Automatically Classifying Edit Categories in Wikipedia Revisions

Johannes Daxenberger[†] **and Iryna Gurevych**[†‡]

† Ubiquitous Knowledge Processing Lab
Department of Computer Science, Technische Universität Darmstadt

‡ Information Center for Education
German Institute for Educational Research and Educational Information

http://www.ukp.tu-darmstadt.de

Abstract

In this paper, we analyze a novel set of features for the task of automatic edit category classification. Edit category classification assigns categories such as spelling error correction, paraphrase or vandalism to edits in a document. Our features are based on differences between two versions of a document including meta data, textual and language properties and markup. In a supervised machine learning experiment, we achieve a micro-averaged F1 score of .62 on a corpus of edits from the English Wikipedia. In this corpus, each edit has been multi-labeled according to a 21-category taxonomy. A model trained on the same data achieves state-of-the-art performance on the related task of fluency edit classification. We apply pattern mining to automatically labeled edits in the revision histories of different Wikipedia articles. Our results suggest that high-quality articles show a higher degree of homogeneity with respect to their collaboration patterns as compared to random articles.

1 Introduction

Due to its ever-evolving and collaboratively built content, Wikipedia has been the subject of many NLP studies. While the number of newly created articles in the online encyclopedia declined in the last few years (Suh et al., 2009), the number of edits in existing articles is rather stable.[1] It is reasonable to assume that the latter will not change in the near future. One of the major reasons for the popularity of Wikipedia is its up-to-dateness (Keegan et al., 2013), which in turn requires constant editing activity. Wikipedia's revision history stores all changes made to any page in the encyclopedia in separate revisions. Previous studies have exploited revision history data in tasks such as preposition error correction (Cahill et al., 2013), spelling error correction (Zesch, 2012) or paraphrasing (Max and Wisniewski, 2010). However, they all use different approaches to extract the information needed for their task. Ferschke et al. (2013) outline several applications benefiting from revision history data. They argue for a unified approach to extract and classify edits from revision histories based on a predefined edit category taxonomy.

In this work, we show how the extraction and automatic multi-label classification of any edit in Wikipedia can be handled with a single approach. Therefore, we use the 21-category edit classification taxonomy developed in previous work (Daxenberger and Gurevych, 2012). This taxonomy enables a fine-grained analysis of edit activity in revision histories. We present the results from an automatic classification experiment, based on an annotated corpus of edits in the English Wikipedia. Additional information necessary to reproduce our results, including word lists and training, development and test data, is released online.[2] To the best of our knowledge, this is the first approach allowing to classify each single edit in Wikipedia into one or more of 21 different edit categories using a supervised machine learning

[1] http://stats.wikimedia.org/EN/TablesDatabaseEdits.htm

[2] http://www.ukp.tu-darmstadt.de/data/edit-classification

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 578–589,
Seattle, Washington, USA, 18-21 October 2013.

approach.

We define our task as edit category classification. An *edit* is a coherent, local change which modifies a document and which can be related to certain meta data (e.g. its author, time stamp etc.). In edit category classification, we aim to detect all n edits $e^k_{v-1,v}$ with $0 \leq k < n$ in adjacent versions r_{v-1}, r_v of a document (we refer to the older revision as r_{v-1} and to the newer as r_v) and assign each of them to one or more edit categories. There exist at least two main applications of edit category classification: First, a fine-grained classification of edits in collaboratively created documents such as Wikipedia articles, scientific papers or research proposals, would help us to better understand the collaborative writing process. This includes answers to questions about the kind of contribution of individual authors (Who has added substantial contents?, Who has improved stylistic issues?) and about the kind of collaboration which characterizes different articles (Liu and Ram, 2011). Second, automatic classification of edits generates huge amounts of training data for the above mentioned NLP systems.

Edit category classification is related to the better known task of document pair classification. In *document pair classification*, a pair of documents has to be assigned to one or more categories (e.g. paraphrase/non-paraphrase, plagiarism/non-plagiarism). Here, the document may be a very short text, such as a sentence or a single word. Applications of document pair classification include plagiarism detection (Potthast et al., 2012), paraphrase detection (Madnani et al., 2012) or text similarity detection (Bär et al., 2012). In *edit category classification*, we also have two documents. However, these documents are different versions of the same text. This scenario implies certain characteristics for a well-designed feature set as we will demonstrate in this study.

The main contributions of this paper are: First, we introduce a novel feature set for edit category classification. Second, we evaluate the performance of this feature set on different tasks within a corpus of Wikipedia edits. We propose the new task of edit category classification and show that our model is able to classify edits from a 21-category taxonomy. Furthermore, our model achieves state-of-the-art performance in a fluency edit classification task (Bronner and Monz, 2012). Third, we analyze collaboration patterns based on edit categories on two subsets of Wikipedia articles, namely featured and non-featured articles. We detect correlations between collaboration patterns and high-quality articles. This is demonstrated by the fact that featured articles have a higher degree of homogeneity with respect to their collaboration patterns as compared to random articles.

The rest of this paper is structured as follows. In Section 2, we motivate our experiments based on previous work. Section 3 explains our training data and the features we use for the machine learning experiments. In Section 4, we present and discuss the results of our experiments. We also demonstrate an application of our classifier model in Section 5 by mining frequent collaboration patterns in the revision histories of different articles. Finally, we draw a conclusion in Section 6.

