Similarity-Based Support for Text Reuse in Technical Writing

Axel J. Soto
Faculty of Computer Science
Dalhousie University
Halifax, Canada
soto@cs.dal.ca

Abidalrahman Mohammad
Faculty of Computer Science
Dalhousie University
Halifax, Canada
amohd@cs.dal.ca

Andrew Albert
Innovatia Inc.
St. John, Canada
andrew.albert@innovatia.net

Aminul Islam
Faculty of Computer Science
Dalhousie University
Halifax, Canada
islam@cs.dal.ca

Evangelos Milios
Faculty of Computer Science
Dalhousie University
Halifax, Canada
eem@cs.dal.ca

Michael Doyle
Innovatia Inc.
St. John, Canada
michael.doyle@innovatia.net

ABSTRACT

Technical writing in professional environments, such as user manual authoring for new products, is a task that relies heavily on reuse of content. Therefore, technical content is typically created following a strategy where modular units of text have references to each other. One of the main challenges faced by technical authors is to avoid duplicating existing content, as this adds unnecessary effort, generates undesirable inconsistencies, and dramatically increases maintenance and translation costs. However, there are few computational tools available to support this activity. This paper investigates the use of different similarity methods for the task of identification of reuse opportunities in technical writing. We evaluated our results using existing ground truth as well as feedback from technical authors. Finally, we also propose a tool that combines text similarity algorithms with interactive visualizations to aid authors in understanding differences in a collection of topics and identifying reuse opportunities.

Categories and Subject Descriptors

I.2.7 [**Natural Language Processing**]: Text Analysis; I.7.1 [**Document and Text Editing**]: Document Management

Keywords

Text Similarity, Document Analysis, Authoring Tools and Systems, Visual Text Analytics

1. INTRODUCTION

Technical writing or technical communication is a broad field that can be defined differently depending on the context. In this paper, we broadly define it as the written communication about technical or specialized subjects, such as computational systems, medical procedures, or environmental regulations. This task is performed by companies and government agencies for writing manuals, websites and procedures on a regular basis.

In general, technical writing follows a modular approach for content creation. *Topic-based* authoring is a common approach for the creation of technical documents, where content is structured around topics. Note that we refer to the term 'topic' in a completely different sense here to the one that is commonly used within the text mining community. In this paper, we refer to topics as text fragments that "...are typically about a specific subject, have an identifiable purpose and can stand alone". The Darwin Information Typing Architecture (DITA) [1, 22] is the most common topic-based data model for authoring and publishing.

The main idea behind topic-based authoring (also typically referred as *single-source* writing) is to keep modular units of text that are stored in a single centralized repository, so that this allows an effective reusing of content [11] in different contexts. For example, assume a company has technical manuals, release notes, and an online help web page about a specific device. Very likely, a large part of the content of these documents will be shared. A new edition of a device being released typically leads to changes in the text referring to the updated features of the device. By keeping the writing modular and single-sourced, not only is the text to change easier to pin-point, but it is updated once and reused by any other resources that require that information. This reduces maintenance costs and the chances of inconsistencies. Another useful scenario for highlighting the importance of topic-based writing is language translation, where it is desirable to avoid translating the same text more than once.

One challenge in topic-based writing is identifying opportunities for topic reuse. In other words, given a set of topics, the task is to identify topics or subsets of topics, such as sentences or paragraphs, with similar text content, so that the common text can be referenced and reused, and thus unnecessary repetition is avoided. Note that the problem requires identifying not just the same sequence of text in two topics,

DocEng'15, September 8–11, 2015, Lausanne, Switzerland.

ACM 978-1-4503-3307-8/15/09.
DOI: http://dx.doi.org/10.1145/2682571.2797068.

but also similar enough content that can be slightly modified to allow reuse. In addition, similar or exact content may not always be a good case for reuse, due to the common content being too short, or due to texts that while similar, are likely to mean different things.

Text mining algorithms can help to find topics that are good candidates for reuse. Therefore, one of our goals is to better understand the impact of different similarity approaches for identifying topic reuse candidates. Furthermore, from initial interactions with technical writers we recognized that the actual modification of topics by rewriting or reusing of content is a task that cannot be accomplished in a fully automated manner. For this reason, we are also interested in investigating the design of appropriate tools that allow technical authors to explore the collection of topics, in such a way that they can integrate the results of the similarity methods, and hence make more informative writing decisions.

This paper is organized as follows. The next section reviews related work and methods in this area. In Section 3 we present the different similarity methods we investigated, while Section 4 describes the evaluations we applied on them. The proposal of two interactive visualizations is described in Section 5. Finally, Section 6 summarizes the main contributions of this work and discusses future extensions.

2. RELATED WORK

There are several commercial and non-commercial computational tools that aim at supporting topic-based authoring, such as DITA-optimized XML editors or DITA Content Management Systems. Despite technical writing not being a new area and the DITA standards have been first proposed in 2001, the computer science research community has not looked much into this area. Some exceptions are the work of Paris et al. [20], where a support tool for technical writers of multi-lingual instructions is described, or Baptista's description of the adoption of DITA into a project [2].

While topic-based writing and DITA standards allow and facilitate the reuse of content, technical authors are faced with a difficult task when they need to create or modify technical content, so that they can find similar topics, and hence make the most out of single-sourced type of writing. However, to the best of our knowledge, we have found no research paper that looks into this very concrete problem of how to identify reuse opportunities more easily in the context of technical writing.

Our hypothesis is that text similarity algorithms can provide a powerful basis for detecting potential reuse cases among topics. Many types of text similarity algorithms have been proposed in the literature. An organization of these algorithms can be broadly characterized depending on whether word order is considered or not, whether merely syntactic similarity (word matching) or semantic similarity (matching of words that convey similar meanings) is captured, and whether real-based similarity or binary-based near-duplicate identification is performed [17, 18, 19]. While not in the context of technical writing, some studies have benchmarked and proposed several similarity methods for paraphrasing and plagiarism detection in social media, news and Wikipedia, such as [3], [23], and [26].

