpus) and some flattening heuristics to obtain Italian chunk trees. English trees, on the contrary, are created through a toolkit trained on social media specifically for the chunking task on a much larger dataset.

Finally, to our knowledge, there are no high-coverage sentiment lexical resources for Italian available yet. We did run some experiments on inducing an Italian sentiment dictionary from the SentiWordNet [47] via the MultiWordNet synset mappings [48], following the approach of [36]. This induced dictionary, however, is too noisy (for example, the word “avere” (“to have”) is marked as `positive`)

[9] Results for the other direction (`TABLETS`$\rightarrow$`AUTO`) show similar behavior and are thus omitted.

and, at the same time, has a limited coverage. The SentiStrength lexicon showed a better performance in our pilot experiments and was therefore adopted for the current study. However, SentiStrength has a much lower coverage (our English lexicon is 4x larger).

The latter two issues are crucial for the `STRUCT` model: since our structural representations integrate different types of linguistic (part-of-speech, chunks) and task-specific (sentiment clues) evidence, they can get noisy when the preprocessing becomes less accurate. We therefore expect the difference between the models to be less pronounced for Italian than for English.

Table 6 reports the results for the in-domain and cross-domain experiments on Italian. Despite the suboptimal setup, our evaluation experiments actually show that the structural representation works also for Italian. In both experiments (in-domain and across-domain) we do see an improvement up to 4.2% absolute in accuracy (for type classification from AUTO to TABLETS). Thus, the same trend holds for both Italian and English, with the only exceptions of (i) the in-domain type classification of TABLETS, where both models for Italian reach the same performance; and (ii) AUTO to TABLETS sentiment classification, where `STRUCT` slightly decreases the system accuracy (not statistically significant result), and (iii) only minor improvements were obtained for the in-domain TABLETS setup.

Source	Target	Task	`FVEC`	`STRUCT`
		Sent	59.3	61.6
`AUTO`	`AUTO`	Type	68.1	70.7†
		Full	44.4	45.6†
		Sent	63.6	64.4
`TABLETS`	`TABLETS`	Type	77.3	77.3
		Full	51.6	52.4
		Sent	62.4	61.2
`AUTO`	`TABLETS`	Type	59.7	63.8†
		Full	27.7	29.7†
		Sent	53.3	54.3
`TABLETS`	`AUTO`	Type	55.3	56.4†
		Full	30.6	31.7†

Table 6: In-domain and cross-domain experiments for comments in Italian. The accuracy using `FVEC` and `STRUCT` models for in-domain source and target are the same. For the cross-domain, we use both directions, i.e. `AUTO`$\rightarrow$`TABLETS` and `TABLETS`$\rightarrow$`AUTO`. † denotes results statistically significant at $p < 0.05$ (via pairwise t-test).

5.6. Discussion

Our `STRUCT` model is more accurate that the `FVEC` model 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 it thus correctly classifies the comment as `negative`.

However, our model has its limitations as well. For instance, the largest group of errors are implicit sentiments. Thus, some comments do not contain any explicit positive or negative opinion, 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, for instance, also by linking product aspects to the actual product.

6. Conclusions and Future Work

This paper presents the results of a research effort targeting opinion mining on YouTube comments. We tackled the problem as multi-class supervised classification task, where the goal is to detect the comment type and polarity. A peculiarity of our approach is that we distinguish between video and product related opinions. We proposed a novel structural representation based on shallow syntactic trees enriched with additional conceptual information and show that it outperforms traditional approaches that are based on bag-of-words models.

To sum up, our research effort comprises the following contributions: (i) it shows that effective OM can be carried out with supervised models trained on high quality annotations; (ii) it introduces a novel manually annotated multilingual corpus of YouTube comments, which we make available for the research community; (iii) it defines new 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; (iv) it addresses the opinion mining task in a multilingual setting, showing that our structural models are robust enough to be used for languages other than English.

For future work, we plan to work on a joint model to classify all the comments of a given video, such that 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) and perform cross-lingual experiments, in order to exploit the data from one language to predict opinions on the other.

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. We thank the HLT group at Fondazione Bruno Kessler for providing us with the syntactic parsing model for Italian.

References

[1] B. Pang, L. Lee, Opinion mining and sentiment analysis, Foundations and trends in information retrieval 2 (1-2) (2008) 1–135.

[2] P. Nakov, S. Rosenthal, Z. Kozareva, V. Stoyanov, A. Ritter, T. Wilson, SemEval-2013 Task 2: Sentiment Analysis in Twitter, in: SemEval, 2013.

[3] S. Mohammad, S. Kiritchenko, X. Zhu, NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets, in: SemEval, 321–327, 2013.

[4] O. Uryupina, B. Plank, A. Severyn, A. Rotondi, A. Moschitti, SenTube: A Corpus for Sentiment Analysis on YouTube Social Media, in: LREC, 2014.

[5] A. Severyn, A. Moschitti, O. Uryupina, B. Plank, K. Filippova, Opinion Mining on YouTube, in: ACL, 2014.

[6] A. Moschitti, Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees, in: ECML, 2006.

[7] A. Montoyo, P. Martínez-Barco, A. Balahur, Subjectivity and sentiment analysis: An overview of the current state of the area and envisaged developments, Decision Support Systems 53 (4) (2012) 675–679.

[8] T. Wilson, J. Wiebe, P. Hoffmann, Recognizing contextual polarity in phrase-level sentiment analysis, in: EMNLP, 2005.

[9] M. Hu, B. Liu, Mining and summarizing customer reviews, in: KDD, 2004.

[10] J. Blitzer, M. Dredze, F. Pereira, Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification, in: ACL, 2007.

