Multi-layered edits for meaningful interpretation of textual differences

Angelo Di Iorio
Department of Computer Science and Engineering
University of Bologna
Bologna, Italy
angelo.diiorio@unibo.it

Gianmarco Spinaci
DHDK, University of Bologna
Bologna, Italy
gianmarco.spinaci@studio.unibo.it

Fabio Vitali
Department of Computer Science and Engineering
University of Bologna
Bologna, Italy
fabio.vitali@unibo.it

ABSTRACT

The way humans and algorithms look at and understand differences between versions and variants of the same text may be very different. While correctness and overall byte length are fundamental aspects of good outputs of diff algorithms, they do not usually provide immediately interesting values for humans trying to make sense of the events that lead from one version to another of a text.

In this paper we propose 3-edit, a layered model to group and organize individual differences (i.e., edits) between document versions in a conceptual value-based scaffolding that provides an easier and more approachable characterization of the modifications occurred to a text document. Through the structural and semantic classification of the individual edits, it becomes possible to differentiate between modifications, so as to show them differently, show only some of them, or emphasize some of them, so that the human mind can more easily identify the types of modifications that matter for its reading purpose.

An algorithm that provides structural and semantic grouping of basic mechanical INS/DEL edits is described as well.

CCS CONCEPTS

• **Information systems → Information systems applications**;
• **Applied computing → Text editing**.

KEYWORDS

Change detection, versioning, diff, textual differences, interpretation of changes

ACM Reference Format:
Angelo Di Iorio, Gianmarco Spinaci, and Fabio Vitali. 2019. Multi-layered edits for meaningful interpretation of textual differences. In *ACM Symposium on Document Engineering 2019 (DocEng '19), September 23–26, 2019, Berlin, Germany*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3342558.3345406

DocEng '19, September 23–26, 2019, Berlin, Germany

ACM ISBN 978-1-4503-6887-2/19/09…$15.00
https://doi.org/10.1145/3342558.3345406

1 INTRODUCTION

The issue of identifying and representing the differences between two documents, or two versions of the same document, can be reasonably called a solved problem. A number of easy to use algorithms and libraries exist that provide fast and accurate differences between text documents.

Most algorithms generate lists of differences as sequences of atomic operations such as INS (the insertion of new content) and DEL (the removal of old content). The simplicity of the model has advantages but also drawbacks. Differences between versions of documents often ends up as huge lists of small individual edits hard to parse and harder to understand by the human eye. Regardless of whether we are making fundamental changes to the meaning of a sentence, fixing grammatical typos in a sentence, or reorganizing the content of a paragraph by splitting it in two, we will only see a long list of INS or DEL operations all mixed up together.

Rather than increasing the set of supported operations, in this paper we propose a layered model to organize and group individual **mechanical** edits into higher level modifications whence their **structural** and **semantic** justification can emerge and be represented appropriately.

The paper also sketches an algorithmic approach to identify these edits and represent higher-level from a long list of low-level ones. The variety of higher-level edits is by no mean complete: additional types can be ideated and described in the future, but they are a first and hopefully useful starting point for a deeper reflection on how humans perceive and understand modifications on text documents.

The rest of the paper is structured as follows: Section 2 presents some related works; Section 3 introduces the 3 edit model, while in Section 4 we describe the algorithm to build the layered model from the output of any traditional diff engine and some ideas for further works.

2 RELATED WORKS

The research community has studied the visualization and interpretation of changes from several angles. In [1][2] the authors proposed a formal model to describe the differences between documents at different levels of abstraction built on top of each other. The model, called UDM, allows designers to describe precisely the piece of content that is affected by a change (at character level, taking into account even the encoding of the character), the type of one single change and the encapsulation of single changes into aggregated (higher level) ones. [3] also proposed a layered model to describe changes and to structure the output of a diff algorithm. The authors

stressed on the distinction between a correct output, that can be used to rebuild the original revision of a document, and a meaningful output, that can be used to clearly explain to the users what happened. The paper also proposed a taxonomy of changes and a diff algorithm able to detect these changes on XML documents. The authors of DocTreeDiff[6] also analysed patterns in editing office documents and extracted rules for their identification in the underlying XML trees and their translation in readable changes for the users.