Given the lack of studies in the domain of technical text reuse, our goal is to investigate the use of different similarity methods that can be categorized differently in the taxonomy we just described. Therefore, in this paper we experimented with Cosine similarity [18], Longest Common Subsequence [6], Google TriGram similarity method [15] and Locality Sensitive Hashing [24]. Cosine similarity is one of the most basic, yet popular, methods that has been used in very different domains. It captures syntacting matching of words assuming a bag-of-words model (i.e. without considering word order). In Longest Common Subsequence the order of words is important, as the algorithm finds the longest ordered sequence of matching words between two texts. This method has been applied for plagiarism detection in different works [3, 10, 7]. Google Trigram Method is a semantic similarity algorithm that has been proven to be the state-of-the-art in capturing the semantic meaning between texts. It uses a corpus-based approach to capture relatedness between words, which has shown to be superior to many knowledge-based methods [15]. Finally, Locality-Sensitive Hashing represents a family of methods for detecting near-duplicates in very large corpora efficiently. Some applications on finding duplicate content in the web include [12, 21]. In the next section we will describe these similarity algorithms in more detail.

Outside the technical writing domain, several research papers have investigated the modeling and identification of text reuse along time. Researchers have studied ways in which text is copied from one literary work to another, such as in ancient Greek texts [5], newspapers [25] and in the web [21]. It is worth noting that a common aspect in all these papers is the use of visualizations to reflect the findings and allow further understanding of how text components are replicated. Additionally, the work by Janicke et al. [16] presented various interesting visualization strategies to understand how different versions of a document or even different documents share commonalities. Another related domain that has matured greatly both in reuse strategies and in the use of visual aids is that of software. For instance, the work by Druzinski et al. [8] presents a framework, based on the concept of variant analysis, that supports visualizing the commonalities as well as variations amongst software components.

A second hypothesis we consider in this paper is that the proper deployment of any computational tool for topic reuse has to consider the author in the loop, so that the findings of the algorithms can be verifiable and applicable by non text mining experts. The presence of visual strategies and metaphors in most of the content reuse papers that we have reviewed support this hypothesis. However, as opposed to most of these works our focus is not on visualizing existing reuses of content, but rather on using the visualization as a tool to enable an interactive data exploration that facilitates the identification of reuse opportunities.

3. TEXT SIMILARITY METHODS

In this section we describe the four similarity methods that we have experimented with in this work. We chose algorithms of different characteristics so that we could better understand their benefits and limitations in this context.

3.1 Cosine Similarity

Cosine similarity is one of the most popular similarity algorithms that have been applied to text. It measures the degree of similarity of two documents as the correlation be-

tween their corresponding vector representations, which can be quantified as the cosine of their angle. Given two documents $\vec{d_1}$ and $\vec{d_2}$, their cosine similarity is:

$$COS(\vec{d_1}, \vec{d_2}) = \frac{\vec{d_1}.\vec{d_2}}{\| \vec{d_1} \| \| \vec{d_2} \|}.$$

Despite its simplicity and the fact that it ignores the relative order of the words in the document, it offers a competitive baseline for text similarity [18].

3.2 Longest Common Subsequence

Longest Common Subsequence (LCS) is another widely employed technique to measure similarity between texts. It measures the total length of the longest matching substrings in both texts, where these substrings are allowed to be non-contiguous as long as they appear in the same order [6, 14]. While the original algorithm was applied to find substrings of characters, a natural extension is to consider it for words, i.e. the longest common substring has to be composed by a sequence of full words only. The final similarity score can be obtained by dividing the number of words of the longest common subsequence by the length in words of the shortest document under comparison.

3.3 Google Tri-gram Similarity

The Google Tri-gram similarity method (GTM) is an unsupervised corpus-based approach for measuring semantic relatedness between text. GTM uses unigrams and trigrams from the Google Web 1T N-gram corpus[1] to compute the relatedness between words [15], and also extends this concept to quantify the relatedness between text documents. The Google Web 1T N-gram corpus counts the frequency of English word n-grams (unigrams to 5-grams) calculated over one trillion words of web page texts collected by Google in 2006.

The relatedness of two words is computed by considering the trigrams that start and end with the given pair of words, and normalized by their mean frequency using the unigram frequency of each of the words as well as the most frequent unigram in the corpus, as shown in Equation 1. In this equation $C(\omega)$ stands for the frequency of the word ω, $\mu_T(\omega_1, \omega_2)$ is the mean frequency of the trigrams that either start with ω_1 and end with ω_2, or start with ω_2 and end with ω_1, and $C_{\max}$ is the maximum frequency among all unigrams.

$$\mathrm{GTM}(\omega_1, \omega_2) = \begin{cases} \dfrac{\log \frac{\mu_T(\omega_1,\omega_2) C_{\max}^2}{C(\omega_1)C(\omega_2)\min(C(\omega_1)C(\omega_2))}}{-2 \times \log \frac{\min(C(\omega_1),C(\omega_2))}{C_{\max}}} \\ \qquad \text{if } \log \frac{\mu_T(\omega_1,\omega_2) C_{\max}^2}{C(\omega_1)C(\omega_2)\min(C(\omega_1)C(\omega_2))} > 1 \\ \dfrac{\log 1.01}{-2 \times \log \frac{\min(C(\omega_1),C(\omega_2))}{C_{\max}}} \\ \qquad \text{if } \log \frac{\mu_T(\omega_1,\omega_2) C_{\max}^2}{C(\omega_1)C(\omega_2)\min(C(\omega_1)C(\omega_2))} \leq 1 \\ 0 \quad \text{if } \mu_T(\omega_1, \omega_2) = 0 \end{cases} \tag{1}$$

