Extraction de commentaires utilisateurs sur le Web

Julien Subercaze, Christophe Gravier, Frederique Laforest

Extraction de commentaires utilisateurs sur le Web

Résumé. Dans cet article, nous présentons CommentsMiner, une solution d'extraction non supervisée pour l'extraction de commentaires utilisateurs. Notre approche se base sur une combinaison de techniques de fouille de sous-arbres fréquents, d'extraction de données et d'apprentissage de classement. Nos expérimentations montrent que CommentsMiner permet de résoudre le problème d'extraction de commentaires sur 84% d'un jeu de données représentatif et publiquement accessible, loin devant les techniques existantes d'extraction.

1 Introduction

Possessing user-generated contents is one the great challenge in today's Web ecosystem. Companies such as Twitter, Facebook, Instagram, Google, to name a few, have long understood the value of the content produced by their users. They managed to reach millions of users, mainly by offering free high-quality services. Analysing and processing these data raise many interesting research challenges. It is therefore not surprising that content posted on these mainstream platforms are attractive to researchers, especially because they are accessible as structured data through APIs. Although these social media and networks are in the limelight, they however represent only a fraction of user-generated content on the Web. Other user-generated content include reviews, comments, wikis and others to create content on the Web.

Gathering user-generated comments at Web Scale offer not only business opportunities but also research issues. That is the reason why Web content extraction and social media analysis has gained a lot of traction in the past years. However, the comment mining task, expected to be unsupervised for Web-scale extraction, is surprinsingly understudied.

In this paper, we propose `CommentsMiner`, a two-stage algorithm that extracts comments from webpages along their conversational structure. Our approach allows nested comments extraction, which enables conversation extraction, a decisive feature for social analysis. The problem of users' comments extraction can be defined as retrieving the pattern p that was used to generate the HTML fragments embedding users' comments. This pattern is valid at site scale. As a consequence, once the pattern is determined, it can be used to extract comments from every page of the site.

Figure 1 depicts the overall architecture of `CommentsMiner`. In the first stage, we consider the problem of comments and nested comments mining as constrained frequent subtree mining. In the second stage, we learn a ranking model to select the winner subtree among the candidates outputted at the previous step.

`CommentsMiner` achieves a perfect mining score on the TestBed for information extraction from Deep Web (TBDW), and we present further results on a surrogate dataset called NUCE – another technical contribution presented with this paper, publicly available.

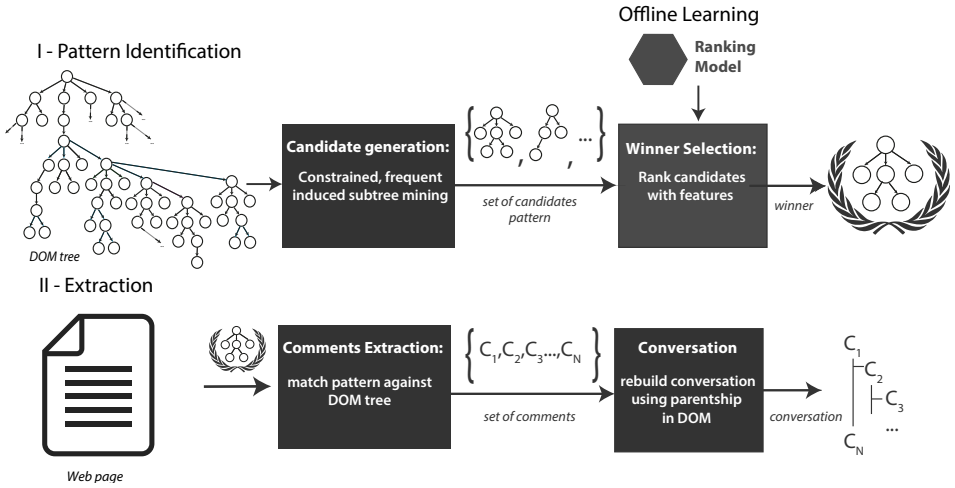

FIG. 1: Outline of CommentsMiner

2 Candidate Generation

In order to mine comments that are nested in a conversational structure, the subtree mining process must be able to skip leaves horizontally. Thus, using bottom-up subtrees would not be sufficient to match this pattern. However the vertical removal that is permitted using embedded subtrees is not a wishful feature and would lead to unnecessary expensive computations. The frequent subtrees we are mining have therefore the property to be induced.

