Tochtermann, Maurer (Eds.): Proceedings of the I-KNOW '05, Graz
5th International Conference on Knowledge Management
Journal of Universal Computer Science, pp. 596-603, ISSN 0948-695x

The Suffix Tree Document Model Revisited

Sven Meyer zu Eissen
(Paderborn University, Germany
smze@upb.de)

Benno Stein
(Bauhaus University Weimar, Germany
benno.stein@medien.uni-weimar.de)

Martin Potthast
(Paderborn University, Germany
beebop@upb.de)

Abstract: In text-based information retrieval, which is the predominant retrieval task at present, several document models have been proposed, such as boolean, probabilistic, or (extended) vector models [Baeza-Yates and Ribeiro-Neto 1999]. Interestingly, the suffix tree document model is usually not discussed in the literature on the subject though it comes along with a property that sets it apart from the other models: It encodes information about word order. The suffix tree document model owes much of its popularity from the Vivísimo search engine, which operationalizes on-the-fly categorization of Internet search results.

While the classical document models can be considered as vectors of words, the suffix tree document model as well as the related similarity measures are graph-based. Both types of document models provide an efficient means to compute document similarities, and, according to various publications, both types of document models work well in practice. However, there is no comparison between both paradigms that explains the concepts of one in terms of the other, or that contrasts their advantages and disadvantages with respect to certain retrieval tasks. In this paper we start to tackle this gap by shading light on the following questions: (1) How does similarity computation work in the suffix tree document model? (2) Based on the insights of Question 1, is it possible to combine concepts of both document model types within classification or categorization tasks? (3) Which of the document model types is more powerful with respect to unsupervised document classification?

Key Words: information retrieval, document models, suffix tree clustering, document similarity

Category: H.3 Information Storage and Retrieval

1 Document Models

A retrieval task defines a relation between a user, a query, and a set of documents to be searched. The operationalization of a retrieval task is based on a conceptualization of these determinants and must provide, among others, a document model and a query-specific similarity concept—while capturing the user's information need as well as considering efficiency issues.

A document model is a concept that describes how a meaningful set of features, $\mathbf{d}$, is computed from a document's (preprocessed) words. "Meaningful" means that a function φ can be stated that maps from the feature sets $\mathbf{d_1}$ and $\mathbf{d_2}$ of two documents d_1, d_2 into the interval $[0;1]$ and that has the following property: If $\varphi(\mathbf{d_1}, \mathbf{d_2})$ is close to one then the documents d_1 and d_2 are similar; likewise, a value close to zero indicates a high dissimilarity.

1.1 Vector-Based Document Models

According to [Stein and Meyer zu Eißen 2004a], vector-based document models can be classified with respect to four dimensions:

1. A term concept, which defines the granularity of the text units that are used as features: words, n-grams, noun phrases, clauses, etc.
2. A term weighting scheme, which defines how a measure of importance for an individual term is computed.
3. Linguistic statistics like syntactic group analysis or part-of-speech analysis.
4. Simple text statistics and presentation-related statistics.

The first two of these dimensions form the basis of models used for topic-centered similarity assessment tasks like text categorization, the others play a major role in connection with text genre analysis or text synthesis [Burrows 1987; Finn and Kushmerick 2003; Roussinov et al. 2001; Meyer zu Eißen and Stein 2004].

Let n be the number of documents in a corpus, and let m be the total number of different words after preprocessing, also called the dictionary or bag of words. In its simplest form a model of a document d is a vector of length m whose ith entry indicates whether or not the ith word of the dictionary occurs in d. I. e., documents are considered as vectors in the m-dimensional space of all dictionary entries, and hence this model is called vector space model. It can be extended by introducing weights instead of Boolean values. A widely accepted variant combines the (normalized) term frequency, tf, with the inverse document frequency, idf. In particular, $tf(i,j)$ denotes the frequency of term i in document j, and $idf(i)$ may be defined as $\log(\frac{n}{df(i)})$, where $df(i)$ is the number of documents that contain the term i. The hypothesis behind the idf-weighting scheme is that terms which occur only in few documents are of highly discriminative power.

1.2 The Suffix Tree Document Model

Observe that vector-based document models encode no information about the order by which the words occur in a document [see 1]. A more sophisticated document model that preserves the complete word order information is the suffix tree document model; it defines the similarity between two documents in terms of string overlaps in their common suffix tree.