2 Related Work

Wikipedia is a huge data source for generating training data for edit category classification, as all previous versions of each page in the encyclopedia are stored in its revision history. Unsurprisingly, the number of studies extracting certain kinds of Wikipedia edits keeps growing. Most of these use manually defined rules or filters find the right kind of edits. Among the latter, there are NLP applications such as the detection of lexical errors (Nelken and Yamangil, 2008), spelling error correction (Max and Wisniewski, 2010; Zesch, 2012), preposition error correction (Cahill et al., 2013), sentence compression (Nelken and Yamangil, 2008; Yamangil and Nelken, 2008), summarization (Nelken and Yamangil, 2008), simplification (Yatskar et al., 2010; Woodsend and Lapata, 2011), paraphrasing (Max and Wisniewski, 2010; Dutrey et al., 2011), textual entailment (Zanzotto and Pennacchiotti, 2010; Cabrio et al., 2012), information retrieval (Aji et al., 2010; Nunes et al., 2011) and bias detection (Recasens et al., 2013).

Bronner and Monz (2012) define features for the supervised classification of factual and fluency edits. Their features are calculated both on character- and word-level. Furthermore, they use features based on POS tags, named entities, acronyms, and a lan-

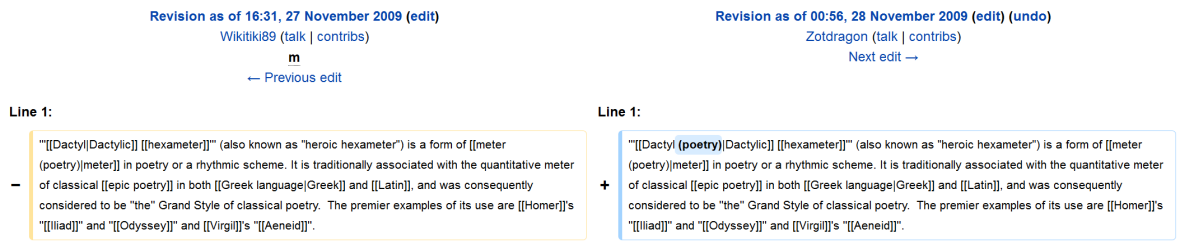

Figure 1: An example edit from WPEC labeled with REFERENCE-M, as displayed by Wikimedia's diff page tool.

guage model (word n-grams). In their experiments, character-level features and named entity features show the highest improvement over the baseline.

Vandalism detection in Wikipedia has mostly been defined as a binary machine learning task, where the goal is to classify a pair of adjacent revisions as vandalized or not-vandalized based on edit category features. In Adler et al. (2011), the authors group these features into meta data (author, comment and time stamp of a revision), reputation (author and article reputation), textual (language independent, i.e. token- and character-based) and language features (language dependent, mostly dictionary-based). They carry out cross-validation experiments on the PAN-WVC-10 corpus (Potthast and Holfeld, 2011). Classifiers based on reputation and text performed best. Adler et al. (2011) use Random Forests as classifier (Breiman, 2001) in their experiments. This classifier was also used in the vandalism detection study of Javanmardi et al. (2011) where it outperformed the classifiers based on Logistic Regression and Naive Bayes.

Different to the approach of Bronner and Monz (2012) and previous vandalism classification studies, we built a model which accounts for multi-labeling and a fine-grained edit category system. Our feature set builds upon existing work while adding a substantial number of new features.

3 Experiments

3.1 Wikipedia Edit Category Corpus

For our experiments, we used the freely available Wikipedia Edit Category Corpus (WPEC) compiled in previous work (Daxenberger and Gurevych, 2012). In this corpus, each pair of adjacent revisions is segmented into one or more edits. This enables an accurate picture of the editing process, as an author may perform several independent edits in the same revision. Furthermore, edits are multi-labeled, i.e. each edit is assigned one or more categories. This is important for a precise description of major edits, e.g. when an entire new paragraph including text, references and markup is added. There are four basic types of edits, namely Insertions, Deletions, Modifications and Relocations. These are calculated via a line-based diff comparison on the source text (including wiki markup). As previously suggested (Daxenberger and Gurevych, 2012), inside modified lines, only the span of text which has actually been changed is marked as edit (either Insertion, Deletion or Modification), not the entire line. We extracted the data which is not contained in WPEC (meta data and plain text of r_{v-1} and r_v) using the Java Wikipedia Library (JWPL) with the Revision Toolkit (Ferschke et al., 2011).

In Daxenberger and Gurevych (2012), we divide the 21-category taxonomy into text-base (meaning-changing edits), surface (non meaning-changing edits) and Wikipedia policy (VANDALISM and REVERT) edits. Among the text-base edits, we include categories for templates, references (internal and external links), files and information, each of which is further divided into an insertion (I), deletion (D) and modification (M) category. Surface edits consist of paraphrases, spelling and grammar corrections, relocations and markup edits. The latter category contains all edits which affect markup elements that are not covered by any of the other categories and is divided into insertions, deletions and modifications. This includes, for example, apostrophes in '''bold text'''. We also suggested an OTHER category, which is intended for edits which cannot be labeled due to segmentation errors. Figure 1 shows an example edit from WPEC, labeled with the REFERENCE-