GTM computes a score between 0 and 1 to indicate the relatedness between two texts based on the relatedness of the words within the texts. For given texts P with m words (i.e., $P = \{p_1, p_2, \cdots, p_m\}$) and R with n words (i.e., $R = \{r_1, r_2, \cdots, r_n\}$), where $m \leq n$, first all the common words (the number of common words is δ) are removed, and then a matrix is built, where each entry $a_{ij} \leftarrow GTM(p_i, r_j)$ is the relatedness between words p_i and r_j taken from P and R, respectively.

$$M = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1(n-\delta)} \\ a_{21} & a_{22} & \cdots & a_{2(n-\delta)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{(m-\delta)1} & a_{(m-\delta)2} & \cdots & a_{(m-\delta)(n-\delta)} \end{pmatrix}$$

From each row $M_{i:} = \{a_{i1} \cdots a_{i(n-\delta)}\}$ in the matrix, the significant elements $A_i = \{a_{ij} | a_{ij} > \mu(M_{i:}) + \sigma(M_{i:})\}$ are selected, where $\mu(M_{i:})$ and $\sigma(M_{i:})$ are the mean and standard deviation of row i. The summation of the means of all the $m - \delta$ rows is $\sum_{i=1}^{m-\delta} \mu(A_i)$. Then, we can compute the document relatedness using the following equation:

$$Rel(P, R) = \frac{(\delta + \sum_{i=1}^{m-\delta} \mu(A_i)) \times (m + n)}{2mn}$$

GTM similarity can be computed online[2].

3.4 Similarity based on Locality-Sensitive Hashing

The last similarity method that we applied relies on a general framework for computing similarity functions known as Locality-Sensitive Hashing (LSH), which was first proposed by Gionis et al. [9]. The main goal of LSH is to avoid the combinatorial comparison of all pairs of instances to find those that are near-duplicates. A good introduction to the topic can be found elsewhere [24, 17].

LSH-based methods typically rely on creating low-dimensional signatures of data instances so that the similarity of the signatures approximates that of the original data instances. A key characteristic of the signature creation process is that it is computationally inexpensive in comparison to dimensionality reduction methods, such as multi-dimensional scaling or methods based on singular-value decomposition. In this paper, we make use of min-hashing [4], which aims at creating signatures that approximate Jaccard distance in the context of sparse data representations.

Once the signatures are generated, they are divided in pieces of equal size called "bands". Then, the idea of the approach is to hash the bands of each signature into buckets, so that bands of different signatures that are hashed into the same buckets are likely to correspond to similar data instances.

We applied this same principle to allow the identification of near-duplicate documents in a time that grows linearly with the data. Note that this is a non-deterministic process (dependent on the hash functions) that can generate both false positives, i.e. collisions of signature bands that occur by chance and not because the data instances are similar, as well as false negatives, i.e. signatures of near-duplicates being different and hence not colliding into the same buckets.

For our implementation, we extracted bag of word trigrams as our vector representation. This generates a high-dimensional and sparse representation that is suitable for the application of LSH, whilst simultaneously being able to capture the occurrence of words and their relative order. We chose signatures with a dimensionality equal to 100. We set

[1] `https://catalog.ldc.upenn.edu/LDC2006T13`

[2] `http://ares.research.cs.dal.ca/gtm/`

the number of bands to 50 (i.e. bands of size 2), so that we get more chances for near-duplicates to have similar bands. The similarity score we applied is proportional to the number of times the two signatures bands collide into the same bucket. We also give it a boost to the score when the units that collide are also contiguous in the topic.

4. AUTOMATIC IDENTIFICATION OF REUSE CASES

Our first attempts on the application of the similarity algorithms were carried out by using the whole text of both topics under comparison. However, this approach did not succeed in identifying good reuse candidates, as it failed to provide a high similarity score when only parts of the content were shared. Therefore, this led us to split our topics into smaller units, so that we can compare the text of these units, and hence capture these finer-grain similarities within the topics.

One major challenge we have with LCS, COS and GTM is that the number of comparisons grows quadratically with the number of units to be compared. This means that in some cases, in order to keep the methods within a desirable running time, it may be necessary to constrain the number of topics to be considered in the dataset. This problem gets aggravated by using a finer granularity as our units of reuse, since this further increases the number of comparisons. This was one of our main motivations for incorporating LSH as part of our analysis.

In this section we describe the datasets, the corresponding preprocessing, and the different experiments that we carried out. Due to the different nature of LSH with respect to the other similarity methods, we divided our evaluation by presenting first the results on LCS, COS and GTM, while results using LSH are shown afterwards. We conclude this section discussing another alternative evaluation.

4.1 Datasets and Preprocessing

We conducted our experiments on four different datasets. The first three datasets are books named *CORDAP Content Developer*, *CORDAP Product Owner*, and *CORDAP Reviewer*, which were made available online[3]. The fourth dataset is a proprietary book that we will refer to it as *PropA*. The reason for including this fourth dataset is that it contains a larger number of topics than the first three ones, and hence some computational challenges can be identified due to the large number of comparisons required.

Prior to the similarity computation, books were preprocessed as follows. The DITA topics were parsed so that the text can be extracted from the XML structure. This extracted text is tokenized to allow the application of the different similarity algorithms. In addition to the text extraction, each topic is split into smaller chunks of text, which are considered as our textual units for comparison. These textual units are the ones to be compared in a pairwise manner using the different similarity methods described in Section 3. Thus, when comparing a pair of topics, the actual topic similarity is an aggregate of the pairwise unit similarities. For this aggregate we used the maximum among all pairwise unit similarities between the two topics.