The need for clear visualizations of documents' changes had been studied many years before. Even in [5] the authors stressed the fact that there is no one-fits-all solution. They showed how users might be interested in verbose visualizations in which all changes are shown together, or concise ones in which some are filtered out; they also investigated the effects of different granularities: from sentence-level differences down to word-level or character-level ones. All these visualizations were combined in a single tool and compared under different conditions.

In [4], the authors worked at the document level: they proposed a mapping between basic changes (for instance, the substitution of a word) and more meaningful edits (for instance, the change of the meaning of that word) and implemented a module for MediaWiki that highlights these changes when diffing two revisions of the same document. Such visualization proved to be more effective for the final readers than the traditional one.

[8] studied how to aggregate low-level code changes into higher-level semantic ones. The goal is to identify related commits that contributed to the implementation of a given functionality. The proposed method works on dependencies between code fragments and managed to identify correctly clusters of changes on some Git projects. The average dimension of machine-produced changes was actually smaller than the original ones but equally understandable for the users.

3 THE 3-EDIT MODEL

In this section we introduce the 3-edit layered model for the characterization of edits. The goal is to give an overview of our approach and provide some pointers to specific details. The full documentation and lists of edits is available at https://github.com/three-level-model-diff/documentation.

Our model allows to group and characterize modifications to documents in a manner that is better understandable and filterable by humans than the mere output of a diff tool or a change tracking tool. The 3-edit layered model does not make assumptions on the tools or processes generating the edits belonging to the lower layer nor the underlying data format of the sources being compared.

Indeed, the fundamental understanding of the 3-edit model is that at the lower layer what is being compared is just plain text. If such plain text happens to include markup elements such as HTML, XML or the like, they are interpreted as plain string sequences at the lower layer, and interpreted as markup in the upper layers. This makes it possible to deal with any diff algorithm, regardless of whether it is text-oriented or tree-oriented. The 3-edit layered model is composed, comprehensibly, of 3 layers.

3.1 Mechanical edits

Mechanical edits are a direct representation of the output of textual diff algorithms. We use JSON as serialisation format but this is not mandatory. Since we want to accept the output of all diff engines, we make little or no assumption on the recognized structures and operations. Basic accepted operations are insertion of strings (INS) and deletion of strings (DEL).

The content of mechanical edits may contain any string of text, markup, and mix of both, and even unbalanced or malformed fragments of markup and text, with no constraint whatsoever. Advanced diff algorithms that provide a more complete and sophisticated set of operations can opt to generate a richer set of basic operations that span across mechanical and structural edits as needed, but the minimum requirement that we place is that text is handled and insertions and deletions are described tersely in the supported JSON format. Additional fields (e.g., timestamp and author) are allowed but not required.

The following structure is an example of two distinct mechanical edits:

```
{
  "id": "mech-00028",
  "op": "DEL",
  "pos": 80,
  "content": "<p id='x45' class='t'>text."
},
{
  "id": "mech-00029",
  "op": "INS",
  "pos": 80,
  "content": "<p class='t' id='x45'>new."
}
```

3.2 Structural edits

Structural edits represent the first grouping and organization of lower level mechanical edits. In this level we perform a first distinction between textual edits and markup edits, so that edits that affect the structure of the document are considered separately from edits that modify the content. Structural edits on markup are directly taken from the list of edit operations listed in [3], while edits on text are specializations of insertions and deletions that we found useful to identify for the needs of the third layer, the semantic edits. Table 1 reports some of the structural operations we identified.

Structural edits provide a justification and a context for one or more mechanical edits, so an important aspect of structural edits is the list of mechanical edits they are composed of. An example of composition is shown below.

Note that structural edits on markup rely on the fact that only whole nodes are affected, while structural edits on text rely on the fact that only fragments entirely composed of text are affected. In real edits, things are rarely so clean. Fortunately, mechanical edits can be separated in chunks with no ill effects, so a first job of the algorithm that builds the structural edits is to divide mixed mechanical edits into separate, clean chunks. In the example, two separate mechanical edits are divided and reorganized so as to make sure to capture the correct structural edits that took place.

Table 1: Some structural operations ordered by priority. The list is partial and does not cover the full model.