	Feature	Value	Explanation
Meta Data	Author group	user	Wikimedia user group of the author
	Author is registered*	true	Author is registered (otherwise: IP user)
	Same author*	false	Authors of r_v and r_{v-1} are the same
	Comment length*	0	Number of characters in the comment
	Vulgarism in comment	false	Comment contains a word from in the vulgarism word list
	Comment is auto-generated	false	Entire comment has been auto-generated
	Auto-generated comment ratio	0	Auto-generated part of comment divided by length of the comment
	Incorrect comment ratio	0	Out-of-dictionary word count divided by word count in the comment
	Comment n-grams[1]	—	Presence or absence of token n-grams in the comment
	Is revert*	false	Comment contains a word from in the revert word list
	Is minor	false	Revision has been marked as minor change
	Time difference*	505	Time difference between r_{v-1} and r_v (in minutes)
	Number of edits	1	Absolute number of edits in the (r_{v-1}, r_v)-pair
Textual	Diff capitals*	0	Difference in the number of capitals
	Diff digits*	0	Difference in the number of digits
	Diff special characters*	2	Difference in the number of non-alphanumeric characters
	Diff whitespace characters	1	Difference in the number of whitespace characters
	Diff characters*	9	Difference in the number of characters
	Diff tokens*	1	Difference in the number of whitespace-separated tokens
	Diff repeated characters	0	Difference in the number of repeated characters
	Diff repeated tokens	0	Difference in the number of repeated white-space separated tokens
	Cosine similarity	0	Cosine similarity
	Levenshtein distance*	9	Levenshtein distance
	Optimal string alignment distance	9	Optimal string alignment distance (Damerau-Levenshtein distance)
	Ratio diff to paragraph characters	0.02	Diff characters divided by the length of the edited paragraph
	Ratio diff to revision characters	0.0005	Diff characters divided by the length of r_{v-1}
	Ratio diff to paragraph tokens	0.04	Diff tokens divided by the length of the edited paragraph
	Ratio diff to revision tokens	0.0003	Diff tokens divided by the length of r_{v-1}
	Ratio old to new paragraph	0	Difference in the number of characters in the edited paragraph
	Character n-grams[1]	p,o,e,t,r,y[2]	Presence or absence of n-grams of edited characters
	Token n-grams[1]	poetry[2]	Presence or absence of n-grams of edited tokens
	Simple edit type	Insertion	Modification, Insertion, Deletion or Relocation
Markup	Diff number m	0	Difference in the number of m
	Diff type m	false	Different types of m
	Diff type context m	true[3]	Different types of m within the immediate context of the edit
	Is covered by m	true[3]	Edit is covered by m in r_{v-1}
	Covers m	false	Edit covers m in r_{v-1}
Language	Diff spelling errors*	0	Difference in the number of out-of-dictionary words
	Diff vulgar words*	0	Difference in the number of tokens contained in vandalism word list
	Semantic similarity	-1	Explicit Semantic Analysis with vector indexes from Wiktionary
	Diff POS tags*	false	POS tag sets are symmetrically different
	Diff type POS tags*	0	Number of distinct POS tags

[1] N-gram features are represented as boolean features.
[2] In this example, n = 1 (unigrams).
[3] True if m corresponds to internal link, false otherwise.

Table 1: List of edit category classification features with explanations. The values correspond to the the example edit from Figure 1. m may refer to internal link, external link, image, template or markup element. Features marked with * have previously been mentioned in Adler et al. (2011), Javanmardi et al. (2011) or Bronner and Monz (2012).

M category. WPEC was created in a manual annotation study with three annotators. The overall inter-annotator agreement measured as Krippendorf's α is .67 (Daxenberger and Gurevych, 2012). The experiments in this study are based on the gold standard annotations in WPEC, which have been derived by means of a majority vote for each edit.

WPEC consists of 981 revision pairs, segmented into 1,995 edits. We define edit category classification as a multi-label classification task. For the sake of readability, in the following we will refer to an edit $e^k_{v-1,v}$ as e_i, with $e_i \in E$, where $0 \leq i < 1995$ and E is the set of all edits. An edit e_i is the basic classification unit in our task. Each e_i has to be labeled with a set of categories $y \subseteq C$, where C is the set of all edit categories, $|C| = 21$.

3.2 Features for Edit Category Classification

We grouped our features into *meta data*, *textual*, *markup* and *language* features. An overview and explanation of all features can be found in Table 1. The scheme we apply to group edit category classification features is similar to the system used by Adler et al. (2011). We re-use some of the features suggested by Adler et al. (2011), Javanmardi et al. (2011) and Bronner and Monz (2012), as marked in Table 1. Features are calculated on edited text spans. We label the edited text span corresponding to e_i in r_{v-1} as t_{v-1} and the edited text span in r_v as t_v. In edits which are insertions, we consider t_{v-1} to be empty, while t_v is considered empty for deletions. For Relocations, $t_{v-1} = t_v$.

Table 1 includes the value of each feature for the example edit from Figure 1. This edit modifies the link `[[Dactyl|Dactylic]]` by adding a specification to the target of that link. For spell-checking, we use British and US-American English Jazzy dictionaries.[3] Markup elements are detected by the Sweble Wikitext parser (Dohrn and Riehle, 2011).

Meta data features We consider the comment, author, time stamp or any other flag ("minor change") of r_v as meta data. The Wikimedia user group[4] of an author specifies the edit permissions of this user (e.g. bot, administrator, blocked user). We indicate whether the revision comments or parts of it have been auto-generated. This happens when a page is blanked, i.e. all of its content has been deleted or replaced or when a new page or redirect is created (denoted by the *Comment is auto-generated* feature). Furthermore, edits within a specific section of an article are automatically marked by adding a prefix with the name of this section to the comment of the revision (denoted by the *Auto-generated comment ratio* feature). Meta data features have the same value for all edits in a (r_{v-1}, r_v)-pair.

Textual features Textual features are calculated based on a certain property of the changed text. In a preprocessing step, any wiki markup inside t_{v-1} and t_v is deleted. As for the example edit from Figure 1, t_{v-1} would correspond to an empty string and t_v would be represented as " (poetry)". The n-gram feature spaces are composed of n-grams that are present either in t_{v-1} but not t_v, or vice verse. Character n-grams only contain English alphabet characters, token n-grams consist of words excluding special characters.