We experimented with two different granularities for the textual units: the *coarse* one, which consists of splitting the topics using the first level of element tags of the hierarchical DITA topic structure[4], and the *fine* granularity, which consists of splitting the content of topics according to the lowest level of the DITA element tags.

[3] `http://web.cs.dal.ca/~soto/topicreuse.html`

4.2 Evaluation of Similarity Algorithms

We applied different evaluation methods to our similarity algorithms. We first evaluated the results of our similarity algorithms by comparing them to existing reuse references that are present in the topics of our datasets. These references point to a specific element in another topic, where this element can encompass from as little as a single sentence to as much as the whole topic. In the DITA terminology, these type of references are called *conref*, and we will consider them as our ground-truth of a reuse between the two topics. LCS, COS and GTM are evaluated first, while the evaluation on LSH using different granularities is done afterwards.

The idea of the evaluation is that after calculating the pairwise comparison of all topics and ranking them in descending order of similarity, the ground-truth references should be among the top-ranked pairs. This evaluation can be accomplished by borrowing metrics from information retrieval [18]. We used precision at n ($P@n$), which is the percentage of true positives among the top n most similar pairs. This allows us to know how accurate the method is when suggesting a low number of candidates for reuse. Precision at 3, 4, and 5 for three different datasets can be examined in Tables 1–3. Unless otherwise indicated, we used the coarse granularity for our comparisons.

Table 1: Precision at 3, 4 and 5 for three similarity methods using CORDAP Content Developer

	LCS	COS	GTM
P@3	1.00	1.00	1.00
P@4	1.00	1.00	1.00
P@5	1.00	1.00	0.80

Table 2: Precision at 3, 4 and 5 for three similarity methods using CORDAP Product Owner

	LCS	COS	GTM
P@3	1.00	1.00	1.00
P@4	0.75	1.00	1.00
P@5	0.80	1.00	1.00

Table 3: Precision at 3, 4 and 5 for three similarity methods using CORDAP Reviewer

	LCS	COS	GTM
P@3	0.67	0.67	0.33
P@4	0.75	0.50	0.25
P@5	0.60	0.40	0.40

In order to evaluate the accuracy of the methods in detecting the whole number of reuse cases, we also report the

[4] A DITA topic can be thought as an XML document where content is contained in hierarchical element tags

true positive rate in terms of all ranked pairs. These curves can be found in Figure 1 using one panel for each dataset. The performance of each similarity method can be analyzed by the area under the curve (AUC), i.e. the larger this area is, the higher the true positives are ranked, and hence the better the method is.

As an analysis of these results, we can see in Tables 1–3 that the similarity methods are fairly accurate in detecting reuse cases when a small number of candidates are presented. The true positive rate curves of Figure 1 show that LCS achieves the best performance as far as detecting existing reuse cases is concerned. However, to better interpret these results, it is important to understand our ground truth. These instances of text reuse are not necessarily comprehensive in the sense that not every pair of topics that could have been reused is annotated as such. Section 5.1 shows some cases that highlight this scenario. Also the textual unit we considered for our comparisons is not necessarily the same as the one used in the ground-truth, and hence existing reuse instances that are smaller to our units are not likely to be selected among the top-ranked pairs. The next subsection further analyzes this granularity issue in more detail.

4.2.1 Results with Locality-Sensitive Hashing (LSH)

We recall that LSH allows identifying near-duplicate text using an algorithm with a time complexity that grows linearly with the number of textual units to be compared. Yet this is at the expense of a higher likelihood of failing to detect similar instances that are not identical, i.e. false negatives, and even getting some poor candidates for reuse, i.e. false positives.

A comparison between the best performing algorithm, LCS, and LSH is shown in Figure 2. When we used the coarse granularity, results show that LSH is able to find 70% of the reuse cases, while the remaining reuse cases, which are likely to reference finer-grain elements, go completely unnoticed to LSH. However, when the finer granularity is used, all the reuse instances are found at a rate similar to LCS. Yet, the time needed to find these instances with LSH is several orders of magnitude lower, from minutes to seconds.

The full strength of LSH is highlighted when a larger set of topics is used. In this case we used the dataset PropA, which has around 1,400,000 pairs of coarse units and around 6,000,000 pairs of fine units to be compared. This dataset is interesting due to its size and to the fact that its topics have a large number of reuses at very different granularities. Results in Figure 3 show that LSH can detect around 75% (when using coarse units) and 82% (when using fine units) of the existing reuse instances using far less attempts than the other methods, but then the true positive rate diminishes considerably. Clearly, the cost of the pairwise comparison using the finer units is prohibitive for LCS, COS and GTM, considering that running GTM using the coarse units took more than a day, whereas running LSH took less than 5 minutes.

These results should be also taken with care as merely optimizing these curves considering our current ground-truth would imply in disregarding semantically similar instances that can also represent important reuse opportunities. Also, while a finer granularity analysis is important in some cases, this unavoidably generates more false positive instances. As discussed in the next section, small units of text are not always good candidates for reuse as they may generate more overhead that outweighs its actual benefits.

5. INTERACTIVE ANALYSIS OF REUSE CASES

This research work also included meetings and surveys with domain experts and technical authors. One key point that was drawn from these interactions is that no matter how good a similarity algorithm can be, authors feel reluctant to trust a fully-automatic method that can do a "search & replace" on similar topics. Authors commented that it is important for them to manually read and assess whether similar text is supposed to convey the same idea, or whether texts are coincidentally similar but likely to evolve differently in the future.

