# Requirement Phrasing Assistance using Automatic Quality Assessment

Arman Allahyari-Abhari, Mathias Soeken, and Rolf Drechsler
Cyber-Physical Systems, DFKI GmbH, Bremen, Germany
Department of Mathematics and Computer Science, University of Bremen, Germany
{abhari, msoeken, drechsle}@cs.uni-bremen.de

***Abstract*—The design of modern hardware systems is a very complex and time consuming process. At the beginning of the design process, requirements need to be specified. Errors in that early design stage derived by misinterpretation of the requirements can be hard to detect and require significant effort and costs to get fixed. To prevent errors, requirements should be written in a comprehensive and unambiguous way. Thus, designers are interested in automatic assistance tools that help writing better requirements. Conventional approaches are usually rule-based, thus many syntactic and semantic properties are not considered. In this paper we introduce an alternative approach to ensure the quality of requirements. The approach has two stages and assists the designer by providing (i) all relevant statistics about the syntactic and semantic properties of the sentence, and (ii) a single consolidated nominal quality predicate for the sentence such as good, medium, or bad. Although such statistical quality assessment leads to already satisfying results, the algorithm prediction reliability can further be enhanced by machine learning techniques. The achieved reliability for quality assessment in combination with the overview of the metrics about syntax and semantic can help the designer to write more comprehensive and less ambiguous requirements.***

## I. Introduction

The quality of requirements written in textual specifications has significant impact on the design flow and on the final product itself. Low quality requirements can easily be misinterpreted and therefore likely lead to errors in the design. On top of that, these errors are also hard to detect, if being detected at all. Consequently, the design has to be revised and deadlines must be postponed, finally leading to higher overall costs of production. A further advantage of well written requirements is an easier application of automatic techniques for requirement formalization, whereas requirements of bad quality complicate the application of such methods, e.g., proposed in [1].

In this work, we present an approach for automatically assessing the quality of requirements, which is based on syntactic and semantic analysis and collects statistical information about the sentence. For example, different parses of a sentence correspond to different interpretations. Polysemy in words and chunks is also put into consideration. Besides leading to more comprehensive specifications, the proposed algorithm is of significant interest to information extraction algorithms such as proposed in [1], [2], [3].

Traditionally, guidelines are defined, which shall ensure that written requirements match certain rules such as described in [4] or [5]. Following this approach, there exist several proprietary tools such as the tools TEKchecker[1] or QuARS [6]. Another tool is the *Requirements Assistant*[2] developed by NASA for the purpose of internal requirements quality assurance. Its purpose is to assure that requirements are complete, consistent, feasible, and distinct. Other related works have also discussed the identification of ambiguities and several metrics to determine the linguistic quality of texts as in [7], [8], where the latter does not work on single sentences but on whole text corpora.

Our algorithm uses different techniques from *natural language processing* (NLP), computes several metrics to determine the quality of a requirement, and finally gets improved by machine learning. Further, we evaluated the proposed approach with manually annotated requirements used in industrial specifications.

## II. Preliminaries

This section describes the NLP techniques that are used in the implementation of the proposed algorithm. For a comprehensive overview about NLP concepts and techniques the reader is referred to [9] and [10].

### A. Phrase Structure Trees

*Phrase structure trees* (PSTs) are trees that contain structural information about a phrase, sentence, or text corpus, where sentence parsing is relevant in the work. They consist of a root node representing the whole sentence, non-terminal nodes constituting the syntactic grammar structure, and terminal nodes being the atomic words of the sentence. The structural analysis as well as the annotation is performed by structural parsers such as the *Stanford parser* contained in the *Stanford CoreNLP toolkit* [11].