The ith suffix of a document $d = w_1 \dots w_m$ is the substring of d that starts with word w_i. A suffix tree of d is a labeled tree that contains each suffix of d along a path whose edges are labeled with the respective words. The construction of a suffix tree is straightforward: The ith suffix of d is inserted by checking whether some edge emanating from the root node is labeled with w_i. If so, this edge is traversed and it is checked whether some edge of the successor node is labeled with w_{i+1}, and so on. If,

[1] Some kind of “weak” order information can be introduced by using phrases or just every sequence of n consecutive words, so-called n-grams, instead of single words.

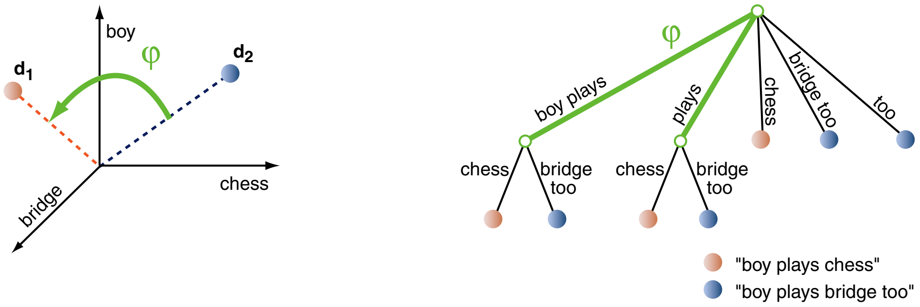

Figure 1: Illustration of the two document model types. The left-hand side shows two documents under the vector-based paradigm; the underlying dictionary contains the words "boy", "chess", and "bridge". As similarity function φ the cosine similarity is shown, which corresponds to the cosine of the angle between $\mathbf{d_1}$ and $\mathbf{d_2}$. The right-hand side shows a suffix tree for the documents "boy plays chess" and "boy plays bridge too". Here, the similarity function φ must quantify the portion of the overlap, which corresponds to the green (thick) edges in the graph.

in some depth k, a node n without a matching edge is reached, a new node is created and linked to node n with an edge labeled with w_{i+k}.

Remarks. Document models and similarity functions φ determine each other: Vector-based document models are amenable to the cosine similarity in first place. The suffix tree document model requires a measure that assesses the similarity between two graphs. Figure 1 illustrates both paradigms.

2 A Closer Look to the Suffix Tree Document Model

In this section we introduce a generic similarity measure for the suffix tree document model. Moreover, we argue that the well-known similarity concepts of the classical document models have their counterpart in the suffix tree document model. Our generic view enables us to seamlessly understand the famous suffix tree clustering algorithm of Zamir and Etzioni as a heuristic to efficiently evaluate the graph-based similarity measure for large document collections.

2.1 A Graph-Based Similarity Measure

As pointed out above, a document model in the form of a suffix tree preserves full word order information. This section introduces a measure that quantifies the similarity of two documents under the suffix tree document model.

Let d^+, d^- designate two documents that are inserted into an initially empty suffix tree T. Each edge e in T gets either labeled "+", "-", or "+-", depending on whether or not e has been traversed while inserting a suffix from d^+ or d^-. Moreover, let E denote the edges in T, and let E^+ and E^- denote those edges in E whose label contain a "+" and a "-" respectively. Then the suffix tree similarity φ_{ST} is defined as

$$\varphi_{ST} = \frac{|E^+ \cap E^-|}{|E^+ \cup E^-|}$$

Obviously, φ_{ST} fulfills the following properties of a similarity measure:

1. *Normalization.* From $0 \leq |E^+ \cap E^-| \leq |E^+ \cup E^-|$ follows that $\frac{|E^+ \cap E^-|}{|E^+ \cup E^-|} \in [0, 1]$.
2. *Reflexivity.* If $d^+ = d^-$ holds, then $E^+ = E^-$, and consequently $|E^+ \cap E^-| = |E^+ \cup E^-|$ and $\varphi_{ST} = 1$.
3. *Symmetry.* The symmetry property follows directly from the fact that the insertion order does not affect edge labeling.

φ_{ST} is the Jaccard coefficient of these edge sets, E^+ and E^-. Other possibilities to measure the match between the two sets include the Dice coefficient, the cosine coefficient, or the overlap coefficient [Rijsbergen 1979]. Observe that φ_{ST} quantifies the frequencies of suffixes in a Boolean sense, since it is not recorded how often an edge is traversed while inserting suffixes of d^+ and d^-. Put another way, φ_{ST} captures "word order matches" rather than term frequencies.