In user-generated comments, the content of comments itself may contain not only user-generated text but also HTML tags. These tags are at the discretion of the users (for instance, `<b>` and `<i>` to decorate text). As a consequence, maximal subtree mining is too restrictive. However closed subtree mining matches our requirement, since it will output both the target and its supertrees: in the case where the comments have different formattings, the target subtree will have a greater support than its supertrees.

For our implementation, we use CMTreeMiner Chi et al. (2005) as the algorithm of our choice – because it is the only one providing frequent subtree mining of ordered induced closed subtrees as reported in da Jiménez et al. (2010). We adapted the algorithm to take into account the constraints described in the next section.

2.1 Constraints

The search space of a DOM tree is very large, DOM tree contains 1300 nodes[1] in average. Regarding the performance reported by the authors of CMTreeMiner, mining a tree of thousand

[1] `http://www.httparchive.org/trends.php?s=Top1000`

nodes with a support of 2 could take minutes, even hours. We define three domain-specific constraints to reduce the search space of the problem.

Lowest common ancestor similarity. As comments are located in a unique area of the DOM tree, occurrences of the target pattern are relatively close to each other. The tree distance between root occurrences of the target pattern is not a priori known, and may vary from page to page. However, the two root occurrences of the target pattern are in the same subbranch of the top tag, i.e. the `<body>` tag. Formally, the *lowest common ancestor* between two occurrences of the pattern cannot be the root of the DOM tree.

Blank occurences deletion. Another simple, yet very efficient constraint is based on the text associated to the occurrences. We discard patterns whose occurrences contains no text or identical text.

Root and rightmost occurrences equality. In CMTreeMiner induced subtrees occurrences are identified during the mining process using their rightmost occurrences. We denote $RootOcc_{t,T}$ and $RmoOcc_{t,T}$ the sets of root and rightmost occurrences of a frequent subtree t in a data-tree T. Each comment has its own root and right most occurrence – they are not shared with other comments. The verification for any candidate subtree is therefore carried out with : $|RootOcc_{t,T}| = |RmoOcc_{t,T}|$.

This stage of the algorithm outputs a set of candidate patterns, among which include the pattern used to generate the comments on the Webpage. The set of candidate patterns is a subset of all generated patterns. To be a candidate for the next stage, a pattern must validate the constraints defined in Section 2.1. In the practice, the number of candidate patterns rarely exceeds twenty.

3 Winner Selection

For a given webpage, the previous subtree mining step outputs a set of candidate patterns. Among these candidates, we aim now at finding the pattern p, that was used as the template. Finding the pattern p, that was used as the template to embed comments, among the subtrees issued by the previous stage can be seen as a ranking problem, where only the first rank matters. To rank these candidates patterns, we use textual and densitometric features as input to learning to rank algorithms, including: SVMRank, MART, RankNet, RankBoost, AdaRank, Random Forests, and Genetic Programming. We first describe the features, then the ranking measure. Experimental results are presented in Section 4.

Features description The main characteristics that distinguish candidates for ranking are both text and densitometric features. One can observe that user-generated content is of variable length Kohlschütter et Nejdl (2008) – unlike menus for which the length and the number of words are very similar among menu items. It is also usually forbidden to include links in comments to avoid spamming, we therefore expect a low density of link in the HTML code. The text density (ratio text vs code) of user-generated comments is also significantly different from the one of boilerplate Agichtein et al. (2008). As the content of user-generated comments differ significantly, the average and the standard deviation of each of these features convey the heterogeneity between occurrences of the same candidate subtree. Therefore we exploit these characteristics as a set of eight features (average and standard deviation from text density, link density, text length and word volume).

Ranking measures Our work deals with a special case of learning to rank, where only the most relevant candidate matters, regardless of other candidates. This kind of binary relevance is usually denoted as *Winner Takes All* (WTA) Xu et al. (2008). Note that `CommentsMiner` relies on a ranking function that must be learnt. However, `CommentsMiner` is considered unsupervised : once the ranking function is learnt, it can be reused for unknown Web domains, and without further learning. This is consistent with the classification introduced by the recent and exhaustive literature review provided in Sleiman et Corchuelo (2013).