Markup features As opposed to textual features, wiki markup features account for the Wikimedia specific markup elements. Markup features are calculated based on the number and type of a markup element m and the surrounding context of an edit. Here, m can be a template, an external or internal link, an image or any other element used to describe markup including HTML tags. The type of m is defined by the link target for internal and external links and images, by the name of the template for templates and by the wiki markup element name for markup elements. Markup features are calculated on text spans t_{v-1} and t_v. Naturally, wiki markup is not deleted beforehand. The edited text spans t_{v-1} and t_v may be located inside a markup element m (e.g. a link or a template). In such cases, our diff algorithm will not label the entire element m, but rather the actually modified text. However, such an edit may change the name of a template or the target of a link (as in the example edit from Figure 1). We therefore include the immediate context s_{v-1} and s_v of each edit and compare the type of potential markup elements m in s_{v-1} and s_v. Here, s_v (s_{v-1}) is defined as t_v (t_{v-1}) including all preceding and follow-

[3] http://sourceforge.net/projects/jazzydicts

[4] http://meta.wikimedia.org/wiki/User_classes

	Revisions	Edits	Cardinality
Train	713	1,597	1.20
Test	89	229	1.24
Dev	89	169	1.21

Table 2: Statistics of the training, test and development set. Cardinality is the average number of edit categories assigned to an edit.

ing characters in r_v (r_{v-1}) which are not separated from t_v (t_{v-1}) by a boundary character (whitespace or line break). The above described features model *what* is actually edited in the text. A number of features are calculated on t_{v-1} only. These features are more likely to inform about *where* an edit is conducted. They specify whether t_{v-1} covers (i.e. contains) a certain wiki markup element and vice versa, i.e. whether t_{v-1} is located inside a text span that belongs to a markup element.

Language Language features are calculated on the context s_{v-1} and s_v of edits, any wiki markup is deleted. For the Explicit Semantic Analysis, we use Wiktionary (Zesch et al., 2008) and not Wikipedia assuming that the former has a better coverage with respect to different lexical classes. POS tagging was carried out using the OpenNLP POS tagger.[5] The vandalism word list contains a hand-crafted set of around 100 vandalism and spam words from various places in the web.

3.3 Experimental Setup

We extract features with the help of ClearTK (Ogren et al., 2008). For the machine learning part, we use Weka (Hall et al., 2009) with the Meka[6] and Mulan (Tsoumakas et al., 2010) extensions for multi-label classification. We use DKPro Lab (Eckart de Castilho et al., 2011) to test different parameter combinations. We randomly split the gold standard data from WPEC into 80% training, 10% test and 10% development set, as shown in Table 2.

Multi-label Classification We report the performance of various machine learning algorithms. A comprehensive overview of multi-label classification algorithms and evaluation measures can be found in Madjarov et al. (2012). Multi-label classification problems are solved by either transforming the multi-label classification task into one or more single-label classification tasks (problem transformation method) or by adapting single-label classification algorithms (algorithm adaption method). Several algorithms have been developed on top of the former methods and use ensembles of such classifiers (ensemble methods). We applied the Binary Relevance approach (BR), a simple transformation method which converts the multi-label problem into $|C|$ binary single-label problems, where $|C|$ is the number of categories. Hence, this method trains a classifier for each category in the corpus (one-against-all). It is the most straightforward approach when dealing with multi-labeled data. However, it does not consider possible relationships or dependencies between categories. Therefore, we tested two more sophisticated methods. Hierarchy of multi-label classifiers HOMER (Tsoumakas et al., 2008) is a problem transformation method. It accounts for possibly hierarchical relationships among categories by dividing the overall category set into a tree-like structure with nodes of small category sets of size k and leaves of single categories. Subsequently, a multi-label classifier is applied to each node in the tree. Random k-labelsets RAKEL (Tsoumakas et al., 2011) is an ensemble method, which randomly chooses l typically small subsets with k categories from the overall set of categories. Subsequently, all k-labelsets which are found in the multi-labeled data set are converted into new categories in a single-labeled data set using the la-

		Random	Majority	BR	HOMER	RAKEL
	Threshold	–	–	.10	.25	.33
Example	**Accuracy**	.09	.13	.50	.44	**.53**
	Exact Match	.06	.13	.35	.36	**.44**
	F1	.09	.13	.55	.47	**.56**
	Precision	.10	.13	.54	.46	**.56**
	Recall	.10	.13	**.61**	.50	.60
Label	**Macro-F1**	.10	.06	.49	.35	**.51**
	Micro-F1	.10	.12	.59	.49	**.62**
Ranking	**One Error**	.90	.87	.42	.48	**.34**

Table 3: Overall classification results with 3 multi-label classifiers and a C4.5 decision tree base classifier, as compared to random and majority category baselines.

[5] Maxent model for English, `http://opennlp.apache.org`

[6] `http://meka.sourceforge.net`

bel powerset transformation (Trohidis et al., 2008). HOMER and BR are among the multi-label classifiers, which Madjarov et al. (2012) recommend as benchmark methods. As underlying single-label classification algorithm, we used a C4.5 decision tree classifier (Quinlan, 1993), as decision tree classifiers yield state-of-the-art performance in the related work.

Multi-label Evaluation We denote the set of relevant categories for each edit $e_i \in E$ as $y_i \in C$ and the set of predicted categories as $h(e_i)$. Evaluation measures for multi-label classification systems are based on either bipartitions or rankings. Among the former, we report example-based (weighting each edit equally) and label-based (weighting each edit category equally) measures. The accuracy of a multi-label classifier is defined as $\frac{1}{|E|}\sum_{i=1}^{|E|}\frac{|h(e_i)\cap y_i|}{|h(e_i)\cup y_i|}$, which corresponds to the Jaccard similarity of $h(e_i)$ and y_i averaged over all edits. We report subset accuracy (exact match), calculated as $\frac{1}{|E|}\sum_{i=1}^{|E|} I$, with $I = 1$ if $h(e_i) = y_i$ and $I = 0$ otherwise. Example-based precision is defined as $\frac{1}{|E|}\sum_{i=1}^{|E|}\frac{|h(e_i)\cap y_i|}{|h(e_i)|}$, recall as $\frac{1}{|E|}\sum_{i=1}^{|E|}\frac{|h(e_i)\cap y_i|}{|y_i|}$, and F1 as $\frac{1}{|E|}\sum_{i=1}^{|E|}\frac{2\times|h(e_i)\cap y_i|}{|h(e_i)|+|y_i|}$. For the label-based measures, we report macro- and micro-averaged F1 scores. As a ranking-based measure, we report one error, which is defined as $\frac{1}{|E|}\sum_{i=1}^{|E|}[\![[arg\ \underset{c\in C}{max} f(e_i,c)] \notin y_i]\!]$, $[\![expr]\!] = 1$ if $expr$ is true and $[\![expr]\!] = 0$ otherwise. $f(e_i, c)$ denotes the rank of category $c \in C$ as predicted by the classifier. The one error measure evaluates the number of edits where the highest ranked category in the predictions is not in the set of relevant categories. It becomes smaller when the performance of the classifier increases.