Sentences can be ambiguous, which is reflected in the existence of multiple PSTs to a single given sentence. The way of generating the PST depends on the utilized syntactic grammar that the parser uses. In this work we used a *probabilistic context-free grammar* (PCFG, [9]) as back-end grammar for the PST computation. When using a PCFG, each PST is assigned a *score* that indicates the probability to be the correct and intended parse. All PSTs are ranked by their

[1]Clearspecs, TEKchecker, 2015. Available at: http://clearspecs.com
[2]NASA, Requirements Assistant, 2015. Available at: http://www.requirementsassistant.nl

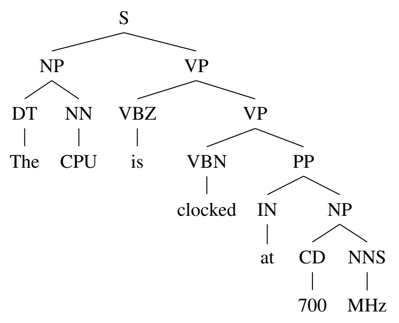

Fig. 1. A typical phrase structure tree

scores. Therefore, the PST with the highest PCFG score is the one which is most likely to be correct.

*Example 1:* A PST for the sentence “The CPU is clocked at 700 MHz.” is illustrated in Fig. 1. The terminal nodes represent the words and the non-terminal nodes directly connected to the terminals are called *part-of-speech* (POS)-tags. DT is a determiner, NN is a noun, VBZ is a verb, etc. One level higher, the non-terminal nodes declare the parts of the sentence, such as noun part (NP), verb part (VP), preposition part (PP), etc. Finally, the root node S is representing the whole sentence.

### B. WordNet

WordNet [12] is a large knowledge database developed by linguists as well as computer scientists at Princeton University and is designed for use under program control. Nouns, verbs, adjectives, and adverbs are grouped into so-called *synsets*, which can be considered as sets of meanings for a word entry. Any word in the database can have several meanings, thus a synset of that word can contain many entries. The WordNet database includes more than 150,000 words and 110,000 synsets, resulting in more than 200,000 pairs connecting words to meanings.

Each entry in a synset is assigned to a short text describing the precise meaning of a word. Additionally, WordNet does not only distinguish between parts of speech such as nouns, verbs, adjectives, and adverbs, but also categorizes the words into subdomains such as *person*, *artifact*, or *quantity*.

### C. Machine Learning

With its origin in the field of artificial intelligence, machine learning can be applied to a wide range of applications, for example, spam detection, face detection, medical diagnosis, or stock trading. The main idea is to identify patterns by learning algorithms. In order to handle the broad range of applications, machine learning algorithms are partitioned into various problem classes, one of them being classification, which has particularly proven to be useful for NLP. Common classifier algorithms are, e.g., Naive Bayes [13], logistic regression [14], and J48 decision tree [15]. A classifier can be supervised or unsupervised and can use filters as preprocessor. Further, the effectiveness of a classifier heavily depends on the set of features chosen for a certain purpose. A classifier tries to build a model on the features by pattern recognition and finally classifies the data with. The selection of adequate features is the central task to accomplish good results from machine learning. The feature set has to harmonize with both, the learning algorithm and the problem. A feature is usually numeric, but can also be nominal such as strings or abstract such as nodes from graphs. To avoid redundancy, the effectiveness measures of a classifier are described in Section VI. There exist very sophisticated tools that can be used for classifying text in natural language, e.g., the Natural Language Toolkit [16] and the Weka data mining software suite [17].

## III. Metrics

The basic idea in this paper is to make use of metrics that can be computed from the data structures described in the previous section to assist the author in writing good requirements. To tackle this problem, we need (i) a single quality metric that can help to predict the quality of a sentence as reliable as possible, and (ii) a set of more detailed metrics which provide the user with an explanation of the main weaknesses of a written sentence in case of a bad quality. It seems obvious that there are several metrics that can be computed, but which metrics can derive the quality best? As mentioned before there are usually many possible phrase structure trees for a given sentence, as they can vary e.g. in their POS-tags, partitioning, or structure. Besides the syntax, a knowledge database like WordNet may also offer many metrics about the semantics. In our approach we consider the syntactic and semantic properties of a given input sentence to build metrics that help to predict the quality of a sentence. For this purpose, we focus on simple statistics such as the sentence length as well as metrics involving possible ambiguities and other varieties in terms of syntax and semantic.