[11] J. S. Kessler, M. Eckert, L. Clark, N. Nicolov, The 2010 ICWSM JDPA Sentiment Corpus for the Automotive Domain, in: ICWSM-DWC, 2010.

[12] E. Cambria, A. Hussain, Sentic Computing: Techniques, Tools, and Applications, Springer, 2012.

[13] C. Bosco, V. Patti, A. Bolioli, Developing Corpora for Sentiment Analysis: The Case of Irony and Senti-TUT, IEEE Intelligent Systems 28 (2) (2013) 55–63.

[14] T. Baldwin, P. Cook, M. Lui, A. MacKinlay, L. Wang, How noisy social media text, how diffrnt social media sources?, in: IJCNLP, 2013.

[15] S. Siersdorfer, S. Chelaru, W. Nejdl, J. San Pedro, How useful are your comments?: Analyzing and predicting YouTube comments and comment ratings, in: WWW, 2010.

[16] B. Pang, L. Lee, A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts, in: ACL, 2004.

[17] A. Yessenalina, Y. Yue, C. Cardie, Multi-level Structured Models for Document Sentiment Classification, in: EMNLP, 2010.

[18] R. Johansson, A. Moschitti, Relational Features in Fine-grained Opinion Analysis, Computational Linguistics 39 (3) (2013) 473–509.

[19] A. Reyes, P. Rosso, T. Veale, A Multidimensional Approach for Detecting Irony in Twitter, Language Resources and Evaluation 47 (1) (2013) 239–268.

[20] P. Carvalho, L. Sarmento, M. J. Silva, E. de Oliveira, Clues for detecting irony in user-generated contents: oh...!! it's so easy;-), in: 1st international CIKM workshop on Topic-sentiment analysis for mass opinion, 2009.

[21] B. Pang, L. Lee, S. Vaithyanathan, Thumbs Up? Sentiment Classification Using Machine Learning Techniques, in: EMNLP, 79–86, 2002.

[22] E. Riloff, S. Patwardhan, J. Wiebe, Feature Subsumption for Opinion Analysis, in: EMNLP, 2006.

[23] P. Panicheva, J. Cardiff, P. Rosso, Identifying subjective statements in news titles using a personal sense annotation framework, Journal of the American Society for Information Science and Technology 64 (7) (2013) 1411–1422.

[24] O. Täckström, R. McDonald, Semi-supervised latent variable models for sentence-level sentiment analysis, in: ACL, 2011.

[25] S. Poria, E. Cambria, G. Winterstein, G.-B. Huang, Sentic patterns: Dependency-based rules for concept-level sentiment analysis, Knowledge-Based Systems, Special Issue on Big Data for Social Analysis (2014) 1–32.

[26] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, C. D. Manning, Semi-supervised recursive autoencoders for predicting sentiment distributions, in: EMNLP, 2011.

[27] S. Wang, C. Manning, Baselines and Bigrams: Simple, Good Sentiment and Topic Classification, in: ACL, 2012.

[28] H. Daumé, III, Frustratingly easy domain adaptation, in: ACL, 2007.

[29] A. Søgaard, A. Johannsen, Robust Learning in Random Subspaces: Equipping NLP for OOV Effects, in: COLING, 2012.

[30] B. Plank, A. Moschitti, Embedding Semantic Similarity in Tree Kernels for Domain Adaptation of Relation Extraction, in: ACL, 2013.

[31] S. Petrov, R. McDonald, Overview of the 2012 shared task on parsing the Web, in: Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL), 2012.

[32] A. Moschitti, Kernel Methods, Syntax and Semantics for Relational Text Categorization, in: CIKM, 2008.

[33] A. Severyn, A. Moschitti, Structural relationships for large-scale learning of answer re-ranking, in: SIGIR, 2012.

[34] A. Severyn, A. Moschitti, Automatic Feature Engineering for Answer Selection and Extraction, in: EMNLP, 2013.

[35] A. Severyn, M. Nicosia, A. Moschitti, Learning Semantic Textual Similarity with Structural Representations, in: ACL, 2013.

[36] V. Basile, M. Nissim, Sentiment analysis on Italian tweets, in: 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, 2013.

[37] T. Joachims, Optimizing Search Engines Using Clickthrough Data, in: KDD, 2002.

[38] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, 2004.

[39] K. Krippendorf, Content Analysis: An Introduction to Its Methodology, second edition, chap. 11, Sage, Thousand Oaks, CA, 2004.

[40] R. Artstein, M. Poesio, Inter-coder agreement for computational linguistics, Computational Linguistics 34 (4) (2008) 555–596.

[41] A. Ritter, S. Clark, Mausam, O. Etzioni, Named entity recognition in tweets: an experimental study, in: ACL, 2011.

[42] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, N. A. Smith, Part-of-speech tagging for Twitter: annotation, features, and experiments, in: ACL, 2011.

[43] O. Owoputi, B. OConnor, C. Dyer, K. Gimpel, N. Schneider, N. A. Smith, Improved part-of-speech tagging for online conversational text with word clusters, in: NAACL-HLT, 2013.

[44] E. Pianta, C. Girardi, R. Zanoli, The TextPro tool suite, Tech. Rep., FBK-IRST, 2007.

[45] C. Bosco, V. Lombardo, Comparing linguistic information in treebank annotations, in: LREC, 2006.

[46] M. Thelwall, K. Buckley, G. Paltoglou, Sentiment strength detection for the social Web, Journal of the American Society for Information Science and Technology 63 (1) (2012) 163–173.

[47] S. Baccianella, A. Esuli, F. Sebastiani, SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining, in: LREC, 2010.

[48] E. Pianta, L. Bentivogli, C. Girardi, MultiWordNet: developing an aligned multilingual database, in: First International Conference on Global WordNet, 2002.