Op	*Description*
NOOP	A textual modification that has been captured by the diff engine but does not affect the document in any way (e.g., a modification in the order of the attributes of a node)
WRAP/UNWRAP	The insertion of a fragment of the document inside a new node, child of the current node (e.g., the transformation of a text fragment in bold, by adding a B element in HTML)
JOIN/ SPLIT	The separation of one node into two contiguous siblings. (e.g., the split of a paragraph in two, after the insertion of a carriage return)
REPLACE	A pair of INS and DEL of one or more markup nodes. They must be associated to the same position in the document. Order and temporal contiguity of INS and DEL.
INSERT/ DELETE	The insertion/deletion of one or more whole markup nodes. This is the residual category for INS/DEL operations that contain markup
PUNCTUATION	A subtype of TEXTREPLACE in which only punctuation has been touched
WORDREPLACE	A subtype of TEXTREPLACE in which a single word is affected
WORDCHANGE	A subtype of WORDREPLACE in which the word is not changed but modified
TEXTREPLACE	A pair of INS and DEL of some text content without markup

```
{
  "id": "struct-00017",
  "op": "TEXTREPLACE",
  "by": "Fabio Vitali",
  "timestamp": "2019-03-10T07:26:44",
  "items": [{
    "id": "mech-00031",
    "op": "DEL",
    "pos": 94,
    "content": "Initial text."
  },{
    "id": "mech-00033",
    "op": "INS",
    "pos": 94,
    "content": "New words."
  }]
}
```

3.3 Semantic edits

Semantic edits are operations that a human can immediately provide a justification for in terms of the meaning and impact on the document. In addition to establish **what** happened to the document, the semantic layer aims to provide an explanation of **why** the modification happened. Of course, given the complexity and variety of justifications for modifying a document, this layer is by construction incomplete. Table 2 shows some of the semantic edits we have found interesting to identify.

All edits are by default modifications of meaning, unless a more precise characterization can be reliably assumed. Some of them are subsequently recognized and possibly grouped into some of the other semantic categories.

Consider the FIX edit, defined as the correction of a fragment that could be considered erroneous and is now fixed. A common way to do that is to replace a few words, an action captured as a WORDREPLACE structural edit, which in turn is composed of a pair of mechanical INS/DEL. The following code snippet shows how mechanical, structural and semantic edits are connected in our model.

```
{
  "id": "sem-00002",
  "op": "FIX",
  "old": "Some words is better than others",
  "new": "Some words are better than others",
  "items": {
    "id": "struct-00021",
    "op": "WORDREPLACE",
    "by": "Fabio Vitali",
    "timestamp": "2019-10-10T07:25:23",
    "items": [{
      "id" : "mech-00083",
      "op":"DEL",
      "pos": 215,
      "content": "is"
    },{
      "id" : "mech-00084",
      "op":"INS",
      "pos": 215,
      "content": "are"
    }]
  }
}
```

Semantic edits can be divided into edits that contain exactly one structural edit, in which case the third layer should be intended more as a way to characterize, i.e. to provide a color, to the structural edits, and edits that contain a *multiplicity* of lower level structural edits, in which case they are grouped and the semantic is associated to the group rather than to each individual edit.

Algorithms that can predict reliably which semantic category each edit belongs to are still being developed. This characterization is, inevitably, the result of the intersection between multiple engines and techniques, based on NLP and machine-learning algorithms,

Table 2: A partial list of semantic operations.

Op	*Description*
MEANING	No specific justification has been found for this edit. Default characterization of every structural edit.
FIX	The modification is a correction of a fragment that, in the previous version, could be considered erroneous and is now fixed.
STYLE	The modification is an improvement in the style that does not change radically the meaning of the sentence or of the document.
EDITCHAIN	A series of similar or identical modifications happening throughout the document, e.g. because of a global replace.
EDITWAR	Edit wars[7] started in Wikipedia when editors started changing and restoring the same text due to fundamental divergences in view on the content.
EDITWAKE	An edit wake is a sequence of edits the first of which changes the meaning/structure and the rest fix the document from the grammatical/structural problems introduced by the first

including the intervention of humans to improve and fix erroneous attributions. This is the main direction of our future research as detailed in the next concluding section.

