Comparison of String Similarity Measures for Obscenity Filtering

Ekaterina Chernyak
National Research University
Higher School of Economics
Moscow, Russia
echernyak@hse.ru

Abstract

In this paper we address the problem of filtering obscene lexis in Russian texts. We use string similarity measures to find words similar or identical to words from a stop list and establish both a test collection and a baseline for the task. Our experiments show that a novel string similarity measure based on the notion of an annotated suffix tree outperforms some of the other well known measures.

1 Introduction

String similarity measures are widely used in the majority of Natural Language Processing tasks (Gomaa and Fahmy, 2013), such as spelling correction (Angell et al., 1983), information retrieval (Schütze, 2008), text preprocessing for further classification or clustering (Islam and Inkpen, 2008), duplicate detection (Elmagarmid et al., 2007), etc. The performance and suitability of different string similarity measures has already been demonstrated in an extensive amount of previous work. Here, we study the suitability of different similarity measures as a tool to detect and filter obscene lexis in Russian texts. The goal is to compare the performance of different string similarity measures in finding obscene words and their derivatives. Since the Russian obscene language follows the whole language tendencies, such as highly inflectional morphology, the amount of obscene words and their derivatives is enormous. The words, generated by social network and social media users, may contain not only explicitly obscene words and/or their derivatives, but also their combinations and paronyms. This makes out task specially challenging.

Although the problem is quite different from a single word query retrieval, because there is no need to introduce neither document nor user relevance, we nevertheless exploit IR metrics to evaluate the quality of results.

In this publication, we want to address the following research questions:

- the suitability of using string similarity measures for obscenity filtering in Russian texts, and, if so,
- the choice of the string similarity measure for the task.

2 Related Work

Obscenity and profanity filtering can be seen as a part of developing content filters (such as parental controls (Weir and Duta, 2012)), cyberbullying detectors (Dadvar et al., 2013) and spam filters (Yoon et al., 2010). Another application of obscenity filtering is found in sentiment analysis, where obscene words are treated as indicators of negative (Ji et al., 2013) or sarcastic reviews (Bamman and Smith, 2015). A more complex application of obscenity filtering is identifying implicitly abusive content (Weir and Duta, 2012). In this case not only the usage of obscene language but also the intentions of the author are crucial.

Unlike the current trends in Natural Language Processing obscenity and profanity filtering does not exploit machine learning, but is usually done using rule-based approach. In almost all application a stop list of words, that are considered obscene is required. The task is than to find occurrence of stop word or their derivations.

3 Data and Annotation

The input data set is twofold. First, we used the extensive list of the words, prohibited for url naming in Cyrillic .“рф” domain zone, further referred as the stop list. This stop list was released by

Russian Federal Service for Supervision of Consumer Rights Protection and Human Welfare, responsible for naming in the .“рф” domain zone. The stop list consists of slightly more than 4000 items, all of them being obscene words and their derivatives. Second, we manually created the collection of texts, rich in obscene lexis. To maintain style diversity, we collected texts from various sources, starting from scientific works on Russian obscenity etymology, poems of classical Russian poets (Pushkin, Esenin, Mayakovsky) and post-modern prose (Yu. Aleshkovsky, I. Guberman, V. Sorokin) up to underground music lyrics (by bands Leningrad, Krasnaya Plesen’) and social media sources (Lurkmore, LJ, vk.com, etc).

Next, to minimize the amount of data to be annotated, we tokenized all the the text and removed numbers and punctuation signs and created one frequency dictionary for further annotation. We annotated all unique words in a binary way: a word is either an obscene word (1) or a normal word (0). In total, there were 294916 tokens and 60868 unique words, of them 1261 were annotated as obscene. As we were quite limited in human recourses, the frequency dictionary was split in several annotation tasks in an non-overlapping way, so that one word was considered only by a single annotator.[1] Hence no agreement measures can be computed, although it might be an interesting direction for future work, which will allow to study whether there are any differences in the perception of obscenity.

4 String Similarity Measures

Formally speaking, for every word t from the input frequency dictionary we have to decide whether it is obscene or not. To make this decision we look for the most similar stop word s from the stop list, i.e for $s^* = \texttt{argmax}_{s \in \texttt{stop list}} sim(s,t)$. If $sim(s^*, t)$ is higher than a predefined threshold, we consider t obscene.