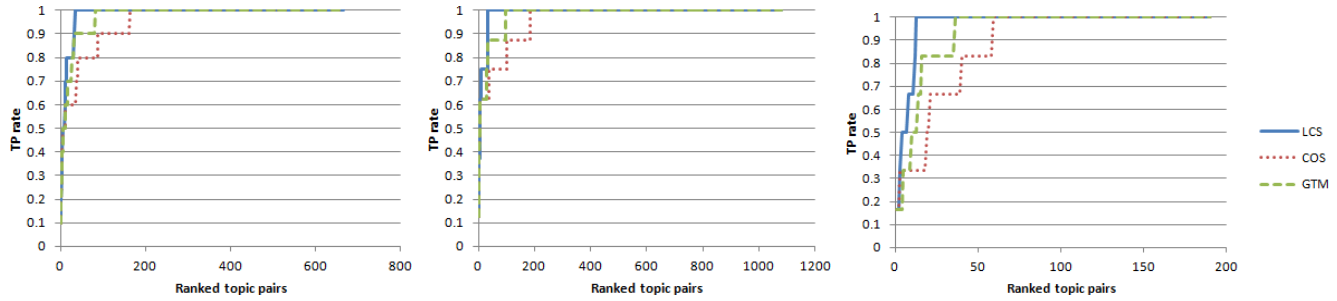
In this section, we first present a small study that aimed at understanding how authors evaluate whether an opportunity for reuse exist between a pair of presented topics. Then, we describe and assess two different interactive tools aimed at supporting technical authors in their work.

5.1 Presentation of highly-ranked pairs to technical authors

The evaluation so far only considered the accuracy of different algorithms in identifying existing reuse references in the text. However, this does not take into account the cases when a pair of topics (or textual units) conveys a similar message with a slightly different wording. Therefore, we took five top ranked pairs of topics in our multiple datasets that were not annotated with an existing reuse case between them, and we presented those to four technical authors. As an illustration, one of the pairs presented is the following (Case 2 in Table 4):

> Topic 1: Adding work packages to an iteration. Iterations can group work packages with the same due dates. At least one iteration must exist. Group all work packages due for the same milestone into one iteration. In My Products, expand the desired product. Click the desired release. Click Iterations. Select the box around the iteration to which the work package belongs. Work packages that are not yet added to an iteration are shown in Unscheduled. All work packages in the selected iteration appear. Drag and drop the work package into the desired iteration. The work package is now part of the selected iteration.
>
> Topic 2: Adding an iteration. Iterations can group your work packages by due dates. Know the due date for your iteration. Use iterations to easily track a group of work packages with similar due dates. In My Products, expand the desired product. Click the desired release. Click Iterations. Click Create new iteration. Enter a name and description for the iteration. Set the start and end dates. Click Save. The iteration is set and ready for work packages. If you need to edit an iteration, click the iteration name. The iteration details window opens for editing.

It is clear that the second sentence in both pairs conveys the same message despite not using the exact same words.



(a) CORDAP Content Developer (b) CORDAP Product Owner (c) CORDAP Reviewer

Figure 1: True positive rate for different datasets. Area under the curves: (a) LCS: 655.50, COS: 631.60, GTM: 648.80; (b) LCS: 1070.25, COS: 1039.75, GTM: 1060.25; (c) LCS: 184.17, COS: 167.17, GTM: 177.33

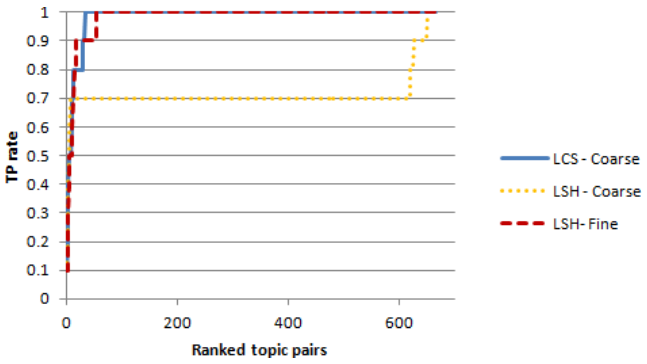


Figure 2: True positive rate using LSH (with a coarse and fine textual unit granularity) on CORDAP Content Developer. LCS performance is the same as the one reported in Figure 1 (a) but included here as a reference.

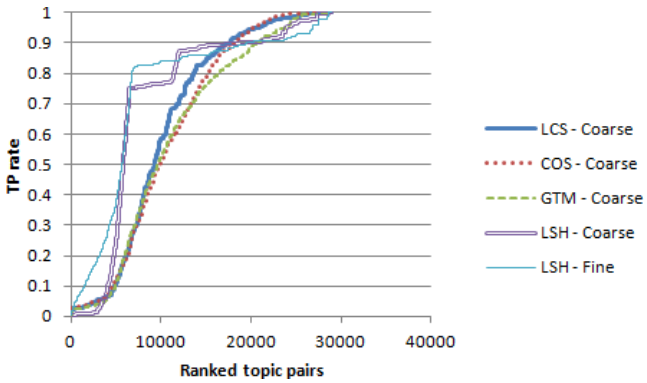


Figure 3: True positive rate using LCS, COS and GTM (coarse units) and LSH (coarse and fine) on PropA.

Overall, upon the presentation of the five selected pairs and the question of whether authors see a reuse opportunity, the responses can be found in Table 4.

Table 4: Author responses upon the presentation of five selected topic pairs and the question of whether authors see a reuse opportunity.

	Author 1	Author 2	Author 3	Author 4
Case 1	Yes	Yes	Yes	Yes
Case 2	Yes	Yes	No	Yes
Case 3	Yes	Yes	Yes	Yes
Case 4	Yes	No	No	No
Case 5	Yes	No	No	Yes

From the authors' feedback on this rather small sample of topic pairs, we could draw some preliminary conclusions. First, whether a case should be reused or not seems to be author-dependent. Some authors may prefer highly modularized topics, while others consider that if there is little text to be shared, then the overhead of reuse does not overcome the benefits, and hence it is not worth the effort. At least there was always one author who considered that each presented topic pair contains a topic reuse opportunity. Also, authors stated that in some cases some sort of restructuring or rewriting was required, which may affect other topics too. All these facts support our hypothesis that a fully automatic tool may not be appropriate and hence some sort of interactive tool may be necessary to address this problem.

