Semantically Weighted Similarity Analysis for XML-based Content Components

Jan Oevermann
University of Bremen &
German Research Center for Artificial Intelligence (DFKI)
Bremen, Germany
jan.oevermann@dfki.de

Christoph Lüth
University of Bremen &
German Research Center for Artificial Intelligence (DFKI)
Bremen, Germany
christoph.lueth@dfki.de

ABSTRACT

Uncontrolled variants and duplicate content are ongoing problems in component content management; they decrease the overall reuse of content components. Similarity analyses can help to clean up existing databases and identify problematic texts, however, the large amount of data and intentional variants in technical texts make this a challenging task.

We tackle this problem by using an efficient cosine similarity algorithm which leverages semantic information from XML-based information models. To verify our approach we built a browser-based prototype which can identify intentional variants by weighting semantic text properties with high performance. The prototype was successfully deployed in an industry project with a large-scale content corpus.

CCS CONCEPTS

• **Information systems → Deduplication**; **Similarity measures**; *Data cleaning*; *Content analysis and feature selection*;

KEYWORDS

Similarity analysis, component content management, semantic weighting, technical documentation, cosine similarity

ACM Reference Format:
Jan Oevermann and Christoph Lüth. 2018. Semantically Weighted Similarity Analysis for XML-based Content Components. In *DocEng '18: ACM Symposium on Document Engineering 2018, August 28–31, 2018, Halifax, NS, Canada.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3209280.3229098

1 INTRODUCTION

Technical documentation is often assembled from multiple content components. These text fragments can be reused and combined across different documents and various authors. Higher reuse rates lead to an efficient publishing process, consistent change management and reduced translation costs. Especially in multi-author environments they allow parallel work and faster text preparation.

Referenced reuse and variant management methods only work reliably if they are *controlled*, which means that reuses and variants can be traced back to their origin. Nearly identical or largely similar content prevents unambiguous retrieval and referencing - this is often the case with *uncontrolled* variants. Examples for variants with high textual similarity but significant semantic differences include two content components describing product variants with the same functionality but different technical data, mirrored tasks (such as assembly/disassembly), or variable product and brand names.

The number of uncontrolled variants is related to the overall consistency of the content and can be influenced by outside factors, such as the number of authors (multi-author environments), the typical lifespan of content and the rigorous use of editorial style guides [2]. Automated data migrations can introduce further variants through splitting documents into content components without cross-checking. However, human errors (*e.g.* copy-pasting and editing instead of constructing a variant) are still most common in real-world data sets.

For complex machinery the amount of technical documentation can grow into hundreds of thousands content components, which is why in most cases XML-based Component Content Management Systems (CCMS) support technical writers in composing, managing and assembling these text fragments. Semantic information models, often implemented as Document Type Definitions (DTDs) or XML Schema Definitions (XSDs), enable authors to structure their texts and tag relevant information with corresponding semantic functions down to the word level. Examples for these kind of information models include *DITA* [6], *PI-Mod* [12] and *S1000D* [10].

Most existing methods for similarity only work on plain text not using semantics from underlying information models and are, therefore, prone to flagging intentional variants as potential duplicates. Other approaches use Natural Language Processing (NLP) to identify semantically important text parts, but depend on complex and language-dependent grammatical models and are, therefore, not suitable for performance-critical processing of data.

In this paper, we want to introduce an efficient method for the similarity analysis of large amounts of XML-based content components which can differentiate between intentional variants and potential duplicates by utilizing semantic weighting of specific text parts. After defining a methodology for text parsing, feature selection and similarity analysis in section 3 we explain our semantic weighting approach with basic examples in section 4. We implemented the method in a browser-based workbench-like tool, which technical writers can use to clean up their content databases. This implementation is further discussed and the performance of the prototype is evaluated with real-world data in section 5.

Research supported by BMBF grant SELFIE, grant no. 01IW16001.

Our contribution beyond the state of the art is the focus on variants in technical documentation and their real-world effects on large-scale content databases in a first approach.

2 PREVIOUS AND RELATED WORK

In earlier previous work [9], we analyzed PDF documents for word clusters to find relations between concepts (in the special application domain of process chains for materials science). However, the work presented here is more based on approaches that utilize cosine similarity as efficient vector space classifier for large quantities of content components [7]. Conclusions about domain-specific text characteristics in technical documentation which can affect similarity measurement in vector space models, such as feature selection, were taken from [8].