Table 3 shows the overall classification scores. We calculated a random baseline, which multi-labels edits at random considering the label powerset frequencies it has learned from the training data. Furthermore, we calculated a majority category baseline, which labels all edits with the most frequent edit category in the training data. In Figure 2, we list the results for each category, together with the average pair-wise inter-rater agreement (F1 scores). The F1 scores are calculated based on the study we carried out in Daxenberger and Gurevych (2012).

Parameters and Feature selection All parameters have been adjusted on the development set using the RAKEL classifier, aiming to optimize accuracy. With respect to the n-gram features, we tested values for n = 1, 2 and 3. For comment n-grams, unigrams turned out to yield the best overall performance, and bigrams for character and token n-grams. The word and character n-gram spaces are limited to the 500 most frequent items, the comment n-gram space is limited to the 1,500 most frequent items. To transform ranked output into bipartitions, it is necessary to set a threshold. This threshold is reported in Table 3 and has been optimized for each classifier with respect to label cardinality (average number of labels assigned to edits) on the development set. Since most of the traditional feature selection methods cannot be applied directly to multi-labeled data, we used the label powerset approach to transform the multi-labeled data into single-labeled data and subsequently applied χ^2. Feature reduction to the highest-ranked features clearly improved the classifier performance on the development set. We therefore limited the feature space to the 150 highest-ranked features in our experiments.

For the RAKEL classifier, we set $l = 42$ (twice the size of the category set) and $k = 3$. In HOMER, we used BR as transformation method, random distribution of categories to the children nodes and $k = 3$. For all other classifier parameters, we used the default settings as configured in Meka respective Mulan.

4 Discussion

The classifiers significantly outperformed both baselines. RAKEL shows best performance for almost all measures in Table 3. The simpler BR approach, which assumes no dependencies between categories, still outperforms HOMER.

We trained and tested the classifier with different feature groups (see Table 1), to analyze the importance of single types of features. As shown in Figure 2, textual features had the highest impact on classification performance. On the opposite, language features played a minor role in our experiments. Among the highest ranked individual features for the entire set of categories, we find textual (*Levenshtein distance*, *Simple edit type*), markup (*Diff number*

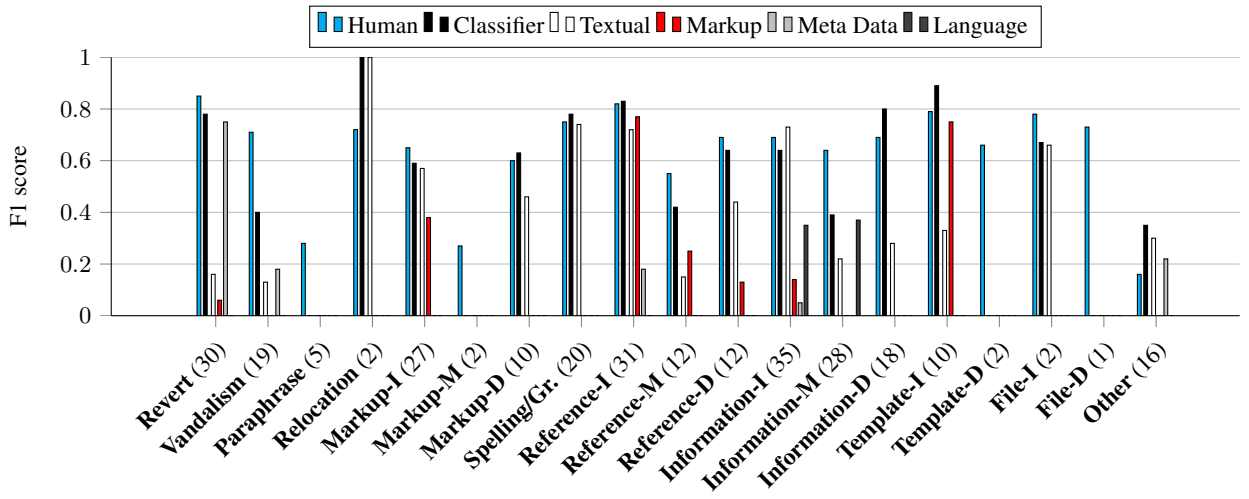

Figure 2: F1 scores of RAKEL with C4.5 as base classifier for individual categories. We add human inter-annotator agreement as average pair-wise F1 scores as well as F1 scores for classifiers trained and tested on single feature groups, cf. Table 1. The number of edits labeled with each category in the test set is given in brackets. The FILE-M and TEMPLATE-M categories are omitted in this Figure, as they had no examples in the development or test set.

markup elements) and meta data (*Number of edits*) features.