5.2 Overall Topic Similarity

Our first visualization was designed to provide an overview of several topics and their similarity. Ideally, these topics are supposed to be presented to an author for creating or updating one or more related books. The use case we had in mind for this visualization is of an author who is assigned to work on a book where several topics already exist. In the beginning this author would be interested in an overview of the set of topics and how they are clustered (groups of topics related to each other). Alternatively, a writer may want to focus on a specific topic and check what other related topics to this one exist, so as to avoid possible inconsistent text repetition.

In order to support this use case we have developed an interactive tool that incorporates the similarity algorithms described in Section 3 in a visual manner. A screenshot of this tool is shown in Figure 4. The interface contains three main components: a force-based topic similarity graph layout (left), a topic search panel (bottom right) and additional options (top right).

In the topic similarity graph, nodes represent topics and edges connect topics in such a way that the length of edges is inversely related to their similarity. Nodes also encode

the topic type[5] and their text length using different marker types and sizes, respectively. The topic search panel allows ordering topics in different ways as well as searching topics by keyterms. There are additional options that allow modifying the graph view in different ways:

1. Thresholding the topic similarity graph, i.e. removing edges below a certain similarity value. In this way authors can interactively focus on different levels of similarity.

2. Filtering of connections incident to certain topic types. This is known to be a common requirement as authors are interested in finding similarities among specific topic types, e.g. looking for similarities among task-type topics only.

These different ways of filtering edges combined with the force-based layout of the graphs facilitate the exploration of sets with several hundred topics, as those disconnected topics get repelled to the borders decluttering graph connections.

There are several other interactions that can be applied on the graph. One of them is to overlay existing (conref) reuse references, which are added as directed links (starting from the topic that has the reference) and in a different color. These existing reuses help indicate for instance which topics would be affected if a referenced topic changed. Another important interaction is the possibility of identifying the most similar topics with respect to a specific topic (i.e. the *focus* topic or node). In this way all edges are hidden except the ones incident to the focus node. This action can be achieved either by shift-clicking a node in the graph or a row in the topic search result table. One final important option is the possibility to use the "Compare" button to bring up the visualization described in Section 5.3 using the topics connected in the graph (up to the top five most similar topics). Readers can experiment with this online tool[6]. Source code has been made available online[7].

5.3 Multiple Topic Text Comparison

Our second visualization was designed as a complement to the previous one to provide an easy way of comparing the text of one topic against other candidate topics for reuse. While this visualization resembles typical interfaces for the popular diff algorithm [13] a key difference here is that we are interested in commonalities, rather than in differences. Furthermore, text is split into smaller units, as explained in Section 4.1, which are compared combinatorially where the relative order of the paragraphs is not important. A screenshot of this interface is shown in Figure 5.

The interface is organized as follows. At the top of the screen there is one topic, which we call the focus topic. This is the one that was shift-clicked in the previous visualization. At the bottom there are up to five topics, which we call the neighboring topics. These are the most similar topics and they are connected to the focus topic as indicated by the previous graph. When hovering over each paragraph of the focus topic, the most similar paragraph in the neighboring topics are also highlighted. A bar chart on the right of the focus topic allows quick identification of similar paragraphs, which in turn indicate potential topics for reuse. On the top left of the screen, the number of neighboring topics shown at the bottom can be adjusted, and the context graph of this panel shows the subset of the topic graph that is being explored.

The visualization shows different similarities depending on the context in which they are applied, namely: "topic similarity" and "paragraph similarity". The topic similarity is just the maximum paragraph similarity of the topic. The paragraph similarity that is shown below the neighboring topics corresponds to the highlighted paragraph. The paragraph similarity shown in the histogram corresponds to the maximum paragraph similarity between the paragraph aligned with the histogram bar and the most similar paragraph in the neighboring topics. This visualization can be accessed from the previous one after the clicking of the "Compare" button. Source code has been made available online[8].

5.4 Preliminary Feedback on the Tools from Technical Authors

The interactive tools were presented to the previously mentioned four technical authors to obtain a preliminary feedback on the tool. Due to the fact that technical authors were distributed geographically in different cities and countries, interaction was through a web survey and not in real time. Although the similarity algorithm underlying the visual interfaces can be changed interactively, for the author evaluation we restricted ourselves to LCS, as this was the method that performed the most consistently in our experiments.

A negative aspect we noticed was that some authors expressed a rather general concern for the tool, especially with the graph, of being distracting as opposed to a feature that could potentially improve efficiency. One author reported on using command line tools, such as the unix command "grep", to manually find similar instances of text in other topics. Future designs could integrate some of these text-oriented functionalities so that authors can get the best out of visual and non-visual types of interaction. One author reported a lack of a clear understanding of what the graph was representing. We think that this could have been alleviated by providing a real time interaction with the author and providing a basic level of training for the tool. In addition, making the interfaces intuitive and hiding any unnecessary complexity seem to be an important aspect for authors to embrace this type of technology.

We also had some encouraging comments, such as: "*This would be useful for large user docs with many procedures and similar GUIs that you could quickly discover similarities and opportunities to reuse content. Often these topics are redundant but over the course of 100+ pages you may not catch all the opportunities to reuse content unless you compared each topic side by side. An option like this interface would save a lot of time both in discovering reuse opportunities and in updating books for future releases*". Interestingly, this same author, who indicated in three out of the five cases in Section 5.1 that the pairs did not present any reuse opportunity, when presented these same cases using the visual text comparison, i.e. the interface of Figure 5, he or she replied

[5]Topics are typically classified in different types depending on their content e.g. task, reference, concept, etc.

