Reconstructing Semantic Structures in Technical Documentation with Vector Space Classification

Jan Oevermann
University of Bremen &
Karlsruhe University of Applied Sciences
76133 Karlsruhe, Germany
jan.oevermann@hs-karlsruhe.de

ABSTRACT

With the increasing popularity of component content management systems, a large part of technical documentation in manufacturing and mechanical engineering is written semantically structured in XML-based information models.

Content delivery portals can utilize these information to provide users with advanced retrieval or filtering functions. However, legacy content is often excluded from such granular access due to the lack of semantic structures in archival file formats, as for instance, untagged PDF documents.

In this paper we introduce an approach that uses the classification knowledge present in available content components to reconstruct document structures in text extracted from legacy files. The method leverages transitions in classification confidence for distributed text chunks to detect boundaries between content components of different semantic classes. Classification is done using a modified vector space model for technical documentation. To measure confidence we derive a measure based on properties of cosine similarity in multiclass scenarios. We present first results that show a strong correlation of predicted semantic structures and original document outlines and give proposals for further improvement.

CCS Concepts

•**Information systems** → **Clustering and classification;** *Content analysis and feature selection;*

Keywords

Technical documentation; text segmentation; document structuring; text classification; machine learning

1. INTRODUCTION

Component content management systems (CCMS) enable technical writers to create documentation in a modular fashion using *content components* [11]. These components can be reused across and within different documents [9] which leads to higher consistency, faster preparation and decreased translation cost. In most cases content is stored and edited in XML-based information models such as DITA, DOCBOOK or PI-MOD [3].

Classification models.

The semantic structures of these information models are further enhanced by standardized classification models, such as PI classification [3]. In those models content component are assigned intrinsic (e.g. *information type*) and extrinsic (e.g. *product series*) classes which are defined as taxonomies. Assignment of these classes is usually done manually by technical writers at the time of creation and is based on experience and editorial guidelines. A content component is therefore an instance of one of the possible information classes. This kind of classification allows automated assembling of content components to documents and better retrieval properties for use in online information portals such as content delivery portals.

Content Delivery Portals.

Content delivery portals (CDP) are online information portals which are characterized by distributing modular or aggregated information and providing access for different target groups through content-based retrieval mechanisms [13]. CDPs utilize the classification knowledge of content components to enable users to filter information efficiently with information classes as one of the most important features.

CDPs are not only used to provide better access to modular content but also to include legacy content, such as PDF files for online distribution. However, granular access to these documents is not possible due to their monolithic nature and lack of semantic structure.

Legacy content.

Manufacturers have a legal obligation to keep any technical documentation for several years after putting the product into circulation. As an example, in the European Union the mandatory duty to preserve documentation for machinery is 10 years [1]. Technical documentation is mostly published in PDF file format due to its wide support and good properties for archiving, online distribution and printing (e.g. preserving document appearance).

Therefore, most of the legacy documents are stored as PDF files. Due to the large amount of legacy content most manufacturers have, it is necessary to develop methods for an automated preparation of such data for an enhanced semantic access.

To appear in SEMANTiCS 2016 Posters and Demos track.

2. HYPOTHESIS

If a text, consisting of multiple components with distinct information classes, is divided into several small text chunks, we can classify these text chunks and detect class boundaries at positions where classification confidence drops to a local minimum. This behavior can be used to reconstruct the semantic structures of the original document.

3. METHODOLOGY AND TEST SETUP

3.1 Training data & model

Two independent data sets (A & B) provided from manufacturers were available, consisting of both, classified training data (XML-based content components), as well as unstructured legacy documents (PDF files). Content used for training was manually classified as distinct information types with 9 (A) or 19 (B) classes[1] and structured into 570 (A) respectively 3947 (B) components. For our tests we used PDF files which were untagged[2] but had bookmarks defined (used only for verification of results). The documents had an average page count of 255 (A) respectively 235 (B).

We chose a *bag of n-grams* model with $n = 3$ to represent classes and weighted n-grams according to the TF-IDF-CF method described in [10] to adjust for characteristics of component content management. As a classifier we use *cosine similarity* [6]. While training the model, we also store information about the average word count of content components for later use (referenced as a).

3.2 Implementation

Supervised learning, as well as classification, are implemented as a *Node.js* application written in *JavaScript* which can be adapted to run as web application in a web browser. Models and classification results are stored as *JSON* objects.

3.3 Text extraction

Reconstructing sentences, headings or paragraphs from untagged PDF files in a reliable way is an ongoing problem due to the format's characteristics as a page description language focusing on visual representation, not preserving semantic structure [4]. Especially when working with a large amount of heterogeneous documents, an automatic recognition of, for example paragraphs, can fail.