Bronner and Monz (2012) report an accuracy of .88 for their best performing system on the binary classification task of distinguishing fluency and factual edits. The best performing classifier in their study was Random Forests (Breiman, 2001). To compare our features with their approach, we mapped the 21 edit categories from Daxenberger and Gurevych (2012) to the binary category set (factual vs. fluency) of Bronner and Monz (2012). Edits labeled as SPELLING/GRAMMAR, MARKUP, RELOCATION and PARAPHRASE are considered fluency edits, the remaining categories factual edits. We removed all edits labeled as OTHER, REVERT or VANDALISM from WPEC. After applying the category mapping, we deleted all edits which were labeled with both the fluency and factual category. The latter may happen due to multi-labeling. This resulted in 1,262 edits labeled as either fluency or factual. On the 80% training split from Table 2, we trained a Random Forests classifier with the optimized feature set and feature reduction as described in Section 3.3. The number of trees was set to 100, with unlimited depth. On the remaining data (test and development split), we achieved an accuracy of .90. Although we did not use the same data set as Bronner and Monz (2012), this result suggests that our feature set is suited for related tasks such as fluency detection.

With respect to vandalism detection in Wikipedia, state-of-the-art systems have a performance of around .82 to .85 AUC-PR on the English Wikipedia (Adler et al., 2011). We suspect that the low performance of our system for Vandalism edits is mostly due to a lower amount of training data, a higher skew in the training and test data and the fact that we did not include features which inform about future actions (e.g. whether a revision is reverted).

Error Analysis Sparseness is a major problem for some of the 21 categories, as shown in Figure 2 by categories such as FILE-D, TEMPLATE-D, MARKUP-M or PARAPHRASE which have only very few examples in training, development and test set. Categories with low inter-annotator agreement in WPEC such as MARKUP-M, PARAPHRASE or OTHER also yielded low classification accuracy. We analyzed frequent errors of the classifier with the help of a confusion matrix. PARAPHRASE edits have been confused with INFORMATION-M by the classifier. Furthermore, the classifier had problems to distinguish between VANDALISM and REVERT as well as INFORMATION-I. Generally, modifications as compared to insertions or deletions perform worse. All of the classifiers we tested, build

their predictions by thresholding over a ranking, cf. Table 3. This generates a source of errors, because the classifier is not able to make a prediction, if it does not have enough confidence for any of the categories. The imbalance of the data, because of the high skew in the category distribution, is another reason for classification errors. In ambiguous cases, the classifier will be biased toward the category with more examples in the training data.

5 A closer look at edit sequences: Mining collaboration patterns

An edit category classifier allows us to label entire article revision histories. We applied the best-performing model from Section 3.3 trained on the entire WPEC to automatically classify all edits in the Wikipedia Quality Assessment Corpus (WPQAC) as presented in previous work (Daxenberger and Gurevych, 2012). WPQAC consists of 10 featured and 10 non-featured articles[7], with an overall number of 21,578 revisions (9,986 revisions from featured articles and 11,592 from non-featured articles), extracted from the April 2011 English Wikipedia dump. The articles in WPQAC are carefully chosen to form comparable pairs of featured and non-featured articles, which should reduce the noise of external influences on edit activity such as popularity or visibility. In Daxenberger and Gurevych (2012), we have shown significant differences in the edit category distribution of articles with featured status before and after the articles were featured. We concluded that articles become more stable after being featured, as shown by the higher number of surface edits and lower number of meaning-changing edits.

Different to our previous approach which is based on the mere distribution of edit categories, in the present study we include the chronological order of edits and use a 10 times larger amount of data for our experiments. We segmented all adjacent revisions in WPQAC into edits, following the approach explained in Daxenberger and Gurevych (2012). During the classification process, we discarded revisions where the classifier could not assign any of the 21 edit categories with a confidence higher than the threshold, cf. Table 3. This resulted in 17,640 remaining revisions. We applied a sequential pattern mining algorithm with time constraints (Hirate and Yamana, 2006; Fournier-Viger et al., 2008) to the data. The latter is based on the PrefixSpan algorithm (Pei et al., 2004). Calculations have been carried out within the open-source SPMF Java data mining platform.[8]

We created one time-extended sequence database for the 10 featured articles and one for the 10 non-featured articles. The sequence databases consist of one row per article. Each row is a chronologically ordered list of revisions. Each revision is represented by the itemset of all edit categories for all edits in that revision (in alphabetical order).

The output of the algorithm are sequential patterns with time constraints. To obtain meaningful results, we constrained the output with the following parameters:

- Minimum support: 1 (the patterns have to be present in each article)
- Time interval allowed between two successive itemsets in the patterns: 1 (patterns are extracted only from adjacent revisions)
- Minimum time interval between the first itemset and the last itemset in the patterns: 1 (the length of the patterns is 2 or higher)

As this output reflects recurring sequences of adjacent revisions labeled with edit categories, we refer to it as *collaboration patterns*. With these parameters, the algorithm discovered 1,358 sequential patterns for featured articles and 968 for non-featured articles. The number of shared patterns in featured and non-featured articles is 427, this corresponds to the number of frequent patterns in a sequence database which contains all 20 featured and non-featured articles. The maximum length of patterns which were found was 6 for featured articles, and 5 for non-featured articles. These numbers show that the defined collaboration patterns seem to have discriminative power for different kinds of articles. Featured articles can be characterized by a higher

[7] http://en.wikipedia.org/wiki/Wikipedia:FA

[8] http://www.philippe-fournier-viger.com/spmf

Featured	① INFORMATION-I ② INFORMATION-I ③ INFORMATION-I ④ INFORMATION-I ⑤ INFORMATION-I
	① INFORMATION-D, INFORMATION-I ② INFORMATION-I ③ INFORMATION-I ④ REFERENCE-I
	① TEMPLATE-D ② REFERENCE-I
Non-Featured	① INFORMATION-I ② INFORMATION-I, REFERENCE-I ③ INFORMATION-I ④ REFERENCE-I ⑤ MARKUP-I
	① MARKUP-I ② REFERENCE-D ③ MARKUP-I
	① VANDALISM ② REVERT

Table 4: Examples of collaboration patterns which have been found in either all featured or all non-featured articles of WPQAC.

degree of homogeneity with respect to their collaborative patterns due to a higher number and length of frequent sequential patterns in featured articles as compared to non-featured articles.