[6]`http://web.cs.dal.ca/~soto/topicreuse.html`

[7]https://github.com/axelsoto/DITA-Topic-Graph

[8]`https://github.com/axelsoto/DITA-one-on-many-comparison`

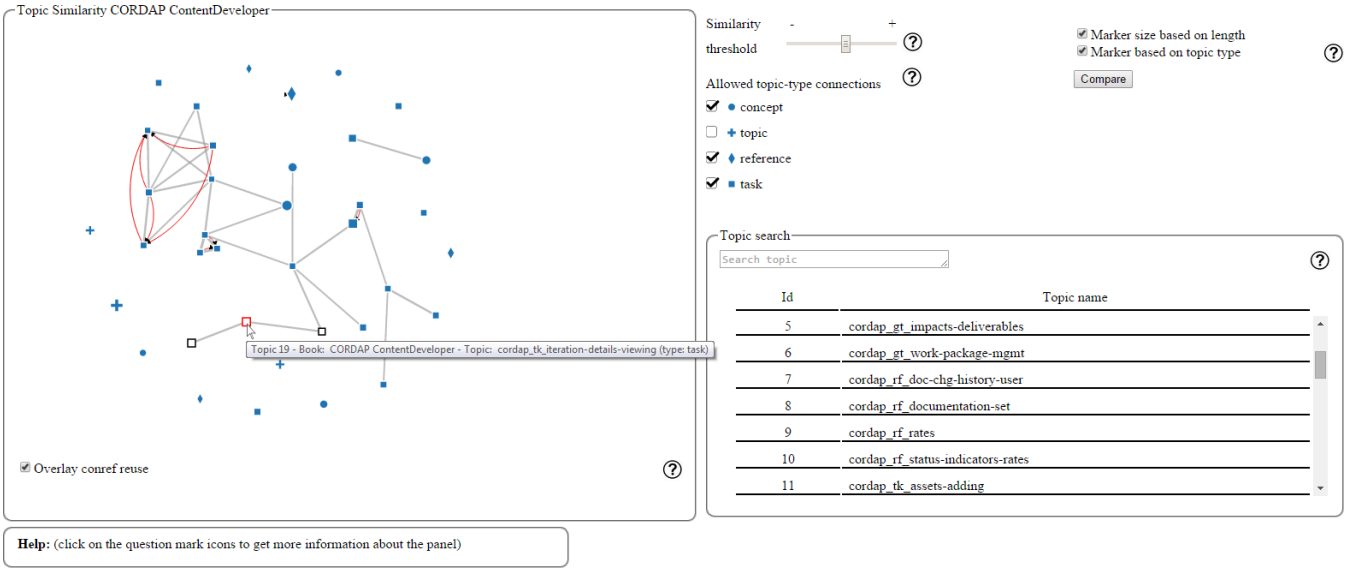


Figure 4: Overall Topic Similarity interactive visualization. Authors can interact with topics and see how they relate to each other.

Figure 5: Multiple topic text comparison. Authors can analyze one topic against several ones and identify pieces of text within the topics that are candidate for reuse. For example, topics 25 and 26 seem to have paragraphs that are candidate for reuse despite not being identical.

that all these cases were a good opportunity for reuse. This would suggest that the visual highlighting was helpful in identifying reuse opportunities more easily.

5.5 Design Guidelines for Prospective New Visualizations

The proposed visualizations constitute an innovative approach towards the analysis of an existing set of technical topics. While these interfaces were implemented at a proof-of-concept level, we envision them as a tool to be integrated into the author's regular writing workflow. In this way, possible topics for reuse should be suggested to the author while writing, as opposed to the scenario presented here where topics similarities are analyzed in retrospective. However, appropriate interfaces should be designed so that the visual aids are suggested in a non-obtrusive manner. In addition, after the identification of similar topics, proper interactions should be facilitated for the generation of the necessary metadata to establish the reuse of a topic with minimal manual effort.

These visualizations would encompass multiple benefits for technical authors, especially for junior authors or those unfamiliar with the topics they may be working on. They would provide ways of identifying at a glance potential topics for reuse, and hence reducing time spent reading and searching for reuse opportunities. In addition, a network visualization of existing topic reuses raises the awareness of the impact on the modification of reused topics.

6. CONCLUSIONS

This paper presented the application of different similarity algorithms in the domain of topic-based technical authoring for text reuse. These similarity algorithms are representative and state-of-the-art approaches within the taxonomy described in Section 2. This study allowed us to compare and to determine the advantages and benefits of different

similarity methods for the task of topic reuse. In addition, we have proposed two interactive visualizations that aim at supporting technical writers in their tasks. The first one allows the interactive exploration of a collection of topics and their similarity, while the second one allows comparing and inspecting commonalities of a topic against other similar topics.

We found LCS to be the best performing algorithm to detect existing (conref) reuse cases. While GTM performed slightly worse for our ground-truth, its capacity of capturing semantic similarity allows finding potential reuse candidates between topic pairs that may be missed by the other methods. LSH finds a high-percentage of near-duplicates in a considerably lower complexity time. Regarding the interactive visualizations, preliminary feedback suggested that this type of technologies could have an important impact on authors' productivity, considering that much of their work is manually intensive, and also that they would not trust on any fully automatic tool.

As future work we plan to combine the different strengths of the similarity algorithms. In this way, a hybrid approach that takes the output of all methods can be obtained. This is in alignment with other approaches presented elsewhere for the tasks of plagiarism and paraphrase identification [3, 26]. One important challenge to address is to obtain first a high-quality ground-truth dataset that goes beyond the use of existing reused topics. This can be achieved from the manual annotation of a large set of topic pairs or from a time-stamped repository, where the evolution of topics at different points in time can be captured. In the latter case, we could determine how topics look before and after reuse. Such a high-quality ground-truth dataset would allow us to experiment with different parameterizations for the similarity methods, without the risk of overfitting to the particular case we considered in this paper.