Other earlier work was concerned with semantically-based change management [1], where we investigated how changes in a document propagate to the semantics. The approach is fully generic, as long as the syntactic and semantic structure of the document can be expressed in XML. The rules to extract the semantics from the syntax and to propagate changes are all written in a domain-specific rule language in terms of XML elements.

Domain-specific similarity analysis of content components in technical documentation were, for example, discussed in [11] focusing on the application to increase reuse rates by proposing similar text fragments already available in the database (1:*n* comparison).

The applications of text similarity on short segments of text were extensively discussed by [3] and [4]. Although our work shares some of the same challenges (data sparseness, lack of context), they differ in the analyzed text size (sentence/query vs content component) and available structure information (plain text vs XML).

Various measurement methods of text semantic similarity are, for example, evaluated and discussed in [5].

3 METHODOLOGY AND SET-UP

3.1 Overview

The overall methodology consists of three major steps (cf. fig. 1): parsing, feature extraction and similarity analysis. The parsing process extracts and stores text from content components and semantically weighted elements separately.

In the following step all features are selected and counted from the extracted text. Semantically weighted features are multiplied with a weighting quantifier and added to the occurrence count of the unweighted text. The artificially increased occurrence count influences the similarity analysis in predictable ways which can be used to identify uncontrolled variants.

3.2 Parsing

The XML file is parsed based on the parsing information from the user (see section 5.1) and plain text is extracted from the XML elements defined as content components. XML attributes and their values are not evaluated. Special characters are removed. The text is then separated into words via a simple word boundary detection.

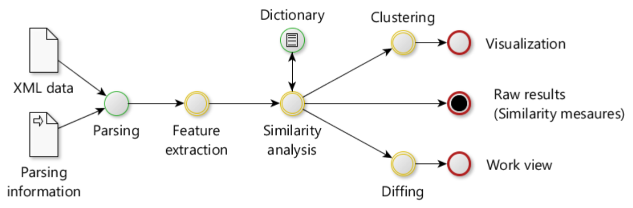


Figure 1: Dataflow for implementation

3.3 Feature selection

As text features we chose a combination of unigrams and bigrams of words ($n = \{1, 2\}$), based on the results from [8] and the necessity to identify weighted differences down to the single-word level (see section 4). Generated *n*-grams are concatenated into objects representing a specific text fragment.

The occurrences of generated features are counted per content component and for the overall corpus. The number of feature occurrences contained in semantically weighted XML elements is multiplied by the weighting quantifier q (we chose $q = 10$) and added to the existing count. The most suitable value for q can depend upon specifics of the underlying information model or the size of content components. A content component is therefore characterized by the distribution of weighted feature occurrences.

For every object a vector is built using all unique features in the corpus as vector components and the object-specific weighted occurrences as values for these components (0 if a feature is not present in an object). Because of the large number of unique features, this process is split into smaller subroutines (we chose a chunk size of $10,000$ vector components, the most suitable value depends on the memory size). The directions of these vectors can now be compared to each other in a vector space model (VSM) via cosine similarity.

3.4 Similarity analysis

Similarity between two content components is symmetrical but must be measured for each possible pair (*n*:*m* comparison). Therefore, the overall number of combinations C for n objects can be calculated as:

$$|C| = \frac{n * (n-1)}{2}$$

For each combination the cosine of vectors $\vec{a}$ and $\vec{b}$ is calculated as similarity measure (where $\vec{a}$ and $\vec{b}$ are two arbitrary vectors resulting from the method described in section 3.3):

$$s = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}$$

Resulting values which are below a previously defined similarity threshold r are discarded ($s < r$, default: $r = 0.9$) for memory efficiency. The value of r directly influences the number of possible uncontrolled variants a technical editor has to examine. In most real-world scenarios a value between 0.85 and 0.95 is chosen.

Since the semantic weighting was achieved by artificially increasing the occurrence count of specific features, the method does not add to the complexity of the algorithm.