There are two ways to incorporate term frequencies in φ_{ST}. One possibility is to combine φ_{ST} with a traditional vector space model similarity measure by means of a weighted sum, say, $\varphi_{HYB} = \lambda \cdot \varphi_{ST} + (1 - \lambda) \cdot \varphi_{\text{cos}}$, with $\lambda \in [0, 1]$. Alternatively, frequency information for each edge can be recorded during the construction of T. The latter approach has the advantage that frequency information for word sequences that are longer than one (suffix frequencies) can be considered for similarity computation.

We construct the suffix tree as described above; all suffixes of d^+ and d^- are inserted into an initially empty tree T. During insertion the functions $n^+(e)$ and $n^-(e)$ are computed, which define for each edge e how often it is traversed. Then the similarity value φ_{STF} that incorporates suffix frequencies is given as

$$\varphi_{STF} = \frac{1}{|E|} \sum_{e \in E} \frac{\min\{n^+(e), n^-(e)\}}{\max\{n^+(e), n^-(e)\}}$$

The properties of normalization, reflexivity, and symmetry also hold for φ_{STF}. Note that $n^+(e)$ and $n^-(e)$ capture the term frequencies of d^+ and d^- for all edges e that are incident with T's root.

Research on vector space models has shown that term weighting schemes for document collections that rely on both term frequency and inverse document frequency outperform schemes that are based on only one of these concepts [Sparck-Jones 1972]. Note that under the suffix tree document model the inverse document frequency can also be measured for a document collection $D = \{d_1, \ldots, d_n\}$: We construct a suffix tree T for all suffixes of the $d_i \in D$ and associate an initially empty set S with each edge $e \in T$. If a suffix of d_i creates or traverses e, we set $S(e) := S(e) \cup \{i\}$. Since $|S(e)|$ captures the document frequency of the suffix that is represented by the path that starts at the root and ends with e, the inverse document frequency can be measured by $IDF(e) = \log(n/|S(e)|)$, leading to

$$\varphi_{STFIDF} = \frac{1}{|E|} \sum_{e \in E} \frac{\min\{n^+(e), n^-(e)\}}{\max\{n^+(e), n^-(e)\}} \cdot IDF(e)$$

2.2 STC: A Fusion Heuristic for the Suffix Tree Document Model

Given a similarity measure like the cosine similarity for the vector space model or one of the suffix tree similarity measures introduced above, the construction of a similarity graph is in $O(n^2)$ for a document collection D of size n. However, [Zamir and Etzioni 1998] introduced the suffix tree clustering algorithm (STC), which runs in $O(n)$ without computing $O(n^2)$ similarity values. In detail, STC is made up of three steps.

Step 1. A suffix tree for all suffixes of each document in $D = \{d_1, \ldots, d_n\}$ is constructed, and each suffix is associated with the set of documents wherein it is contained. In other words, using the notation given above, for each edge e (each of which represents a certain suffix) the set $S(e)$ is computed. The sets $S(e)$ with $|S(e)| \geq 2$ are called "base clusters" and identify the documents d_i with $i \in S(e)$.

Step 2. Each base cluster is assigned a score f, which is a function of $|S(e)|$ and the length of the suffix that is represented by e. In [Zamir and Etzioni 1998] the authors propose f as the product of $|S(e)|$ and the length of the suffix that is represented by e.

Step 3. The k base clusters $S_1, \ldots, S_k$ that score best under f are selected. A similarity graph in which the base clusters form the node set is generated, and an edge between two nodes S_i and S_j is added if the Jaccard coefficient of S_i and S_j is larger than 0.5, say, when $\frac{|S_i \cap S_j|}{|S_i \cup S_j|} > 0.5$. The connected components of this graph form the final clusters.

2.3 Analysis of the STC Heuristic

STC has proven to work well on document snippets that are returned by search engines [Zamir and Etzioni 1998], but its properties have not been analyzed yet. As pointed out above STC is a heuristic which is highly efficient, but which also has some drawbacks. The following observations will provide a rationale for some of STC's characteristics.

Non-Exclusiveness. Documents may be associated with several base clusters. Consequently, the documents may appear in more than one of the found categories.

Incompleteness. A clustering that is generated by STC does not necessarily contain all documents of the original collection. An incomplete categorization happens for document collections which comprise documents that share only few short word sequences with the remaining documents. The reason for this behavior is that the emerging base clusters will not score high.

DF-based. A base cluster scores higher if the document frequency of its associated suffix increases. This can lead to big clusters, because it is likely that high-scoring base clusters that contain terms with a high document frequency share more than half of their associated documents with other base clusters and consequently are merged in Step 3 of the STC algorithm.