Another important extension would be to study whether authors could also take a more important role in controlling certain method parameters, such as the acceptable rate of false positives or false negatives. For instance, a lower false positive rate can be obtained by increasing the granularity of the textual units, while a lower false negative rate can be obtained by giving more importance to a semantic-based similarity algorithm, like GTM, or by increasing the number of bands in LSH. Finally, another interesting research direction would be to study the application of our approaches in a real-time context, where users get support as they write, as opposed to an *a posteriori* analysis as presented here.

7. ACKNOWLEDGMENTS

This work was carried out with the aid of grant 2013-LACREG-07 from the International Development Research Centre, Ottawa, Canada and a CALDO-FAPESP grant (Proc. 2013/50380-0). Brazilian researchers are also supported by grants from CNPq (205291/2014-7) and FAPESP (2011/22749-8), and researchers based in Canada by grants from NSERC and a contract from Innovatia Inc.

8. ADDITIONAL AUTHORS

Additional authors: Rosane Minghim (Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, email: `rminghim@icmc.usp.br`) and Maria Cristina Ferreira de Oliveira (Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, email: `cristina@icmc.usp.br`).

9. REFERENCES

[1] Darwin information typing architecture (DITA) version 1.2 - OASIS standard `http://docs.oasis-open.org/dita/v1.2/spec/DITA1.2-spec.html`.

[2] J. Baptista. Pragmatic DITA on a budget. In *Proceedings of the 26th Annual ACM International Conference on Design of Communication*, pages 193–198. ACM, 2008.

[3] D. Bär, T. Zesch, and I. Gurevych. Text reuse detection using a composition of text similarity measures. In *Proceedings of the International Conference on Computational Linguistics*, volume 1, pages 167–184, 2012.

[4] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 327–336. ACM, 1998.

[5] M. Büchler, A. Geßner, T. Eckart, and G. Heyer. Unsupervised detection and visualisation of textual reuse on ancient greek texts. *Journal of the Chicago Colloquium on Digital Humanities and Computer Science*, 1(2), 2010.

[6] F. Y. L. Chin and C. K. Poon. A fast algorithm for computing longest common subsequences of small alphabet size. *Journal of Information Processing*, 13(4):463–469, 1991.

[7] P. Clough and M. Stevenson. Developing a corpus of plagiarised short answers. *Language Resources and Evaluation*, 45(1):5–24, 2011.

[8] S. Duszynski, J. Knodel, and M. Becker. Analyzing the source code of multiple software variants for reuse potential. In *18th Working Conference on Reverse Engineering (WCRE) 2011*, pages 303–307, Oct 2011.

[9] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, pages 518–529, 1999.

[10] B. Gipp and N. Meuschke. Citation pattern matching algorithms for citation-based plagiarism detection: greedy citation tiling, citation chunking and longest common citation sequence. In *Proceedings of the 11th ACM Symposium on Document Engineering*, pages 249–258. ACM, 2011.

[11] N. Harrison. The Darwin information typing architecture (DITA): Applications for globalization. In *Proceedings of International Professional Communication Conference*, pages 115–121. IEEE, 2005.

[12] M. Henzinger. Finding near-duplicate web pages: A large-scale evaluation of algorithms. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 284–291. ACM, 2006.

[13] J. W. Hunt and M. MacIlroy. *An algorithm for differential file comparison*. Bell Laboratories, 1976.

[14] R. W. Irving and C. Fraser. Two algorithms for the longest common subsequence of three (or more) strings. In *Proceedings of the Third Annual*

Symposium on Combinatorial Pattern Matching, CPM '92, pages 214–229. Springer-Verlag, 1992.

[15] A. Islam, E. Milios, and V. Kešelj. Text similarity using google tri-grams. In *Advances in Artificial Intelligence*, pages 312–317. Springer, 2012.

[16] S. Jänicke, A. Geßner, M. Büchler, and G. Scheuermann. Visualizations for text re-use. In *Proceedings of the 5th International Conference on Information Visualization Theory and Applications*, pages 59–70, 2014.

[17] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of massive datasets*. Cambridge University Press, 2014.

[18] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

[19] R. Mihalcea, C. Corley, and C. Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the 21st National Conference on Artificial Intelligence*, volume 6, pages 775–780, 2006.

[20] C. Paris, K. Vander Linden, M. Fischer, A. Hartley, L. Pemberton, R. Power, and D. Scott. A support tool for writing multilingual instructions. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 1398–1404, 1995.

[21] M. Potthast, M. Hagen, M. Völske, and B. Stein. Crowdsourcing interaction logs to understand text reuse from the web. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1212–1221, 2013.

[22] A. Rockley, S. Manning, and C. Cooper. *DITA 101: Fundamentals of DITA for Authors and Managers*. Soc. Technical Communication, FAIRFAX, VA, USA, 2010.

[23] M. Sanchez-Perez, G. Sidorov, and A. Gelbukh. The winning approach to text alignment for text reuse detection at PAN 2014. *Notebook for PAN at CLEF*, pages 1004–1011, 2014.

[24] M. Slaney and M. Casey. Locality-sensitive hashing for finding nearest neighbors. *Signal Processing Magazine, IEEE*, 25(2):128–131, 2008.

[25] D. Smith, R. Cordell, and E. Dillon. Infectious texts: Modeling text reuse in nineteenth-century newspapers. In *2013 IEEE International Conference on Big Data*, pages 86–94, Oct 2013.

[26] N. P. Vo, S. Magnolini, and O. Popescu. Paraphrase identification and semantic similarity in twitter with simple features. In *The 3rd International Workshop on Natural Language Processing for Social Media*, page 10, 2015.