4 SEMANTIC WEIGHTING

The semantic weighting presented in this work is based on the underlying structure of a semantic XML-based information model.Therefore, parts of the text which have special semantic meaning (*e.g.* technical data or proper names) are tagged with a corresponding XML element.

Considering two example text excerpts[1] in the PI-Mod information model [12], which have a high similarity but significant differences:

```
<paragraph nodeid="a">This device is designed to work with a
voltage of <inlinedata><si-value><number>110</number>
<unit>V</unit></si-value></inlinedata> only.</paragraph>
```

Example A: First variant with 110 V (excerpt)

```
<paragraph nodeid="b">This device is designed to work with a
voltage of <inlinedata><si-value><number>220</number>
<unit>V</unit></si-value></inlinedata> only.</paragraph>
```

Example B: First variant with 220 V (excerpt)

Applying the methodology described in section 3 we can now calculate the similarity measures with (s_w) and without (s) semantic weighting ($q = 10$, $r = 0.9$) applied to the XML element **`<si-value>`**: $s(A, B) = 0.9021$ vs $s_w(A, B) = 0.4471$.

Without semantic weighting the feature selection ($n = \{1, 2\}$) around the relevant sections produces the following features with corresponding occurrence count (examples A and B):

$$\texttt{[of, 110]} : \mathbf{1}; \texttt{[110, V]} : \mathbf{1}; \texttt{[V, only]} : \mathbf{1}; \texttt{[110]} : \mathbf{1}; \texttt{[V]} : \mathbf{1}$$

$$\texttt{[of, 220]} : \mathbf{1}; \texttt{[220, V]} : \mathbf{1}; \texttt{[V, only]} : \mathbf{1}; \texttt{[220]} : \mathbf{1}; \texttt{[V]} : \mathbf{1}$$

Applying the semantic weighting quantifier ($q = 10$) to text within **`<si-value>`** results in (quantified occurrences are added to unweighted occurrences):

$$\texttt{[of, 110]} : \mathbf{1}; \texttt{[110, V]} : \mathbf{11}; \texttt{[V, only]} : \mathbf{1}; \texttt{[110]} : \mathbf{11}; \texttt{[V]} : \mathbf{11}$$

$$\texttt{[of, 220]} : \mathbf{1}; \texttt{[220, V]} : \mathbf{11}; \texttt{[V, only]} : \mathbf{1}; \texttt{[220]} : \mathbf{11}; \texttt{[V]} : \mathbf{11}$$

While a comparison without semantic weighting could lead to a false positive duplicate flagging ($s(A, B) > r$), the weighted calculation leads to a decreased measure, which indicates non-similar content components in this case.

This behavior is caused by higher occurrence counts for some features which then slightly change the direction of the vectors representing a content component. This ultimately leads to less similarity when calculating the cosine of the vectors.

The method also has advantages when analyzing similarity between alternate phrasings. Considering the following additional examples with slightly different wording:

```
<paragraph nodeid="c">This device works with a voltage of
<inlinedata><si-value><number>110</number> <unit>V</unit>
</si-value></inlinedata> only.</paragraph>
```

Example C: Second variant with 110 V (excerpt)

```
<paragraph nodeid="d">This device works with a voltage of
<inlinedata><si-value><number>220</number> <unit>V</unit>
</si-value></inlinedata> only.</paragraph>
```

Example D: Second variant with 220 V (excerpt)

Calculating similarity between alternate phrasings with the same weighted data yields $s(B, D) = 0.7455$ vs $s_w(B, D) = 0.9844$, while the same phrasing with different weighted data results in $s(C, D) = 0.85$ vs $s_w(C, D) = 0.3605$. This example shows, that the method also increases the similarity measure for content components with the same weighted text property.

5 IMPLEMENTATION

The prototype was implemented as a client-side browser-based application written in JavaScript and is available online.[2] The source code and example files are available in a public repository.[3]

The implementation is based upon a classification framework built for previous work [8]. File handling and basic user interface elements could be reused with little adjustments, as well as VSM-related computing. The processing of similarity analyses and their evaluation were developed for this work and are executed completely client-side in the browser.