4.1 Coincidence

For each word in the frequency dictionary, we check whether the word itself or the lemma of the word of the stem of the word are present in the stop list. To lemmatize words we used two of the available Russian lemmatizers, mystem (Segalovich, 2003), developed by Yandex, and pymoprhy2 (Korobov, 2015), which is an open source project. We also stemmed all the words and the stop words using Porter stemmer (Porter, 2001) and repeated the same procedure for stems: for each word in the frequency dictionary we checked, whether its stem coincides with one of the stop word stems.

4.2 Jaccard Coefficient

Jaccard coefficient is a well-known set-theoretical similarity measure. Given to sets, A and B, their similarity sim is measured as $\frac{|A \cap B|}{|A \cup B|}$. To apply Jaccard coefficient to the measure similarity between two strings, we need to split these string in character n-grams, i.e., sequences of n consequent letters. For example, the Jaccard coefficient for the string “mining” and “dining” based on 3-grams is equal to $\frac{3}{5}$ and based on 4-grams – to $\frac{2}{3}$. In our study we experiment with different values of n from 3 to 6.

4.3 Annotated Suffix Tree

Annotated suffix tree (AST) is a data structure, used to calculate and store all frequencies of all fragments of an input string collection. First introduced for spam filtering (Pampapathi et al., 2006), it was effectively used in a variety of NLP tasks, such as text summarization (Yakovlev and Chernyak, 2016), fuzzy full text search (Frolov, 2016), etc. The AST is an extended version of the suffix tree, which is used for a variety of NLP tasks too (Ravichandran and Hovy, 2002; Zamir and Etzioni, 1998).

To construct an AST for a single string, we need first to split this string in suffixes $s_i = s[i:]$. Next we take the first suffix s_1 and create a chain of nodes in an empty AST with frequencies equal to unity. For all next suffixes we do the following: we check, if there is a path in the AST, which coincides with the beginning of the current suffix, i.e., so-called *match*. If there is such a match for the current suffix in the AST, we increase the frequencies of the matched nodes by unity and add the not matched characters to end of the match, if any. Same way can construct a generalized AST for the collection of input strings. Fig. 1 shows and example of a generalized AST for string “mining” and “dining”.

We adopt a scoring procedure from (Pampapathi et al., 2006) and use it as a similarity measure. Briefly, the scoring procedure computes average frequency of the input string in the AST. Given again a string s, we split it in the suffixes s_i. The

[1]The annotated frequency dictionary is available at https://github.com/echernyak/filter

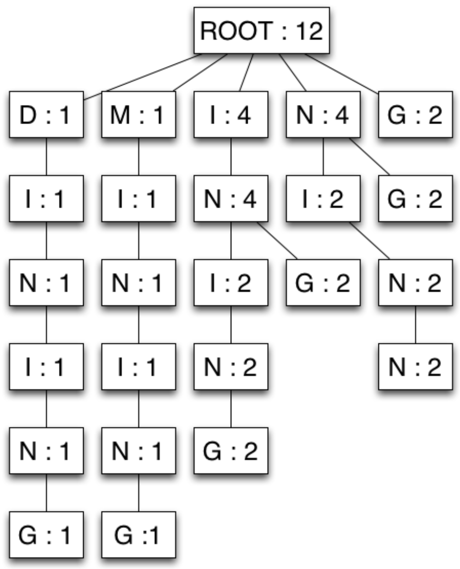

Figure 1: The generalized AST for string "mining" and "dining".

first step of scoring is to match and score each suffix individually:

$$score(match(s_i, AST)) = \frac{\sum_{n \in match} \frac{f(n)}{f(p(n))}}{|match|}$$

where $f(n)$ is the frequency of the node n and $f(p(n))$ is its parent frequency. Next, we sum up the individual scores and weight them by the length of the string:

$$SC(s, AST) = \frac{\sum_{s_i}(score(match(s_i, AST)))}{|s|}.$$

The final SC function may serve as string similarity function.

For our task we construct one generalised AST from the stop list and match and score each word to this AST. Based on the achieved values we decide, is the word obscene or not.

4.4 Edit Distance

Edit distance, also known as Levenshtein distance, stands for the number operations needed to transform a string s_1 into a string s_2, given that they are generated from the common alphabet Σ. Usually the possible operations are limited to insertion, deletion and substitution. For example, the edit distance between strings "mining" and "dining" is equal to 1, since only one substitution operation is required to transform one string into another.