Once the winner pattern has been selected by the ranking process, we proceed to the extraction of the comments. As depicted in Figure 1, we first match the pattern against the DOM tree and then rebuild the conversational structure of the comments. Matching the pattern against the DOM tree is performed in linear time using a depth-first strategy (breadth-first is also suitable here): the algorithm first lists all the occurrences in the DOM tree of the root element of the pattern. For each occurrnce, it successively checks that the first child of the pattern is matched in the datatree. The validation continues similarly to a depth-first tree traversal.

4 Experimental results

In this section we first present the baselines and the experimental setup. Then we report and discuss the accuracy and performance of `CommentsMiner` for the comments and their conversational structure extraction task.

Baselines To the best of our knowledge, only MiBat Song et al. (2010) was designed for the comment extraction task. Unfortunately, the materials used in MiBat (software or datasets) are not publicly available, nor upon request. The perfect matching success rate of 75.653% was obtained for several pages belonging to the same Web domain (this is inferred from the illustrations within the paper, yet the precise number is unknown) – this skews the evaluation. Another baseline is DEPTA, a follow-up of MDR (see Section **??**). DEPTA requires a full browser rendering and a visual analytics that result in poor scalability, and it is unable to extract parent-child relationships. While DEPTA is not accessible, MDR can be retrieved online – which makes it the candidate to be considered a standard baseline in several works as reported in the survey Sleiman et Corchuelo (2013).

Other eligible candidates are TPC and RST, yet none are publicly available. They were however evaluated against the same dataset, the TBDW dataset. `CommentsMiner` achieves a success rate of 100% on this dataset, which makes hardly a difference with TPC and RST (resp. 96.23% and 98.06% precision, and resp. 97.03% and 97.88% recall value). Henceforth, we will focus on the more challenging dataset that is NUCE.

Datasets Both TPC Miao et al. (2009) and RST Bing et al. (2011) are competitors to our approach. They were evaluated using the TestBed for information extraction from Deep Web (TBDW). We discuss how `CommentsMiner` performs on this dataset with respect to these competitors in the next section (4). However, there are some primary issues on benchmarking the comments and nested comments extraction task on the TBDW dataset – mainly, it no longer reflect today's Web programming habits Particularly *(i)* `<table />` and `<form />` tags are

no more used to organize page layout, *(ii)* there is no case of nested subregions *(iii)* the average webpage size has increased by 237% from December 2010[2] to February 2015[3].

Since `CommentsMiner` achieve a perfect score for frequent subtree mining for the ground truth offered by the TBDW dataset, we built a more challenging dataset with attributes including: i) up-to-date web programming paradigms, ii) diverse and multilingual web domains, and iii) Webpages with nested regions. We started from Google News in English, French and German. For each domain, we found a page containing more than two comments and downloaded its content through the Web browser in order to avoid AJAX calls issue Gyllstrom et al. (2012). We stricly consider only one page per domain. Some services like Wordpress, Disqus, Livefyre, Facebook, etc., provide commenting features. In order to avoid any bias, we kept one page using each service. The dataset consists in 211 labeled Web pages. We called this labeled dataset NUCE, which stands for Nested User-generated Content Extraction dataset. Our surrogate dataset is publicly available to download, and includes for each page its browser-side rendered webpage as well as the associated ground truth.

Results The evaluation depends on the quality of the learning-to-rank step since the expected pattern p is always included in the set of the pattern candidates set (as discussed in Section 2). We utilized different learning methods for learning to rank pattern candidates. While it is out of the scope of this paper to provide a complete state-of-the-art on learning-to-rank methods, the authors can refer to Cao et al. (2007) and Xia et al. (2008) for further details. The genetic programs were trained using the WTA metric and the following operators were available for the learner : addition, multiplication, substraction, division, power, along with any values in the range $[2; 10,000]$. All learners were trained and tested using the eight features described previously. Results are presented in Table 1. Training was done using 20% data partitioning and a five-fold cross-validation.