In Table 4, we list some examples of collaboration patterns with a minimum support of 1 which we found in featured, but not non-featured articles, or vice verse. Unsurprisingly, patterns which contain combinations of the most frequent categories (INFORMATION-I, REFERENCE-I), have a high overall frequency. The diversity inside collaboration patterns measured by the number of different edit categories was higher in non-featured articles. For example, the VANDALISM - REVERT pattern was only found in non-featured articles. Patterns in featured articles tended to be more homogeneous, as shown by the first pattern in Table 4, a repetition of additions of information. We conclude that distinguished, high-quality articles, show a higher degree of homogeneity as compared to a subset of non-featured articles and the overall corpus.

6 Conclusion

In this study, we evaluated a novel feature set for building a model to automatically classify Wikipedia edits. Using a freely available corpus (Daxenberger and Gurevych, 2012), our model achieved a micro-averaged F1 score of .62 classifying edits within a range of 21 categories. Textual features had the highest impact on classifier performance, whereas language features play a minor role. The same classifier model obtained state-of-the-art performance on the related task of fluency edit classification. Applications which potentially benefit from our work include the analysis of the writing process in collaboratively created documents, such as wikis or research papers. We have demonstrated how our model can be used to detect collaboration patterns in article revision histories. On a subset of articles from the English Wikipedia, we found that high-quality articles show a higher degree of homogeneity in their collaborative patterns as compared to random articles. Furthermore, automatic edit category classification allows to generate huge amounts of category-filtered training data for NLP tasks, e.g. spelling and grammar correction or vandalism detection. With respect to future work, we plan to include more resources, e.g. the PAN-WVC-10 (Potthast and Holfeld, 2011) or WiCoPaCo (Max and Wisniewski, 2010) to increase the size of training data. A larger amount of labeled data would certainly help to improve the classifier performance for weak categories (e.g. VANDALISM and PARAPHRASE) and sparse categories (e.g. TEMPLATE-D, MARKUP-M). Based on our trained classifier, annotating more examples can be alleviated with the help of active learning.

Acknowledgments

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant No. I/82806, and by the Hessian research excellence program "Landes-Offensive zur Entwicklung Wissenschaftlich-ökonomischer Exzellenz" (LOEWE) as part of the research center "Digital Humanities". We thank the anonymous reviewers for their valuable feedback.

References

B Thomas Adler, Luca Alfaro, Santiago M Mola-Velasco, Paolo Rosso, and Andrew G West. 2011. Wikipedia Vandalism Detection: Combining Natural Language, Metadata, and Reputation Features. In Alexander Gelbukh, editor, *Computational Linguistics*

and Intelligent Text Processing, Lecture Notes in Computer Science, pages 277–288. Springer.

Ablimit Aji, Yu Wang, and Eugene Agichtein. 2010. Using the Past To Score the Present: Extending Term Weighting Models Through Revision History Analysis. *ReCALL*, pages 629–638.

Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. 2012. UKP: Computing Semantic Textual Similarity by Combining Multiple Content Similarity Measures. In *Proceedings of the 6th International Workshop on Semantic Evaluation, held in conjunction with the 1st Joint Conference on Lexical and Computational Semantics*, pages 435–40, Montreal, Canada, USA.

Leo Breiman. 2001. Random Forests. *Machine Learning*, 45(1):5–32.

Amit Bronner and Christof Monz. 2012. User Edits Classification Using Document Revision Histories. In *European Chapter of the Association for Computational Linguistics (EACL 2012)*, pages 356–366, Avignon, France.

Elena Cabrio, Bernardo Magnini, and Angelina Ivanova. 2012. Extracting Context-Rich Entailment Rules from Wikipedia Revision History. In *Proceedings of the 3rd Workshop on The People's Web meets NLP*, pages 34–43, Jeju Island, Republic of Korea.

Aoife Cahill, Nitin Madnani, Joel Tetreault, and Diane Napolitano. 2013. Robust Systems for Preposition Error Correction Using Wikipedia Revisions. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 507–517, Atlanta, GA, USA.

Johannes Daxenberger and Iryna Gurevych. 2012. A Corpus-Based Study of Edit Categories in Featured and Non-Featured Wikipedia Articles. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 711–726, Mumbai, India.

Hannes Dohrn and Dirk Riehle. 2011. Design and implementation of the Sweble Wikitext parser. In *Proceedings of the 7th International Symposium on Wikis and Open Collaboration*, pages 72–81, Mountain View, CA, USA.

Camille Dutrey, Houda Bouamor, Delphine Bernhard, and Aurélien Max. 2011. Local modifications and paraphrases in Wikipedia's revision history. *Procesamiento del Lenguaje Natural*, 46:51–58.

Richard Eckart de Castilho, Iryna Gurevych, and Richard Eckart de Castilho. 2011. A Lightweight Framework for Reproducible Parameter Sweeping in Information Retrieval. In *Proceedings of the Workshop on Data Infrastructures for Supporting Information Retrieval Evaluation*, pages 7–10, Glasgow, UK.

Oliver Ferschke, Torsten Zesch, and Iryna Gurevych. 2011. Wikipedia Revision Toolkit: Efficiently Accessing Wikipedia's Edit History. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. System Demonstrations*, pages 97–102, Portland, OR, USA.

