Annotated Suffix Trees for Text Clustering

Ekaterina Chernyak and Dmitry Ilvovsky

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

Abstract. In this paper an extension of *tf-idf* weighting on annotated suffix tree (AST) structure is described. The new weighting scheme can be used for computing similarity between texts, which can further serve as in input to clustering algorithm. We present preliminary tests of using AST for computing similarity of Russian texts and show slight improvement in comparison to the baseline cosine similarity after applying spectral clustering algorithm.
Keywords: annotated suffix tree, clustering, similarity measure

1 Introduction

The text clustering applications exploit two major different clustering approaches: either a text is represented as a vector of features, and distance-based algorithms (such as k-Means) are used, or the similarity between texts are computed and the similarity-based algorithms (such k-Medoids or Normalized cuts) are used. While the former requires to extract features from texts, the latter requires the definition of similarity measure. The other algorithms, such as Suffix Tree Clustering [8] explore internal features for the text collection and find clusters straightforward. We will concentrate of the preliminary step of applying distance-based algorithms, i.e. computing similarity between texts. According to [2], there are several approaches to computing text similarity: there are string-based (which include character-based and term-based), corpus-based and knowledge-based approaches. This project belongs to the characters-based approach, which means, we do not take corpora data or semantics into account. We describe a new similarity measure, which is based on the notion of an annotated suffix tree and present an example of using this measure.

2 Annotated suffix tree

The suffix tree is a data structure used for storing of and searching for strings of characters and their fragments [3]. An annotated suffix tree (AST) is a suffix tree whose nodes are annotated by the frequencies of the strings fragments. An algorithm for the construction and the usage of AST for spam-filtering is described in [6]. Some other applications are described in [4, 5].

2.1 Definition of AST

According to the annotated suffix tree model [4–6], a text document is a set of words or word n-grams, which we will address as strings. An annotated suffix tree is a data structure used for computing and storing all fragments of the strings and their frequencies (see Fig. 1 for an example of the AST for the string "mining"). It is a rooted tree in which:

– Every node corresponds to one character
– Every node is labeled by the frequency of the text fragment encoded by the path from the root to the node.

.

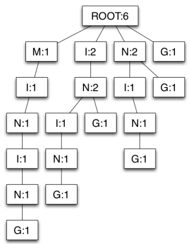

Fig. 1. AST for string "mining"

The AST has two important proprieties: the frequency of a parent node is equal to:

1. the sum of the frequencies of children nodes;
2. the sum of the frequencies of underlying leaves.

According to these properties we can calculate the frequency of the root: it is equal to the sum of the frequencies on the first level of the AST. For example, the frequency of the root of the AST in Fig. 1 is equal to 2+2+1+1 = 6.

2.2 Similarity measure

To estimate the similarity between two texts we find the common subtree of the corresponding ASTs. We do the depth-first search for the common chains of nodes that start from the root of the both ASTs. After the common subtree is constructed we need to annotate and score it.

We annotate the common subtree in the following way. A new node of a common subtree is annotated by two numbers:the minimum and the maximum frequencies of the corresponding nodes of original ASTs.

Let us provide an example of common subtree construction and annotation. Given two ASTs:

– for two strings “mining” and “dining” in Fig. 2
– for one string “dinner” in Fig. 3

we construct the common subtree for them. There are three common chains, which start from the roots: “D I N”, “I N”, “N”. All the nodes of the chain “D I N” have frequencies equal to unity in both ASTs, so in the common subtree the minimum and the maximum frequencies of all three nodes coincide and are equal to unity. The chain “I N” occurs once in the AST in Fig. 3 and four times in the AST in Fig. 2, hence the minimum frequencies are equal to unity and the maximum frequencies are equal to four. The node “N” is annotated by four in the AST in Fig. 2 and by two in the AST in Fig. 3. So its minimum and maximum frequencies are equal to two and four, respectively.