Algorithm	Settings	*Mean*	*STD*
MART	$1,000\ trees$	66.4	6.3
SVMRank	$RBF,\ c = 0.1$	77.1	6.3
RankNet	$100\ iterations$	67.1	6.8
RankBoost	$300\ rounds$	84	4.9
AdaRank	WTA for training	66	15.4
Coord. Ascent	WTA for training	81.6	3.6
ListNet	$1,500\ iterations$	90.1	13.5
Gen. Prog.	$\mathbf{50\ iterations}$	**84.3**	**0.8**
MiBAT (*baseline*)	N/A	75.6	Unknown

TAB. 1: Performance and settings of trained learners on the NUCE dataset.

ListNet best model provides a P@1 of 90.170 over 100 runs. However ListNet-based learning-to-rank models suffer from a very significant standard deviation. We conclude that Genetic Programming models are the most suitable for the learning-to-rank step. Although those models do not achieve the best success rate (84.375 in average), it is still very good while providing more guarantees on its generalization. Genetic Programming based models offers a standard deviation of 0.854. Genetic Programming models therefore offer a stability of success rates of the utmost pratical interest for a *learn once, extract many* crawling strategy.

[2] `http://httparchive.org/interesting.php?l=Dec%2028%202010`
[3] `http://httparchive.org/interesting.php?l=Dec%2015%202013`

5 Conclusions and future work

In this paper, we presented `CommentsMiner`, a novel approach to extract user-generated comments. `CommentsMiner` bridges the gap between frequent subtree mining and web information extraction by succesfully extracting HTML templates that embed user-generated comments. A specificity of users comments is their conversational structure. Our approach based on constrained mining of closed frequent subtrees is able to extract nested comments. By constraining the mining process, we are able to avoid the combinatorial explosion that usually characterizes subtree mining. To identify the winner subtree among those output by the mining step, we use a learning-to-rank approach and compared the result of several algorithms. We finally compare our extraction result to existing approaches on both a popular and a surrogate datasets, thus acknowledging the improvement brought by `CommentsMiner`.

References

Agichtein, E., C. Castillo, D. Donato, A. Gionis, et G. Mishne (2008). Finding high-quality content in social media. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pp. 183–194. ACM.

Bing, L., W. Lam, et Y. Gu (2011). Towards a unified solution: data record region detection and segmentation. In *Proc. of the 20th CIKM Conference*, pp. 1265–1274.

Cao, Z., T. Qin, T.-Y. Liu, M.-F. Tsai, et H. Li (2007). Learning to rank: from pairwise approach to listwise approach. In *Proc. of the 24th ICML Conference*, pp. 129–136.

Chi, Y., Y. Xia, Y. Yang, et R. R. Muntz (2005). Mining closed and maximal frequent subtrees from databases of labeled rooted trees. *Knowledge and Data Engineering, IEEE Transactions on 17*(2), 190–202.

da Jiménez, A., F. Berzal, et J.-C. Cubero (2010). Frequent tree pattern mining: A survey. *Intell. Data Anal. 14*(6), 603–622.

Gyllstrom, K., C. Eickhoff, A. P. de Vries, et M.-F. Moens (2012). The downside of markup: examining the harmful effects of css and javascript on indexing today's web. In *Proc. of the 21st CIKM Conference*, pp. 1990–1994.

Kohlschütter, C. et W. Nejdl (2008). A densitometric approach to web page segmentation. In *Proc. of the 17th ACM CIKM Conference*, pp. 1173–1182.

Miao, G., J. Tatemura, W.-P. Hsiung, A. Sawires, et L. E. Moser (2009). Extracting data records from the web using tag path clustering. In *Proc. of WWW*, pp. 981–990.

Sleiman, H. et R. Corchuelo (2013). A survey on region extractors from web documents. *IEEE trans. on Knowledge and Data Engineering 25*(9), 1960–1981.

Song, X., J. Liu, Y. Cao, C.-Y. Lin, et H.-W. Hon (2010). Automatic extraction of web data records containing user-generated content. In *Proc. of CIKM*, pp. 39–48.

Xia, F., T.-Y. Liu, J. Wang, W. Zhang, et H. Li (2008). Listwise approach to learning to rank: theory and algorithm. In *Proc. of the 25th ICML Conference*, pp. 1192–1199.

Xu, J., T.-Y. Liu, M. Lu, H. Li, et W.-Y. Ma (2008). Directly optimizing evaluation measures in learning to rank. In *Proc. of the 31st SIGIR conference*, pp. 107–114.