Therefore we decided on simple text extraction to increase the compatibility across legacy documents. For extracting we used *pdf2json* [12], a Node.js wrapper for *PDF.js* [8].

After text extracting, unnecessary fragments are removed including page numbers, page headers and page footers. All remaining text is combined into a single string while removing hyphenation and punctuation. This string is then tokenized into words by segmentation on word boundaries.

3.4 Segmentation

After text extraction, the obtained set of words is segmented into several text chunks of the same size as the average content component. These text chunks are automatically generated with a predefined offset to each other, for identifying former boundaries between information classes (e.g. between two chapters).

We group the set of extracted words W in arbitrary text chunks $C = \{c_1, ..., c_n\}$, where $c_i \subset W$. The size of chunks is based on the previously collected average word count of content components: a. To distribute text chunks across the document content we choose a natural number r as offset with $r \leq a$. This offset defines the starting position for each chunk. Therefore, a text chunk c_i at position i can be defined as followed (for $i > 1$):

$$c_i = \{W_{(i-1)*r}, W_{(i-1)*r+1}, ..., W_{(i-1)*r+a}\} \quad (1)$$

The total number of chunks $|C|$ generated for a given set of words W dependent on size and offset of chunks can be calculated as followed (with $\lfloor\ \rfloor$ denoting the floor function):

$$|C| = \lfloor \frac{|W| - a}{r} \rfloor \quad (2)$$

A small value for r increases the total number of chunks and therefore the resolution of boundary search, but has also a negative impact on classification performance. The offset can be chosen as a fraction of the average component size.

For our tests we chose an offset value of $r = \lfloor \frac{a}{3} \rfloor$. Offsets smaller than this value don't offer significant advantages in interpreting the results while increasing computation time.

3.5 Classification & confidence scoring

In the following step all text chunks are sequentially classified with cosine similarity [6], where a measure of similarity is calculated for each class in the training model.

To reconstruct the semantic structure of the document, we need to define a score which is high for unambiguous similarity (text chunk is within one information class) and low for having high similarities of two classes at the same time. This occurs when a text chunk spans across two different information classes (see figure 1).

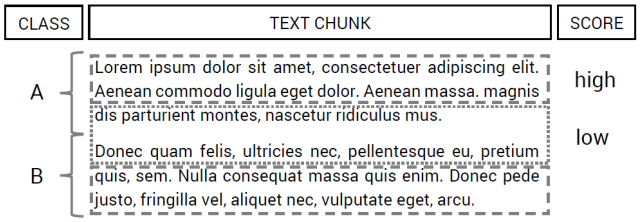

Figure 1: Text chunk classification: Hypothesis

Figure 2: Example for confidence scoring

[1]All class labels are in German and part of company-specific PI classification models.

[2]Tagged PDF "defines a set of rules for representing text in the page content so that characters, words, and text order can be determined reliably" [2]

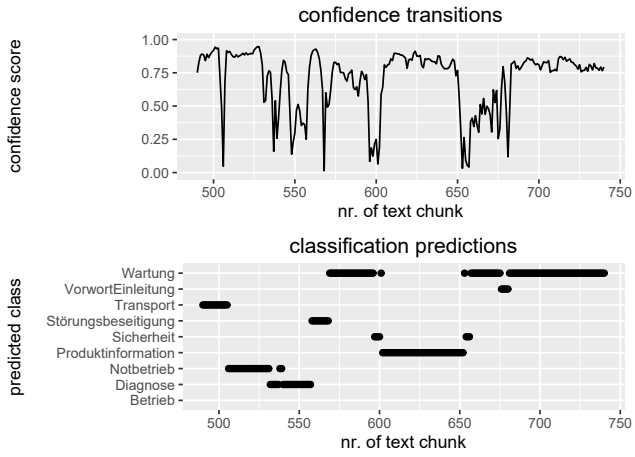

Figure 3: Results confidence transitions (Set A)

There are several methods for comparing per-class classification scores, such as the *softmax function* or the *standard deviation*, however, neither of them suited our need for this distinction. We base our confidence score p on the presence of single outliers (high confidence) or close runner-ups (low confidence) (see figure 2).

Per-class classification scores s_c for n classes c are sorted from high (1) to low (n). p is then calculated as ratio of first to second and first to last classification result of the cosine similarity classifier:

$$p = \frac{s_1 - s_2}{s_1 - s_n} \tag{3}$$

p therefore determines the confidence, that the classification prediction s_1 (highest similarity measure) is the correct prediction for the available set of classes. The confidence score can also be utilized as a measure for quality assurance in automated classification of content components [10].