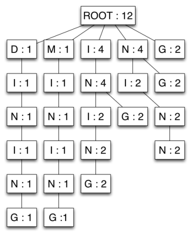

Fig. 2. AST for strings “mining” and “dining”

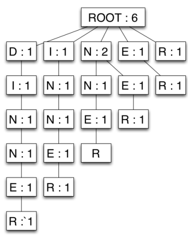

Fig. 3. AST for string “dinner”

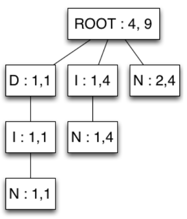

Fig. 4. Common subtree of ASTs in Fig. 2 and Fig. 3

Following the general principle of the AST-based string to text scoring we suggest to score the common subtree in several steps:

1. weighting each node by computing the mean between two frequencies. At this step we can use different type of means, such as geometric mean or harmonic mean. For the sake of simplicity we use further the arithmetic mean;
2. scoring every chain of the common subtree;
3. summing up all chain scores and standardizing them by dividing by the number of chains;

To score the chain we compute the sum of the node frequencies divided by their parents frequencies, which is again divided by the length of the chain:

$$\texttt{score}(chain) = \frac{\sum_{node \in chain} \frac{f^{node}}{f^{parent}}}{|chain|} = \frac{\sum_{node \in chain} \frac{(f^{node}_{min} + f^{node}_{max})/2}{(f^{parent}_{min} + f^{parent}_{max})/2}}{|chain|}$$

For example, the scoring of the common subtree in Fig. 4 is computed as:

$$\frac{\texttt{score(``D I N'')} + \texttt{score(``I N'')} + \texttt{score(``N'')}}{3},$$

where

$$\texttt{score(``D I N'')} = \frac{\frac{(1+1)/2}{(4+9)/2} + \frac{(1+1)/2}{(1+1)/2} + \frac{(1+1)/2}{(1+1)/2}}{3} = \frac{\frac{2}{13} + 1 + 1}{3} = 0.718$$

$$\texttt{score(``I N'')} = \frac{\frac{(1+4)/2}{(4+9)/2} + \frac{(1+4)/2}{(1+4)/2}}{2} = \frac{\frac{4}{13} + 1}{2} = 0.6538$$

$$\texttt{score(``N'')} = \frac{\frac{(2+4)/2}{(4+9)/2}}{1} = \frac{6}{13} = 0.4615$$

and the final scoring is:

$$\frac{0.718 + 0.6538 + 0.4615}{3} = 0.6111$$

At this point the scoring of the common subtree is based on using only the frequencies of the strings and their fragments. To make the scoring analogous to computing *tf*-*idf* we can introduce the *idf*-like component to the scoring.

Let us think about a collection of texts. As a source for *idf* values we can construct a global AST from the whole text collection. To construct the global AST we split every text in strings and exclude from further computations repeating strings and construct the AST then from these unique strings. This way we calculate not the frequencies of the strings, but the number of texts where every string and their fragments occur, that is exactly the *df*'s of all the possible fragments of the texts.

To combine common subtrees and the global tree, we update the chain scoring step:

$$\texttt{score}(chain) = \frac{\sum_{node \in chain} \frac{f^{node}}{f^{parent}} \times \frac{df^{node}}{df^{parent}}}{|chain|},$$

where *df* are extracted from the global tree.

2.3 Construction of AST

It is possible to construct an AST straightforward form a set of tokens using quite an obvious iterative algorithm, which requires splitting each token in suffixes and adding them consecutively to the AST [5]. However, is is shown in [1], that it is more time- and space-efficient to construct a suffix tree, using one of the well-known algorithms and than annotate the tree with the frequencies.

3 Evaluation

We manually created the text collection for further testing of the proposed algorithm. The collection consists of 50 documents in Russian, every 10 text devoted to different definitions of the word "jaguar": an animal, a car, a beverage, a film or a sewing machine. We supposed, that the clusters we would achieve should coincide with the predefined text classes, i.e. we should get five clusters, every cluster corresponding to the initial class. We used the Shi-Malik algorithm [7] with the default parameters to cluster the similarity matrices. Two approaches to the similarity matrix construction:

1. the *tf-idf* transformation and the cosine similarity
2. the AST technique, presented above.