## IV. Assessment of Sentence Quality

This section describes a two-stage algorithm that determines a single quality measure to a given sentence. We are trying to solve the following problem.

*Problem 1 (Sentence quality):* Given an English sentence, the *sentence quality* problem asks to determine a quality measure indicating whether the sentence is *good*, *medium*, or *bad* in terms of comprehension.

To tackle this problem, we make use of established syntactic as well as semantic analysis methods in NLP. The approach is not domain specific and can be applied to any English sentence. The syntactic and semantic quality measures of a sentence are determined separately and their outcome is consolidated into a single value indicating the overall quality of the input sentence.

### A. Syntactic Quality

In this work the syntactic quality of a sentence is considered as directly ambivalent to the number of structural ambiguities. Therefore, a small number of structural ambiguities yields a better syntactic quality. There are a lot of ways to analyze the structure of a sentence. The basic underlying data structure for the computations are phrase structure trees with PCFGs

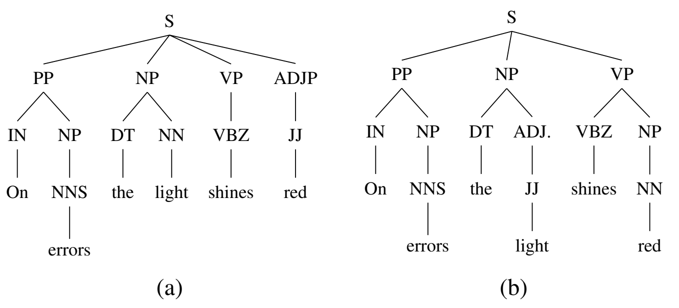

Fig. 2. Two different PSTs of a sentence

as back-end grammar. As a basis for the computation of the syntactic ambiguities we use the PST with the highest PCFG score and compare the structural differences to the next best 100 parses. Based on these PSTs, the computation of the syntactic quality is computed by the following steps:

1) *Isomorphic subtrees:* The highest scored parse tree is compared to each of the next best 100 parse trees. The algorithm computes the difference of any parse tree to the best parse tree in terms of its subtrees. More precisely, for any of the next best 100 parse trees we compute the portion of isomorphic subtrees to the best parse tree and then consolidate these values to an average. The average of these subtree matchings over all 100 next best parses is referred to as *isomorphic subtree ratio* (ISR) and can be assigned a value between 0 and 1. A low ratio corresponds to more structural ambiguities in the sentence. The ISR is the main indicator for syntactic quality.
2) *Sentence length penalty:* A longer sentence corresponds to less structural ambiguities and thus would be assessed a better syntactic quality than a short sentence. This is because longer sentences can be structured in relatively less different ways. Therefore to keep the quality measure consistent, in a last step the syntactic quality computed based on the ISR is decreased by a value according to the amount of words in the sentence. Experiments showed that a penalty of $0.75 \cdot \frac{l}{100}$ seems to be appropriate, where $l$ is the sentence length.

*Example 2:* Given the sentence *"On errors the light shines red."* The corresponding phrase structure tree with the highest PCFG-score (best parse) is depicted in Fig. 2(a), while another possible parse tree (alternative parse) is given by Fig. 2(b). For the best parse, the word *light* is annotated as a noun (NN), but the alternative parse indicates it to be an adjective (JJ). Further, the word *red* is annotated as an adjective (JJ) and as a noun (NN), respectively. To compute the ISR between both trees, any subtree at any level of the best parse is checked whether there is an isomorphic subtree at the same node of the alternative parse. The computed matching percentage is actually the ISR for these two parses. While the subtree beneath the node PP is completely isomorphic for both parses, the rest of the alternative parse slightly differs. E.g., for the NP subtree the parses vary at their right successor, whereas the VP and ADJP subtrees are quite different in their structure. The more subtrees a subtree contains, the more influence it has on the matching ratio. In this case, the ISR for these two parses is at $76.2\%$. The ISR is also computed for another 99 alternative parses and then consolidated into the mean. Further, after the computation the sentence length penalty is applied on the computed syntactic quality. Because the given sentence is quite short, the influence of such penalty is less than usual. Still with its six words, the syntactic quality of the sentence is reduced by $0.75 \cdot \frac{6}{100} = 0.045$, resulting in an overall syntactic quality of $71.7\%$.