5 Evaluation

Note, that for different similarity measures both the range and the threshold differ. For example, the word, lemma or stem coincidence coincidence results only in two values, namely, 0 and 1. Jaccard and AST-based similarity measures range between $[0, 1]$, while the edit distance has no upper bound. Hence, the thresholds are defined in in various ways: the lemma or stem coincidence should be equal to unity to consider the word obscene. We tested Jaccard similarity measure with the threshold equal to 0.8, the edit distance with threshold equal to 5 and 8. For the AST-based similarity measure the value of 0.2 has proven to be a more or less meaningful threshold, since it is around 1/3 of the maximal observed similarity value (Pampapathi et al., 2006; Frolov, 2016; Yakovlev and Chernyak, 2016).

	time complexity
word, lemma or stem coincidence	$O(n * m)$ to check symbol-wise coincidence with each stop word
AST-based similarity measure	$O(m^2)$ to check suffix-wise coincidence with an AST build for the stop list
Jaccard similarity measure	$O(n^2 * m)$ to check all possible pairs of a word a and stop word
edit distance	$O(n^2 * m)$ to check all possible pairs of a word a and stop word

Table 1: Time complexity of exploiting different similarity measures.

After we get a set of candidate obscene words using one of the similarity measures, we can evaluate it by such standard measures, as recall, precision, F-measure and accuracy.

Of these four measures we would consider recall the most important one, since a good filter should have as few false negatives as possible and the number of false positives is not that crucial in our task.

The last but not least feature for comparison of string similarity measure in task of obscenity filtering is the time complexity of computing similarity values. Since the obscenity filtering is likely to be done online, the method used should be as fast as possible. Let us list the time complexity of exploiting different similarity measures using the following O – notation and the following notations: let n be the number of stop words, m – the maximal length of a stop word, $m \ll n$, see Table 1.

6 Results and Discussion

Final results are presented in Table 2 below. If we take precision into account, obviously the best results are achieved by using word coincidence, followed by lemma and stem coincidence. Altough there is no drastic difference between using pymorphy2 or mystem lemmatizers, the latter gives better results than the former. Stemming works slightly worse, than lemmatisation. The precision of using Jaccard coefficient is almost comparable to the one, achieved by word coincidence, with recall being slightly higher. The precision of AST-based similarity measure and edit distance is significantly lower than everything else.

If we consider recall now, the best results are achieved by using edit distance, although the precision of this method is almost close, which does make the results unreliable. The edit distance is followed by AST-based similarity, which overcomes the stem coincidence by almost 20%.

To evaluate the over-all performance we may use accuracy or F-measure. From this point of view, the highest results are achieved by using stem or lemma coincidence, followed by AST-based similarity and Jaccard coefficient.

Let us analyze errors (i.e. false positive and false negative words). During our experiments we noticed the following possible errors:

1. very short words, such as “уг” [abbreviation for “depressing shit”] or “ссы” [“to piss”] result usually in false negatives for the AST-based similarity;

2. long or event compound words, such as “говнофотограф” [“bad photographer”], “скопипиздить” [“to copy paste illegally”] are tough for all measures and result in false negatives. The only measure that is capable to discover such words is the AST-based similarity measure due to it suffix nature;

3. the AST-based similarity measure usually considers verbs as obscene words, which increases the number of false positives. For example, all verbs, that end with “ать” [verbal ending “at́”] tend to be considered as obscene;

4. the Jaccard coefficient suffers from paronyms, such as “эксперименты” [“experiments”] – “экскременты” [“excrement”], which increase the number of false positives;

5. the pure results of edit distance are caused by the substitution of wrong symbols. For example, the word “манере” [“manner”] has edit distance equal to 3 to the word “засере” [“young punk”], although it is not obscene by now means.

	Pr	R	acc	F_2
word coincidence	**0.7288**	0.1363	0.9810	0.2297
lemma coincidence				
pymorphy2	0.6492	0.2466	0.9815	0.3574
mystem3	0.6807	0.3195	**0.9827**	0.4349
stem coincidence	0.6113	0.4028	0.9822	**0.4856**
AST	0.1578	**0.6201**	0.9233	0.2516
Jaccard similarity measure, 0.8				
3-grams	0.6799	0.1633	0.9810	0.2634
4-grams	0.7126	0.1475	0.9810	0.2430
5-grams	0.7168	0.1284	0.9808	0.2179
6-grams	0.6989	0.0975	0.9803	0.1711
edit distance				
$d < 8$	0.0234	**0.9127**	0.8086	0.0456
$d < 5$	0.0209	**0.9825**	0.9629	0.0409

Table 2: Comparison of results.