To compare these approaches we computed the number of errors in the achieved clusters. Given a cluster we found the mode value of the class and calculated how many documents do not belong to this class. The higher this number is, the worse is the result of clustering. Using the cosine similarity and Shi-Malik algorithm for finding five clusters, we achieved four perfect cluster and one cluster, that contained six errors. Hence 44 texts were clustered correctly and 6 were not. Using the AST technique, we got only 2 errors, which means that 48 texts were clustered correctly. The results of clustering are presented in Table 1.

Fig. 5 and Fig. 6 present the heat maps of both similarity measures and reveal some issues of the suggested AST technique. First of all, when the AST technique is applied to compute the similarity of a text to itself, the result is not equal to unity. What is more, for different texts it results in different values. Second, all the similarity values are more or less the same, there is no drastic difference at all between the inside or outside classes values. These are the issues to be solved in the future.

Table 1. Clustering quality

	Accuracy	# of errors
cosine similarity	0.88	6
AST similarity	0.96	2

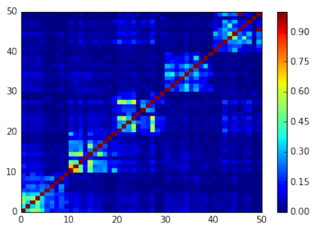

Fig. 5. Heat map of the cosine similarity matrix

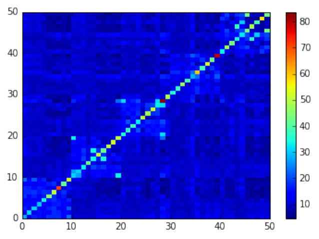

Fig. 6. Heat map of the AST similarity matrix

4 Conclusion

We suggest a new text similarity measure, which is based on annotated suffix trees. To estimate the similarity between text A and text B, one should construct two annotated suffix trees, find the common subtree and score it. The scoring can be extended by document frequencies, if a text collection is given. The preliminary experiments show, that although the proposed similarity measure has some clear advantages in comparison to the baseline cosine similarity, because of being more robust, some formal aspects should be improved. For example, currently the similarity of a text to itself is not equal to unity, which affects the visualisation. Obviously, more experiments should be conducted to find other limitations and possible improvements.

Acknowledgments The paper 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”. The authors was partially supported by Russian Foundation for Basic Research, grants no. 16-29-12982 and 16-01-00583.

References

1. Dubov, M.: Text Analysis with Enhanced Annotated Suffix Trees: Algorithms and Implementation. In Analysis of Images, Social Networks and Texts, 2015, pp. 308-319. Springer International Publishing.
2. Gomaa, W. H., Fahmy, A. A.: Survey of text similarity approaches. International Journal of Computer Applications, 2013, 68(13).
3. Gusfield D.: Algorithms on Strings, Trees, and Sequences, Cambridge University Press, 1997.
4. Chernyak E.L., Chugunova O.N., Askarova J.A., Nascimento S., Mirkin B.G., Abstracting concepts from text documents by using an ontology, in Proceedings of the 1st International Workshop on Concept Discovery in Unstructured Data. 2011, pp. 21-31.
5. Chernyak E.L., Chugunova O.N., Mirkin B.G.: Annotated suffix tree method for measuring degree of string to text belongingness, Business Informatics, 2012. Vol. 21, no.3, pp. 31-41 (in Russian).

6. Pampapathi R., Mirkin B., Levene M.: A suffix tree approach to anti-spam email filtering, Machine Learning, 2006, Vol. 65, no.1, pp. 309-338.
7. Shi, J., Malik, J.: Normalized cuts and image segmentation. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2000, 22(8), 888-905.
8. Zamir, O., Etzioni, O.: Web document clustering: A feasibility demonstration. In Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval (pp. 46-54). 1998, ACM.