## B. Semantic Quality

The semantic quality of a sentence is ambivalent to its amount of semantic ambiguities, which are determined by using WordNet. We consider a word ambiguous, if it is contained in more than one synset in the WordNet dictionary.

Let $n$ be the number of unambiguous words and $m$ be the number of ambiguous words in a given sentence. Then, $a = \frac{n}{(n+m)}$ describes the distinct portion and $b = \frac{m}{(n+m)}$ the ambiguous portion of the sentence. Note that $a + b = 1$, and therefore the sentence is free of ambiguities if $a = 1$. In order to formalize a semantic quality measure we further take the number of meanings for each ambiguous word into account. Let $k_i$ be the number of different synsets of the $i$-th ambiguous word, then the basic semantic quality is given by the formula

$$q_{\text{sem}} = a + b \cdot \frac{m}{\sum_{i=1}^{m} k_i}. \tag{1}$$

*Example 3:* The sentence "The *system* shall *be* able to *sit* at idle and *resume operations* with minimal *delay*." contains 15 words, where the *italic* written words are ambiguous and the others are distinct. The ambiguous words belong to more than one synset in WordNet. In fact, the words *system*, *be*, *sit*, *resume*, *operations*, and *delay* belong to 9, 13, 10, 4, 11, and 2 synsets, respectively. Thus, the ambiguous portion of the sentence consists of 6 words with a sum of 49 synsets. The distinct portion consists of 9 words. Therefore, the distinct portion of the sentence is $a = \frac{9}{15} = 0.6$ and the ambiguous portion is $b = \frac{6}{15} = 0.4$. As a result, the overall semantic quality is $q_{\text{sem}} = 0.6 + 0.4 \cdot \frac{6}{49} \approx 0.65$.

After the computation over ambiguous words, we need to consider compounds, which are nominal compounds such as "header file". If a compound is detected, so if any word chain in the sentence has at least one meaning in WordNet, the semantic quality is modified. The sum of the meanings of every single word in the compound is simply subtracted from the overall sum of meanings and then replaced by the number of meanings of the compound covering these words. Finally, the computation of the quality is also rearranged for the distinct and ambiguous portions of the sentence. Therefore, detecting nominal compounds in a sentence usually reduces the overall number of ambiguities and thus increases the semantic quality.

*Example 4:* The sentence "The operating system shall contain a graphical user interface." comprises nine words. Here, the five words *operate*, *system*, *contain*, *user*, and *interface* have 7, 9, 6, 3, and 4 synsets in WordNet, respectively. Without taking nominal compounds into consideration, there are 29 synsets for the five ambiguous words. Thus, with the naive approach the semantic quality would be $q_{\text{sem}} = \frac{4}{9} + \frac{5}{9} \cdot \frac{5}{29} \approx 0.54$. But the sentence contains the two nominal compounds *operating system* as well as *graphical user interface* and thus the semantic quality has to be assessed appropriately. In WordNet the compounds *operating system* as well as *graphical user interface* have a single synset only. The number of synsets of the single words of *operating system* $(7 + 9 = 16)$ and *graphical user interface* $(1 + 3 + 4 = 8)$ is substituted by 1. Since compounds are considered as single words in terms of their meaning, the distinct and ambiguous portions of the sentence must be conformed. In fact, the semantic quality is computed by $q_{\text{sem}} = \frac{5}{6} + \frac{1}{6} \cdot \frac{1}{6} \approx 0.86$, due to the word *contain* is the only ambiguous word (or compound) left with more than one synset. Therefore, the semantic quality of such sentence is fairly better when nominal compounds are taken into consideration for the computation.