3.6 Confidence transitions

After calculating confidence scores for all sequential text chunks, we can examine the characteristics of the confidence score around class boundaries (see figure 3). It holds true, that local minima of the confidence curve indicate boundaries between different information classes.

To differentiate noise due to classifier variance from meaningful transitions, it is necessary to set a lower threshold, which must be adjusted to the quality of the classification model. In the given example the threshold for set A could be defined as $p = 0.25$. Therefore, boundaries will only be recognized if the confidence score falls below this value.

4. OBSERVATIONS

When plotting classification results along the axis of generated text chunks, we can see the predicted semantic structure of the documents (see figures 4 and 5). Long spans of the same class are a strong indicator for semantically homogeneous chapters (e.g. *Safety advices* or *Product description*). This observation can be verified by comparing text chunk positions with the chapter structure of the document.

It can also be observed, that classes tend to span multiple text chunks (up to over a hundred) and that small clusters (appearing as dots) seem to mostly have lower confidence values and are likely to be wrongly classified text chunks.

Areas with high uncertainty and different classes for a small stretch of text chunks (annotated as 1–3 in figures 4 and 5) indicate structures with high variance in class characteristics. These can be identified as *Table of contents* (1), *Maintenance plan* (2) and *Document index* (3) for both data sets, all of which are mandatory parts of manuals for machinery [5] which by definition contain elements of different information classes.

A noticeable segment in figure 5 is the span from text chunks 750 – 1500, which shows two classes that seam to appear in an alternating fashion. These can be identified as chapter *Operation* consisting of both classes: *Machine control* and *Machine setting*.

In summary, it can be stated, that the predicted semantic structure matches the chapter structure of the document when looking at spans of text chunks with the same class. The level of details in structure reconstruction depends on the number and distinctiveness of the given set of classes.

5. FURTHER IMPROVEMENTS

As first results show, a general structure of the document can be outlined by defining boundaries at confidence transitions. To prevent small negative spikes in confidence level from becoming false positive separations, previous and following text chunks should be taken into account by weighting the prediction accordingly. As observed, relatively small spans are often indicators for wrong predictions, which could be prevented with a predefined minimum length of class spans.

Furthermore, the threshold for boundary search could be extended with methods that factor in the neighborhood of local minima and the course of the confidence curve.

6. RELATED WORK

In [10] the author presents a vector space model based approach for classifying content components. The methods developed there serve as a basis for this work.

In [7] the authors present a model for text segmentation based on ideas from multilabel classification. However, their approach does classification at token level while our method classifies at segment level.

7. CONCLUSIONS

We have shown that transitions in classification confidence can reliably indicate boundaries between information classes, which can then be used to reconstruct the semantic structure of the document. We introduced a method for generating text chunks and a measure for classification confidence for cosine similarity.

The results from our data sets confirm the hypothesis stated at the beginning of this paper and can be used for further improvement of the method. We plan on implementing the mentioned improvements in future work and to build a prototype application with automated boundary detection.

8. ACKNOWLEDGMENTS

I would like to thank Wolfgang Ziegler (Karlsruhe University of Applied Sciences) and Christoph Lüth (University of Bremen) for discussions and support.

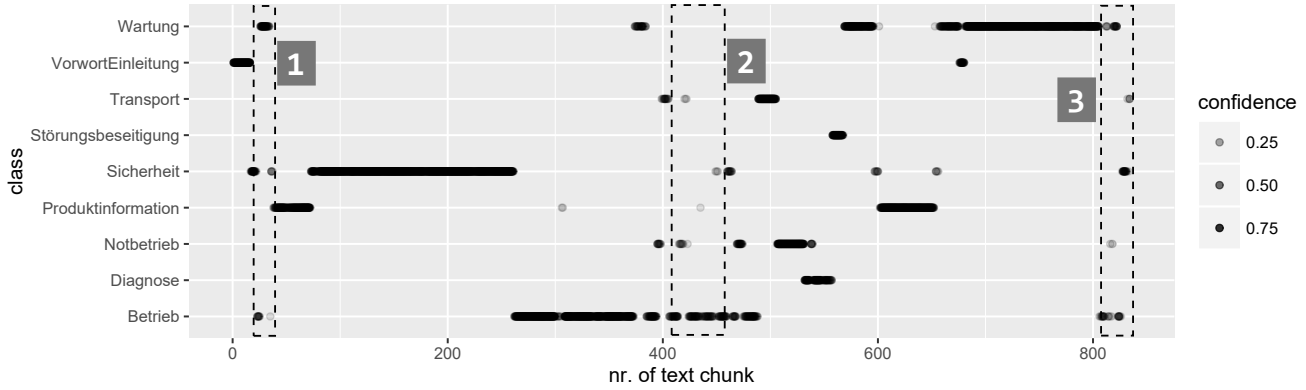

Figure 4: Semantic structure with annotations (Set A)