5.1 Data flow

The program can process XML files which follow two rules: (1) all content components are contained in elements of the same name directly descendant to the root element (in our examples **`paragraph`**), (2) these elements have an attribute with a unique identifier as value (in our example **`nodeid`**). For semantic weighting all elements can be used which appear inside the elements defined in (1). In our examples we used **`si-value`**. These parameters can be configured on-the-fly via a dynamic form.

The data flow inside the implementation follows the methodology outlined in section 3 (cf. figure 1). To account for the symmetry of similarity we built a dictionary of known combinations which is checked before calculating similarity. After calculating all results[4] the application clusters content components which are similar to each other above the configured threshold and initializes the visualization subroutine.

Table 1: Efficiency tests with real-world data sets

Set	**units** (n)	**comb.** ($\lvert C\rvert$)	$\frac{\text{words}}{\text{unit}}$	**total t** [s]	$\frac{t}{\lvert C\rvert}$ [ms]
A	166	13,695	455.8	0.7	0.052
B	1,600	1,279,200	178.0	243.7	0.191
C	2,501	3,126,250	353.4	650.7	0.208
D	4,101	8,407,050	278.9	2,878.0	0.342

5.2 Efficiency

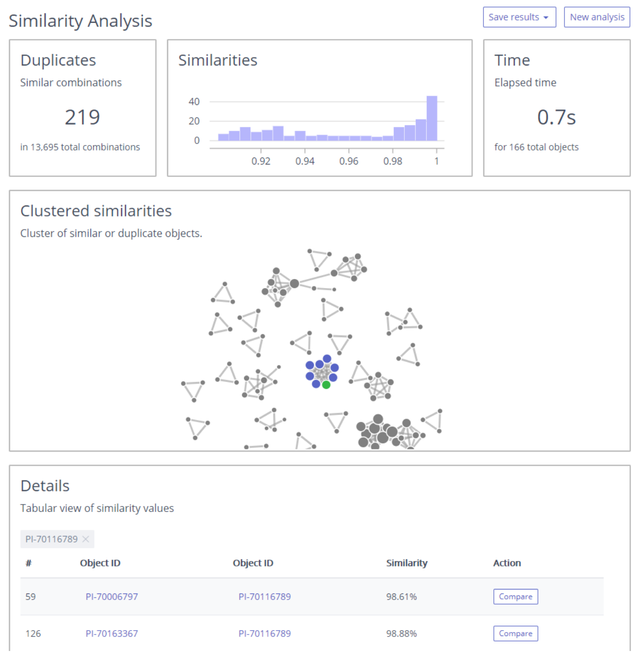
The similarity analysis process is isolated in a web worker thread which separates the calculation-heavy task from the main browser

[1]XML files with the examples used here can be found in the repository mentioned below. Due to the confidential nature of the content we cannot provide the real-world data sets used to evaluate the efficiency of the algorithm (Table 1).

[2]http://semsim.fastclass.de

[3]https://github.com/j-oe/semsim

[4]A result is defined as the combination of two object IDs and their assigned similarity



#	Object ID	Object ID	Similarity	Action
59	PI-70006797	PI-70116789	98.61%	Compare
126	PI-70163367	PI-70116789	98.88%	Compare

Figure 2: Screenshot of the user interface with one cluster selected (cropped, Set A, q = 10 on `<si-value>`, r = 0.9)

thread. In our experiments we could analyze several batches of real-world data on an off-the-shelf laptop [5] with an average computing time per combination of 0.198 ms for data sets between 166 and 4,101 content components of an average word count of about 317 words per objects (see table 1).

5.3 Visualization

We discussed suitable user interfaces with the technical writers who will work with the software. Their preferred way of work is to pick one of the clusters (highlighted in figure 2) of similar content components and then work through them in detail to determine which object of each pair is the leading one (this can either be the newest or the one following the editorial guide).

To achieve this we integrated a standard text diffing algorithm[6] which shows word level differences when the *Compare* button is clicked. Therefore, the user interface (cf. figure 2) provides users with general information about the data set (top), the clusters (middle) and single combinations (bottom).

6 OUTLOOK

As a next step an extensive evaluation of the proposed method has to be carried out which also takes alternative VSM-based similarity measures into account.

For content which is not available in semantic information models, a preprocessing step could tag specific patterns utilizing *Named Entity Recognition* or *Regular Expressions* and transform theses annotations to XML (e.g. for brand names, numbers/units or negations).