## V. Applying Machine Learning

To improve the reliability of the approach, we apply a machine learning algorithm and feed it with the collected data of syntactic and semantic information. In average, the best results were scored with Naive Bayes and J48 decision tree classifying algorithm, respectively. In this paper, we decided to mostly use a Naive Bayes classifier, because it provided more reliable classification results in our experiments.

Besides the classifier, the algorithm must be fed with suitable data. Therefore, the selection of an appropriate feature set is crucial and should be investigated carefully. The computed metrics from the statistical approach give a wide variety of potential features for the sentence quality. Most of the used metrics are numeric ones such as sentence length, the ISR, or the number of semantically ambiguous words. Only the finally consolidated quality predicate is nominal.

Experiments showed that most of the detailed metrics, which are suitable to determine the syntactic or semantic quality, do not seem to be useful for a classifier. Indeed, at the end most metrics are negligible for the approached machine learning algorithm. A good feature set for our purpose is given as follows:

- sentence length (numeric): the syntactic number of words obtained from sentence segmentation
- overall syntactic quality (numeric): a consolidated quality indicator, e.g., determined by sentence length, ISR, and PCFG-scores
- ambiguous words (numeric): the number of semantically ambiguous words
- overall semantic quality (numeric): a consolidated quality indicator e.g. determined by the number of semantically ambiguous words, the share of the ambiguous portion of the sentence, and the average semantic ambiguities per word over the whole sentence
- overall sentence quality (numeric): consolidates the numeric quality values for syntax and semantic to a single quality measure
- quality predicate (nominal): the overall quality assessment of the statistical algorithm, which can be assigned the values *good*, *medium*, or *bad*

The chosen feature set thus contains six features, while two are syntactic, two are semantic, and two are about the quality of the whole sentence, all derived by the statistical algorithm described in Section IV.

## VI. Experimental Evaluation

We implemented the proposed sentence quality assessment algorithm on top of the *lips* IDE [18], which extends the *Eclipse Modeling Framework* (EMF, [19]) with useful features for NLP. All algorithms are implemented in Xtend/Java.

Experiments showed that a 40:60 ratio of syntax to semantics seems to be a good general configuration for the overall sentence quality of the statistical part of the algorithm. In order to compare the result to a human's subjective opinion, the measured sentence quality is transferred into a quality predicate *good*, *medium*, or *bad*. The test set of requirements has been extracted from various specifications by NASA,[3] NRAO,[4] and Intel.[5] It contains 103 different requirements given by a total of 122 sentences. Exemplary sentences from these requirements are *"DFB shall receive and execute commands to the module at up to 32Hz, and complete those commands in less than 1/10 interrupt period."* and *"For local connections LMS supports both IPv4 loopback and IPv6 loopback connections."*

For comparison the requirements were split into their single sentences. In order to be able to compare the results of the algorithm, the sentences were assessed a quality predicate subjectively by humans. For this purpose we used three different subjective assessments and consolidated them into single quality predicates by majority or averaging, respectively. The set of manually annotated sentences is the basis for evaluation and comparison of the algorithms. In order to evaluate the reliability, we performed experiments for three different configurations. First, we utilize machine learning only, then we evaluate the syntax and semantic based approach described in Section IV, and finally the third experiment combines both.

### A. Machine Learning

A common approach to predict nominal values such as a quality predicate is machine learning. For that purpose we utilized a J48 pruned decision tree classifier, since Naive Bayes did not perform as well. As a preprocessor, the sentence string is tokenized by a filter called *StringToWordVector* and stemmed by *IteratedLovinStemmer* provided by Weka. For processing we used standard 10-fold cross validation. For the

[3]R. Harvey (NASA), Flight Software Requirements, 2010.

[4]T. Morgan (National Radio Astronomy Observatory) Requirements and Functional Specification: EVLA Correlator Backend, 2003.

[5]Intel® Active Management Technology (Intel® AMT) 7.0 Release : FW & SW Product Requirements Document (PRD), 2010.