Drifting. Suppose that four base clusters, S_1, S_2, S_3, S_4, are given, where $S_1 = \{1, 2, 3\}$, $S_2 = \{2, 3, 4\}$, $S_3 = \{3, 4, 5\}$, and $S_4 = \{4, 5, 6\}$. Then all documents $d_1, \ldots, d_6$ are merged into a single cluster because the Jaccard coefficient of S_i and S_{i+1} is larger than 0.5. In particular, this single cluster comprises the documents that are associated with S_1 and S_4, which might be completely dissimilar.

Absoluteness. STC considers two documents as similar and assigns them to the same base cluster if they share a rather long suffix or several short suffixes. Regardless of whether their base cluster is merged with other base clusters in Step 3, their base cluster is part of the final clustering. Note that no information about document lengths or suffix mismatches of the rest of those documents is computed, resulting in poor quality clusters. This point is relatively unimportant for short documents—a fact that explains the good performance of STC on document snippets like those returned by search engines.

Topic Generating. Each base cluster is associated with a suffix, which can serve as a label for this cluster. This method solves two basic problems in topic identification for document clusters [Stein and Meyer zu Eißen 2004b]: word order preservation and topic length determination.

3 Quantitative Analysis

The purpose of our experiments is twofold. Firstly, we want to gain evidence on STC's character as a heuristic, say, to measure how good the STC algorithm performs on popular document collections compared to clustering algorithms that rely on the new suffix tree similarity measures or the traditional vector space similarity measures. Secondly, we want to answer the question to which extent word order preservation improves clustering performance. For this purpose we evaluated STC and the clustering algorithms MajorClust [Stein and Niggemann 1999] and Group Average Link using the discussed similarity measures on several categories drawn from RCV1 [Rose et al. 2002].

3.1 Document Sets

RCV1 is a document collection that was published by the Reuters Corporation for research purposes. It contains about 800,000 documents each of which consisting of a few hundred up to several thousands words. The documents have been manually enriched by meta information like category (also called topic), geographic region, or industry sector. RCV1 comprises 103 different categories, arranged within a hierarchy of four top level categories. Each of the top level categories defines the root of a tree of subcategories, where every child node fine grains the information given by its parent. A document d can be assigned to several categories $c_1, ..., c_p$, and all ancestor categories of a category c_i are assigned to d as well.

Two documents, d_1, d_2, are identified with the same category c if they share both the same top level category c_t and the same most specific category c_s. The test document data sets are constructed in such a way that there is no document d_1 whose most specific category c_s is an ancestor of the most specific category of some other document d_2. The number of categories in our test data varies from three to six. For each category between 50 and 300 documents were drawn randomly from the entire category. The data sets have different sizes and class numbers, and we investigated uniformly as well as non-uniformly distributed category sizes. Table 1 gives an overview of the constructed data sets. The document preprocessing involves parsing, stop word removal according to standard stop word lists, and the application of Porter's stemming algorithm [Porter 1980].

	DS1	DS2	DS3	DS4	DS5	DS6
# categories	3	4	3	5	4	6
# documents	300	400	500	600	700	800
uniformly distributed	no	yes	no	yes	yes	no

Table 1: Overview of the constructed document sets.

3.2 Results

We employed STC and the graph-based clustering algorithms MajorClust and Group Average Link to cluster the constructed document sets. For MajorClust and Group Average Link the underlying document models were varied by computing the edge weights according to the cosine similarity measure using the TFIDF term weighting scheme, and the new suffix-tree-based similarity measures. For the hybrid measure, we used $\varphi_{HYB} = \lambda \cdot \varphi_{ST} + (1 - \lambda) \cdot \varphi_{\cos}$ and found that $\lambda = 0.2$ ($\lambda = 0.5$) works well for MajorClust (Group Average Link). Table 2 and 3 show the achieved F-Measure values [Rijsbergen 1979] for MajorClust and Group Average Link respectively. Note that performance improvements of up to 40% for the new hybrid similarity measure in comparison with the cosine simililarity measure can be observed. The outlined disadvantages of STC are reflected in STC's F-Measure values.

	DS1	DS2	DS3	DS4	DS5	DS6	average
STC	0.55	0.40	0.61	0.33	0.40	0.34	0.44
$\varphi_{\cos}$	0.80	0.60	0.62	0.67	0.66	0.49	0.64
φ_{ST}	0.55	0.46	0.61	0.38	0.45	0.55	0.50
φ_{STF}	0.82	0.70	0.70	0.68	0.76	0.55	0.70
φ_{STFIDF}	0.60	0.60	0.71	0.64	0.78	0.62	0.65
φ_{HYB}	0.84	0.83	0.72	0.74	0.93	0.64	0.78
Improvement in %	5%	38%	16%	10%	40%	31%	22%

Table 2: The table shows the achieved F-Measure values for STC (first row), for MajorClust with the traditional similarity measure $\varphi_{\cos}$ on TFIDF vectors (second row), and for MajorClust with the new suffix-tree-based similarity measures (remaining rows). The improvement refers to φ_{HYB} with respect to $\varphi_{\cos}$.