[5] Intel i7 2.6 GHz, 16 GB RAM, OS: Windows 10 64bit, Browser: Google Chrome
[6] https://github.com/kpdecker/jsdiff

7 CONCLUSION

In this paper we discussed how an efficient similarity measurement of content components can be achieved while accounting for controlled and uncontrolled variants which are common in the field of technical communication. Our solutions provides clear benefits over existing non-semantic methods and can be validated with a first implementation.

After we defined a methodology and our test set-up, we explained the semantic weighting mechanism with examples of XML-based text. We presented an implementation and practical user interface of the approach and evaluated the efficiency of the algorithm.

As shown in simple examples, a semantic weighting of text properties which are important for variant management in component content management, can possibly reduce false positive duplicate flagging (cf. ex. A & B) and increase similarity of alternate phrasings with the same weighted features (cf. ex. B & D) at the same time .

The question of how suited different information models are for this kind of weighting is subject of further research. While the proposed method is not as sophisticated as other text similarity measures (which take word similarity or external knowledge into account), it is an performance efficient and simple way to improve results of duplicate detection in large corpora of XML-based content components and can serve as a basis for future research.

ACKNOWLEDGMENTS

Special thanks to Wolfgang Ziegler for discussions and support.

REFERENCES

[1] Serge Autexier and Normen Müller. 2010. Semantics-Based Change Impact Analysis for Heterogeneous Collections of Documents. In *Proceedings of 10th ACM Symposium on Document Engineering*, Michael Gormish and Rolf Ingold (Eds.). ACM, Manchester, 97–106. https://doi.org/10.1145/1860559.1860580
[2] Petra Drewer and Wolfgang Ziegler. 2014. *Technische Dokumentation [eng.: Technical Documentation]* (2 ed.). Vogel, Würzburg.
[3] Yuhua Li, David McLean, Zuhair A Bandar, James D O'shea, and Keeley Crockett. 2006. Sentence similarity based on semantic nets and corpus statistics. *IEEE transactions on knowledge and data engineering* 18, 8 (2006), 1138–1150.
[4] Donald Metzler, Susan Dumais, and Christopher Meek. 2007. Similarity measures for short segments of text. In *European Conference on Information Retrieval.* Springer, 16–27.
[5] Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and Knowledge-based Measures of Text Semantic Similarity. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1 (AAAI'06).* AAAI Press, 775–780.
[6] OASIS. 2010. DITA Version 1.2 Specification. Retrieved 2018-05-15 from http://docs.oasis-open.org/dita/v1.2/spec/DITA1.2-spec.html
[7] Jan Oevermann and Wolfgang Ziegler. 2016. Automated Intrinsic Text Classification for Component Content Management Applications in Technical Communication. In *Proceedings of the 2016 ACM Symposium on Document Engineering.* ACM, Vienna, Austria, 95–98. https://doi.org/10.1145/2960811.2967153
[8] Jan Oevermann and Wolfgang Ziegler. 2018. Automated Classification of Content Components in Technical Communication. *Computational Intelligence* (2018), 30–48. https://doi.org/10.1111/coin.12157
[9] Martin Ring, Christoph Lüth, and Ralf Gläbe. 2009. FactBook 2.0. Retrieved 2018-05-15 from http://factbook.informatik.uni-bremen.de/
[10] S1000D. 2017. S1000D Issue 4.2 – International specification for technical publications using a common source database. Retrieved 2018-05-15 from http://public.s1000d.org/Downloads/
[11] Axel J. Soto, Abidalrahman Mohammad, Andrew Albert, Aminul Islam, Evangelos Milios, Michael Doyle, Rosane Minghim, and Maria Cristina Ferreira de Oliveira. 2015. Similarity-Based Support for Text Reuse in Technical Writing. In *Proceedings of the 2015 ACM Symposium on Document Engineering (DocEng '15).* ACM, New York, NY, USA, 97–106. https://doi.org/10.1145/2682571.2797068
[12] Wolfgang Ziegler. 2011. PI-Mod: An information model for plant construction and mechanical engineering (and others). Retrieved 2018-05-15 from http://i4icm.de/en/research-transfer/pi-mod/