TABLE I
COMPARISON CLASSIFIER AND SUBJECTIVE OPINION

| | | Algorithm | | | | | |
|---|---|---|---|---|---|---|---|
| Quality | Subj. | good | med. | bad | Precision | Recall | $F_1$ |
| good | 26 | **16** | 9 | 19 | 0.727 | 0.364 | 0.485 |
| medium | 42 | 1 | **3** | 18 | 0.167 | 0.136 | 0.150 |
| bad | 54 | 5 | 6 | **45** | 0.549 | 0.804 | 0.652 |
| total | 122 | 22 | 18 | 82 | 0.544 | 0.525 | 0.501 |

test set of 122 sentences, the J48 algorithm considered 567 features and had a pruned tree size of 35 with 18 leaves.

For indicating the effectiveness of a classifier, there exist several measures. The so-called *confusion matrix* gives a good overview on the single results, more precisely about *true positives (TP)*, *true negatives (TN)*, *false positives (FP)*, and *false negatives (FN)*. Based on these, commonly used measures are (i) precision $P = \frac{TP}{TP+FP}$, (ii) recall $R = \frac{TP}{TP+FN}$, and (iii) the $F_1$-measure $F_1 = \frac{2 \cdot P \cdot R}{P+R}$ as the harmonic mean of the two previous ones. Precision can be read as "How many of the positively classified are correct", while recall is "How many of correct ones are in the results", also referred to as hit rate and thus for lots of applications the most important effectiveness indicator.

Table I shows how the classifier performed. The first two columns depict the quality predicate and the number of subjective quality assessments matching that predicate. The next three integrate the confusion matrix, where the bold written numbers for good, medium, and bad declare the matches, whereas the others indicate the corresponding misses. The last three columns declare *precision*, *recall*, and $F_1$. As can be seen, the quality assessment by machine learning seems to work well for sentences assigned with the predicate *bad* with a recall of 80.4%. But for *good* sentences the recall is only 36.4%, for *medium* sentences the recall is even at fairly weak 13.6% only. For example, the sentence "For local connections LMS supports both IPv4 loopback and IPv6 loopback connections." is subjectively assessed as *good*, but is classified as *bad* by the machine learning algorithm. On the other hand many sentences such as "KVM is used for remote diagnostics and repair." are correctly classified as *bad*. After all, the average $F_1$-measure using machine learning is at solid but improvable 50.1%. Note that in average the machine learning algorithm assessed a worse quality predicate than the subjective opinion.

## B. Statistic Algorithm

Table II provides the results of the quality evaluation of the statistical algorithm. For the sake of a better comparability, the same table structure as before is used. As can be seen in the table, except for assessing the quality of the sentence as bad, the algorithm has an already quite high hit rate. Still, there are 19 assessments determined medium, which should be assessed as bad. With an average precision of 69.9%, an average recall of 66.4% and thus a balanced $F_1$-measure of 68.1%, the statistical algorithm alone already performs quite well. Note that in average the statistical algorithm assessed a slightly better quality predicate than the subjective opinion.

TABLE II
COMPARISON STATISTIC APPROACH AND SUBJECTIVE OPINION

| | | Algorithm | | | | | |
|---|---|---|---|---|---|---|---|
| Quality | Subj. | good | med. | bad | Precision | Recall | $F_1$ |
| good | 26 | **20** | 4 | 2 | 0.625 | 0.769 | 0.697 |
| medium | 42 | 8 | **30** | 4 | 0.566 | 0.714 | 0.640 |
| bad | 54 | 4 | 19 | **31** | 0.838 | 0.574 | 0.706 |
| total | 122 | 32 | 53 | 37 | 0.699 | 0.664 | 0.681 |

TABLE III
COMPARISON COMBINED APPROACH AND SUBJECTIVE OPINION

| | | Algorithm | | | | | |
|---|---|---|---|---|---|---|---|
| Quality | Subj. | good | med. | bad | Precision | Recall | $F_1$ |
| good | 26 | **20** | 4 | 2 | 0.769 | 0.769 | 0.769 |
| medium | 42 | 5 | **33** | 4 | 0.623 | 0.786 | 0.695 |
| bad | 54 | 1 | 16 | **37** | 0.860 | 0.685 | 0.763 |
| total | 122 | 26 | 53 | 43 | 0.759 | 0.738 | 0.741 |