4 Conclusion

Both the classical vector space model and the suffix tree model play an important role in text-based information retrieval. Interestingly, these models are used in an isolated way: There is neither a comparison between their advantages and disadvantages nor an attempt to combine their different properties. This paper introduces three basic

	DS1	DS2	DS3	DS4	DS5	DS6	average
STC	0.55	0.40	0.61	0.33	0.40	0.34	0.44
$\varphi_{\cos}$	0.82	0.63	0.69	0.55	0.78	0.51	0.64
φ_{ST}	0.55	0.40	0.61	0.33	0.40	0.55	0.47
φ_{STF}	0.83	0.64	0.71	0.57	0.85	0.63	0.71
φ_{STFIDF}	0.84	0.72	0.71	0.64	0.80	0.60	0.72
φ_{HYB}	0.84	0.74	0.74	0.66	0.92	0.70	0.77
Improvement in %	2%	18%	7%	20%	18%	37%	17%

Table 3: The table shows the achieved F-Measure values for STC (first row), for Group Average Link with the traditional similarity measure $\varphi_{\cos}$ on TFIDF vectors (second row), and for Group Average Link with the new suffix-tree-based similarity measures (remaining rows). The improvement refers to φ_{HYB} with respect to $\varphi_{\cos}$.

questions—and provides answers to better understand the relation between both models: It is shown how a suffix tree can be used as document model, and several similarity measures that operate on this representation are proposed. Our experiments clearly indicate that word order preservation in the document model is important for document categorization tasks. The combination of a vector-space-based similarity measure with a suffix-tree-based similarity measure can lead to a significant improvement of clustering performance, regardless of the chosen clustering algorithm. Moreover, we identified properties of the STC algorithm that explain why this approach can not keep up with any other of the investigated clustering settings on the RCV1.

References

[Baeza-Yates and Ribeiro-Neto 1999] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999. ISBN 0-201-39829-X.

[Burrows 1987] J. F. Burrows. Word-patterns and story-shapes. *Literary & ling. comp.*, 1987.

[Finn and Kushmerick 2003] Aidan Finn and Nicholas Kushmerick. Learning to Classify Documents According to Genre. In *IJCAI-03 Workshop on Computational Approaches to Style Analysis and Synthesis*, 2003.

[Meyer zu Eißen and Stein 2004] Sven Meyer zu Eißen and Benno Stein. Genre Classification of Web Pages. In volume 3228 of *LNAI*, pages 256–269, Berlin Heidelberg, 2004. Springer.

[Porter 1980] M. F. Porter. An Algorithm for Suffix Stripping. *Program*, 14(3):130–137, 1980.

[Rijsbergen 1979] C. J. van Rijsbergen. *Information Retrieval*. Buttersworth, London, 1979.

[Rose et al. 2002] T.G. Rose, M. Stevenson, and M. Whitehead. The Reuters Corpus Volume 1. In *Proc. 3rd Int. Conf. on Language Resources and Evaluation*, 2002.

[Roussinov et al. 2001] D. Roussinov, K. Crowston, M. Nilan, B. Kwasnik, J. Cai, and X. Liu. Genre based navigation on the web. In *Proc. 34th Int. Conf. on System Sciences*, 2001.

[Sparck-Jones 1972] Karen Sparck-Jones. A statistical interpretation of term specificity and its application in retrieval. *J. Documentation*, 28:11–21, 1972.

[Stein and Meyer zu Eißen 2004a] Benno Stein and Sven Meyer zu Eißen. Automatische Kategorisierung für Web-basierte Suche. *KI – Special Issue on Adaptive Multimedia Retrieval*, 2004a.

[Stein and Meyer zu Eißen 2004b] Benno Stein and Sven Meyer zu Eißen. Topic Identification: Framework and Application. In *Proc. of I-KNOW 04, Austria*, J.UCS, Graz, Austria, 2004b.

[Stein and Niggemann 1999] Benno Stein and Oliver Niggemann. On the Nature of Structure and its Identification. In *Graph-Theoretic Concepts in CS*, vol. 1665 of *LNCS*. Springer, 1999.

[Zamir and Etzioni 1998] Oren Zamir and Oren Etzioni. Web Document Clustering: A Feasibility Demonstration. In *Proc. of SIGIR'98*, Univ. of Washington, Seattle, USA, 1998.