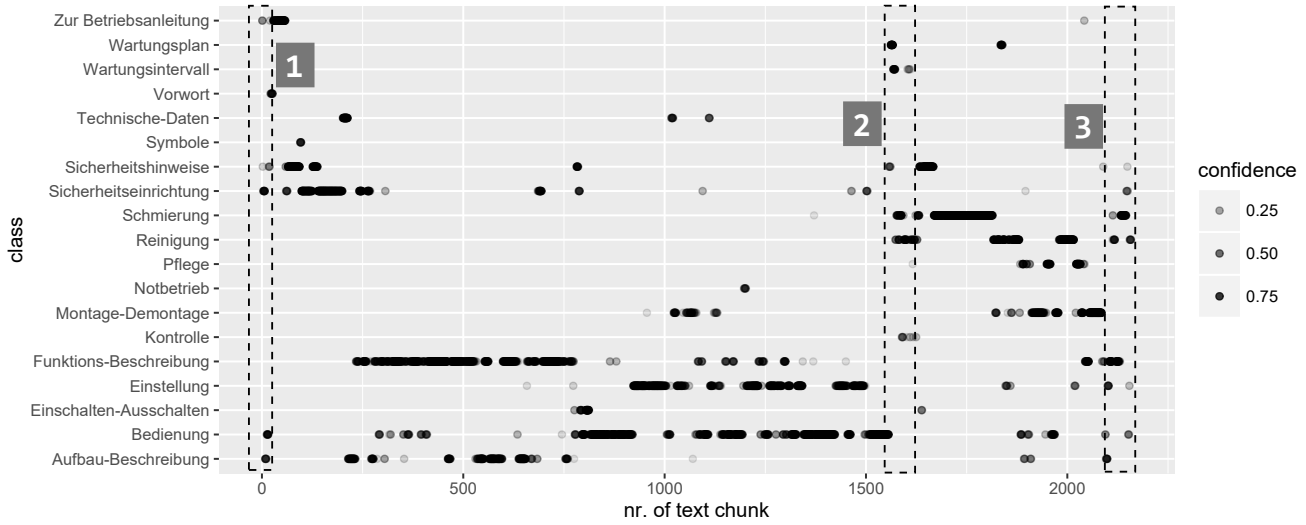

Figure 5: Semantic structure with annotations (Set B)

9. REFERENCES

[1] 2006/42/EC. Machinery directive of the European Parliament and of the Council, 2006.

[2] Adobe Systems, editor. *PDF reference: Adobe portable document format version 1.4*. Addison-Wesley, Boston, 3rd edition, 2001.

[3] P. Drewer and W. Ziegler. *Technische Dokumentation*. Vogel, Würzburg, Germany, 2011.

[4] J. Fang, Z. Tang, and L. Gao. Reflowing-driven paragraph recognition for electronic books in PDF. In *IS&T/SPIE Electronic Imaging*, pages 1–9. International Society for Optics and Photonics, 2011.

[5] IEC 82079-1. Preparation of Instructions for Use - Structuring, Content and Presentation, 2012.

[6] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, Mass, 1999.

[7] R. McDonald, K. Crammer, and F. Pereira. Flexible text segmentation with structured multilabel classification. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 987–994, Stroudsburg, PA, USA, 2005. ACL.

[8] Mozilla Foundation. PDF.js - A general-purpose, web standards-based platform for parsing and rendering PDFs. https://mozilla.github.io/pdf.js, 2016.

[9] C. Oberle and W. Ziegler. Content Intelligence for Content Management Systems. *tcworld e-magazine*, 2012(12), 2012.

[10] J. Oevermann and W. Ziegler. Automated Intrinsic Text Classification for Component Content Management Applications in Technical Communication. In *Proceedings of the 2016 ACM Symposium on Document Engineering*, DocEng '16, New York, NY, USA, 2016. ACM. (to appear)

[11] A. Rockley, P. Kostur, and S. Manning. *Managing Enterprise Content: A Unified Content Strategy*. New Riders, Berkley, CA, 2003.

[12] M. Zhang. A PDF file parser that converts PDF binaries to text based JSON, powered by a fork of PDF.js. https://github.com/modesty/pdf2json, 2016.

[13] W. Ziegler and H. Beier. Content delivery portals: The future of modular content. *tcworld e-magazine*, (02/2015), 2015.