Oliver Ferschke, Johannes Daxenberger, and Iryna Gurevych. 2013. A Survey of NLP Methods and Resources for Analyzing the Collaborative Writing Process in Wikipedia. In Iryna Gurevych and Jungi Kim, editors, *The Peoples Web Meets NLP: Collaboratively Constructed Language Resources*, Theory and Applications of Natural Language Processing, chapter 5. Springer.

Philippe Fournier-Viger, Roger Nkambou, and Engelbert Mephu Nguifo. 2008. A Knowledge Discovery Framework for Learning Task Models from User Interactions in Intelligent Tutoring Systems. In Alexander Gelbukh and Eduardo F. Morales, editors, *Proceedings of the 7th Mexican International Conference on Artificial Intelligence*, Lecture Notes in Computer Science, pages 765–778. Springer.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1):10–18.

Yu Hirate and Hayato Yamana. 2006. Generalized Sequential Pattern Mining with Item Intervals. *Journal of Computers*, 1(3):51–60.

Sara Javanmardi, David W. McDonald, and Cristina V. Lopes. 2011. Vandalism Detection in Wikipedia: A High-Performing, Feature-Rich Model and its Reduction Through Lasso. In *Proceedings of the 7th International Symposium on Wikis and Open Collaboration*, pages 82–90, Mountain View, CA, USA.

Brian Keegan, Darren Gergle, and Noshir Contractor. 2013. Hot Off the Wiki: Structures and Dynamics of Wikipedia's Coverage of Breaking News Events. *American Behavioral Scientist*, 57(5):595–622, May.

Jun Liu and Sudha Ram. 2011. Who does what: Collaboration patterns in the wikipedia and their impact on article quality. *ACM Trans. Management Inf. Syst.*, 2(2):11.

Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and Sašo Džeroski. 2012. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, 45(9):3084–3104.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies, pages 182–190, Montréal, Canada.

Aurélien Max and Guillaume Wisniewski. 2010. Mining Naturally-occurring Corrections and Paraphrases from Wikipedias Revision History. In *Proceedings of the 7th Conference on International Language Resources and Evaluation*, Valletta, Malta.

Rani Nelken and Elif Yamangil. 2008. Mining Wikipedia's Article Revision History for Training Computational Linguistics Algorithms. In *Proceedings of the 1st AAAI Workshop on Wikipedia and Artificial Intelligence*, pages 31–36, Chicago, IL, USA.

Sérgio Nunes, Cristina Ribeiro, and Gabriel David. 2011. Term weighting based on document revision history. *Journal of the American Society for Information Science and Technology*, 62(12):2471–2478.

Philip V. Ogren, Philipp G. Wetzler, and Steven Bethard. 2008. ClearTK: A UIMA toolkit for statistical natural language processing. In *Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP workshop at Language Resources and Evaluation Conference (LREC)*, pages 32–38, Marrakech, Morocco.

Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. 2004. Mining sequential patterns by pattern-growth: the PrefixSpan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1424–1440.

Martin Potthast and Teresa Holfeld. 2011. Overview of the 2nd International Competition on Wikipedia Vandalism Detection. In *Notebook Papers of CLEF 2011 Labs and Workshops*, Amsterdam, Netherlands.

Martin Potthast, Tim Gollub, Matthias Hagen, Johannes Kiesel, Maximilian Michel, Arnd Oberländer, Martin Tippmann, Alberto Barrón-Cedeño, Parth Gupta, Paolo Rosso, and Benno Stein. 2012. Overview of the 4th International Competition on Plagiarism Detection. In *CLEF 2012 Evaluation Labs and Workshop Working Notes Papers*, Rome, Italy.

J. Ross Quinlan. 1993. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic Models for Analyzing and Detecting Biased Language. In *Proceedings of the 51st Annual Meeting on Association for Computational Linguistics*, pages 1650–1659, Sofia, Bulgaria.

Bongwon Suh, Gregorio Convertino, Ed H. Chi, and Peter Pirolli. 2009. The singularity is not near: slowing growth of Wikipedia. In *Proceedings of the 5th International Symposium on Wikis and Open Collaboration*, Orlando, FL, USA.

Konstantinos Trohidis, Grigorios Tsoumakas, George Kalliris, and Ioannis Vlahavas. 2008. Multi-label classification of music into emotions. In *9th International Conference on Music Information Retrieval*, pages 325–330, Philadelphia, PA, USA.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2008. Effective and Efficient Multilabel Classification in Domains with Large Number of Labels. In *Proceedings of the ECML/PKDD 2008 Workshop on Mining Multidimensional Data*, Antwerp, Belgium.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2010. Mining multi-label data. In Oded Maimon and Lior Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, chapter 34, pages 667–685. Springer.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2011. Random k-Labelsets for Multi-Label Classification. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):1079–1089.

Kristian Woodsend and Mirella Lapata. 2011. Learning to Simplify Sentences with Quasi-Synchronous Grammar and Integer Programming. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 409–420, Edinburgh, Scotland, UK.

Elif Yamangil and Rani Nelken. 2008. Mining Wikipedia Revision Histories for Improving Sentence Compression. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. Short Papers*, pages 137–140, Columbus, OH, USA.

Mark Yatskar, Bo Pang, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2010. For the sake of simplicity: unsupervised extraction of lexical simplifications from Wikipedia. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 365–368, Los Angeles, CA, USA.

Fabio Massimo Zanzotto and Marco Pennacchiotti. 2010. Expanding textual entailment corpora from Wikipedia using co-training. In *Proceedings of the COLING-Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*, pages 28–36, Beijing, China.

Torsten Zesch, Christof Müller, and Iryna Gurevych. 2008. Using Wiktionary for Computing Semantic Relatedness. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 861–866, Chicago, IL, USA.

Torsten Zesch. 2012. Measuring Contextual Fitness Using Error Contexts Extracted from the Wikipedia Revision History. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 529–538, Avignon, France.