To cope with some of the errors, we might exploit additional POS filtering and preprocessing as well as some compound splitting algorithms. Anyway it remains an open question whether the edit distance is applicable for the task at all.

7 Conclusions

In this project we establish both a text collection and a baseline for both obscene filtering. We have so far achieved quite moderate results, which nevertheless allow us to make some preliminary conclusions and think of the future directions for improvement.

1. Straightforward similarity measures such as word, lemma or stem coincidence do not cope well with the problem of obscene filtering, no matter what lemmatisation tool or stemming algorithm is used;

2. If we consider recall as the main quality measure, the best results are achieved either AST-based similarity measure or Jaccard coefficient on character n-grams;

3. The edit distance is of too general nature to be applicable for the problem;

4. If the filtering should be conducted online, the AST similarity measure is the best one in terms of time complexity of calculations.

Our main future directions are, first of all, improvements based on conducted error analysis, and, secondly, developing a filter for obscene multiword expressions, such as послать на хуй" ["to fuck off"] and euphemisms, such as послать на три буквы" ["to fuck off"]. The filtering of obscene multiword expressions might be seen as a problem analogous to semantic role labelling, where the obscene word is the main one and the rest are its arguments. The filtering of euphemisms looks much more complicated to us and may require using compositional semantics tools.

Acknowledgements

This work was supported by RFBR grants #16-01-00583 and #16-29-12982 and was prepared within the framework of the Basic Research Program at the National Research University Higher School of Economics (HSE) and supported within the framework of a subsidy by the Russian Academic Excellence Project "5-100".

References

Richard C. Angell, George E. Freund, and Peter Willett. 1983. Automatic spelling correction using a trigram similarity measure. *Information Processing & Management*, 19(4):255–261.

David Bamman and Noah A. Smith. 2015. Contextualized sarcasm detection on Twitter. In *Ninth International AAAI Conference on Web and Social Media*.

Maral Dadvar, Dolf Trieschnigg, Roeland Ordelman, and Franciska de Jong. 2013. Improving cyberbullying detection with user context. In *European Conference on Information Retrieval*, pages 693–696. Springer.

Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. 2007. Duplicate record detection: A survey. *IEEE Transactions on knowledge and data engineering*, 19(1).

Dmitry Frolov. 2016. Using annotated suffix trees for fuzzy full text search. In *Communications in Computer and Information Science, Information Retrieval, 10th Russian Summer School, RuSSIR*. Springer.

Wael H. Gomaa and Aly A. Fahmy. 2013. A survey of text similarity approaches. *International Journal of Computer Applications*, 68(13).

Aminul Islam and Diana Inkpen. 2008. Semantic text similarity using corpus-based word similarity and string similarity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(2):10.

Xiang Ji, Soon Ae Chun, and James Geller. 2013. Monitoring public health concerns using twitter sentiment classifications. In *Healthcare Informatics (ICHI), 2013 IEEE International Conference on*, pages 335–344. IEEE.

Mikhail Korobov. 2015. Morphological analyzer and generator for Russian and Ukrainian languages. In *International Conference on Analysis of Images, Social Networks and Texts*, pages 320–332. Springer.

Rajesh Pampapathi, Boris Mirkin, and Mark Levene. 2006. A suffix tree approach to anti-spam email filtering. *Machine Learning*, 65(1):309–338.

Martin F. Porter. 2001. Snowball: A language for stemming algorithms.

Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 41–47. Association for Computational Linguistics.

Hinrich Schütze. 2008. Introduction to information retrieval. In *Proceedings of the international communication of association for computing machinery conference*.

Ilya Segalovich. 2003. A fast morphological algorithm with unknown word guessing induced by a dictionary for a web search engine. In *MLMTA*, pages 273–280. Citeseer.

George R.S. Weir and Ana-Maria Duta. 2012. Strategies for neutralising sexually explicit language. In *Cybercrime and Trustworthy Computing Workshop (CTC), 2012 Third*, pages 66–74. IEEE.

Maxim Yakovlev and Ekaterina Chernyak. 2016. Using annotated suffix tree suffix tree similarity similarity measure for text summarisation. In *Analysis of Large and Complex Data*, pages 103–112. Springer.

Taijin Yoon, Sun-Young Park, and Hwan-Gue Cho. 2010. A smart filtering system for newly coined profanities by using approximate string alignment. In *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, pages 643–650. IEEE.

Oren Zamir and Oren Etzioni. 1998. Web document clustering: A feasibility demonstration. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 46–54. ACM.