4 MOVING FORWARD: THE 3-EDIT ALGORITHM

In this paper we have introduced 3-diff, a model to group and organize all the low-level edits, generated between different versions of the same document, inside informative categories (i.e., structural and semantic edits) that provide an easier visualization for the human eye. We are currently working on the implementation of the algorithm capable of detecting in an automated way the proposed categories and we are studying newer approaches to (semi-)automatic recognize all the operations described.

The algorithm takes as input the list of all the mechanical edits, in JSON format. It is based on **pattern-matching** and **prioritization**. It starts searching all mechanical edits that match the criteria to be recognized as structural operations. All operations have different priorities. For instance, if two mechanical edits can be labelled as both "PUNCTUATION" or a sequence of "TEXT DELETE" and "TEXT INSERT", they will be categorized as "PUNCTUATION" only, since its priority is higher.

The algorithm look for unmatched mechanical changes and tries to match them to patterns that would categorize them as a structural change. These changes might be just two or even more, in case of combined edits. The patterns are based on position, content, and operation. If there is a match, all changes are removed from the list of mechanical diffs and added instead in the list of structural operations.

As of the semantic operations, the algorithm takes as input the list of structural operations. While some operations are solvable computationally (eg. "Meaning", which is also the residual one), others need very sophisticated techniques. For example, in order to handle a FIX operation the system must understand that the deleted content has a typo. We are experimenting machine-learning algorithms and we achieved quite good results with SVM-based approach, which seem to be the best fit.

The operations that require more than two versions to be recognized, such as EDIT-CHAIN or EDIT-WAKE, require us to extend the list of ML features and to also include temporal information. Our preliminary experiments also suggested us that some human verification will be needed and supervised approaches would work better than unsupervised ones.

REFERENCES

[1] Gioele Barabucci. 2013. Introduction to the Universal Delta Model. In *Proceedings of the 2013 ACM Symposium on Document Engineering (DocEng '13)*. ACM, New York, NY, USA, 47–56. https://doi.org/10.1145/2494266.2494284

[2] Gioele Barabucci. 2018. Diffi: Diff Improved; a Preview. In *Proceedings of the ACM Symposium on Document Engineering 2018 (DocEng '18)*. ACM, New York, NY, USA, Article 38, 4 pages. https://doi.org/10.1145/3209280.3229084

[3] Paolo Ciancarini, Angelo Di Iorio, Carlo Marchetti, Michele Schirinzi, and Fabio Vitali. 2016. Bridging the Gap Between Tracking and Detecting Changes in XML. *Softw. Pract. Exper.* 46, 2 (Feb. 2016), 227–250. https://doi.org/10.1002/spe.2305

[4] Peter Kin-Fong Fong and Robert P. Biuk-Aghai. 2010. What Did They Do? Deriving High-level Edit Histories in Wikis. In *Proceedings of the 6th International Symposium on Wikis and Open Collaboration (WikiSym '10)*. ACM, New York, NY, USA, Article 2, 10 pages. https://doi.org/10.1145/1832772.1832775

[5] Christine M. Neuwirth, Ravinder Chandhok, David S. Kaufer, Paul Erion, James Morris, and Dale Miller. 1992. Flexible Diff-ing in a Collaborative Writing System. In *Proceedings of the 1992 ACM Conference on Computer-supported Cooperative Work (CSCW '92)*. ACM, New York, NY, USA, 147–154. https://doi.org/10.1145/143457.143473

[6] Sebastian Rönnau, Geraint Philipp, and Uwe M. Borghoff. 2009. Efficient Change Control of XML Documents. In *Proceedings of the 9th ACM Symposium on Document Engineering (DocEng '09)*. ACM, New York, NY, USA, 3–12. https://doi.org/10.1145/1600193.1600197

[7] Taha Yasseri, Robert Sumi, András Rung, András Kornai, and János Kertész. 2012. Dynamics of Conflicts in Wikipedia. *PLOS ONE* 7, 6 (06 2012), 1–12. https://doi.org/10.1371/journal.pone.0038869

[8] C. Zhu, Y. Li, J. Rubin, and M. Chechik. 2017. A Dataset for Dynamic Discovery of Semantic Changes in Version Controlled Software Histories. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. 523–526. https://doi.org/10.1109/MSR.2017.49