## C. Combined Approach

To improve the algorithm, we apply machine learning as postprocessor. For the purpose of machine learning, we used the Weka machine learning software suite. Table III provides a comparison between the combined assessment algorithm using statistics and machine learning. For machine learning, we used the Naive Bayes classifier algorithm and the feature set as described in Section V. Compared to the purely statistical approach, the combined approach performs even better. In average, precision increased from 69.9% to 75.9%, while recall raised from 66.4% to 73.8%. Overall, the $F_1$-measure raised from 68.1% to 74.1%. Note that in average the combined algorithm mostly neutralizes the under- and overestimation of the previous approaches. Especially with respect to that there are three possible quality predicates (instead of only two), the reliability of the combined approach can be considered a good assistance for writing better requirements.

## VII. CONCLUSION

In this paper we introduced a combined algorithm using statistical information from syntax as well as semantics and machine learning approaches for automatically assessing the quality of sentences written in requirements that are used in the design of complex hardware systems. Therefore, the utilized automatic sentence quality assessment algorithm can assist the designer in writing better requirements in specifications by providing (i) a single quality predicate out of *good*, *medium*, or *bad*, and (ii) more detailed syntactic and semantic metrics for a given the sentence. The proposed approach works in two stages. At the first stage, statistical measures are computed in terms of syntax as well as semantics and then are consolidated into a single quality measure. At the second stage, the statistical approach is enhanced by applying a machine learning algorithm on the collected measures. As machine learning alone scored an $F_1$-measure of 50.1%, the statistical approach clearly beat that result with an $F_1$-measure of 68.1%. The combination of both approaches leads to the best results with an $F_1$-measure of 74.1%. Overall, the prediction of the quality predicate by the combined assessment algorithm often matches the subjective opinion for the applied set of test sentences extracted from several specifications and can help the designer to write more comprehensive and unambiguous requirements.

For future work a thorough case study in an industrial environment should further evaluate the practicability of our approach. Also, it should be investigated why many as bad annotated sentences are determined as medium by the algorithm.

## ACKNOWLEDGEMENT

This work was supported by the General Federal Ministry of Education and Research (BMBF) within the project SPECifIC under grant no. 01IW13001 as well as by the German Research Foundation (DFG) within a Reinhart Koselleck project under grant no. DR 287/23-1.

Further, we would like to thank Andi Buzo and Liana Musat for their supportive discussions.

## REFERENCES

[1] M. Soeken, R. Wille, and R. Drechsler, “Assisted behavior driven development using natural language processing,” in *TOOLS*, 2012, pp. 269–287.

[2] I. G. Harris, “Extracting design information from natural language specifications,” in *DAC*, 2012, pp. 1256–1257.

[3] M. Soeken, C. B. Harris, N. Abdessaied, I. G. Harris, and R. Drechsler, “Automating the translation of assertions using natural language processing techniques,” in *FDL*, 2014.

[4] I. F. Alexander and R. Stevens, *Writing Better Requirements*. Pearson Education, 2002.

[5] IBM. (2009) Get it right the first time: Writing better requirements.

[6] G. Lami, “QuARS: A tool for analyzing requirements,” DTIC Document, Tech. Rep., 2005.

[7] N. Kiyavitskaya, N. Zeni, L. Mich, and D. M. Berry, “Requirements for tools for ambiguity identification and measurement in natural language requirements specifications,” Tech. Rep., 2008.

[8] A. C. Graesser, D. S. McNamara, M. M. Louwerse, and Z. Cai, “Coh-Metrix: Analysis of text on cohesion and language,” in *Behavior Research Methods*, 2004, pp. 193–202.

[9] D. Jurafsky and J. H. Martin, *Speech and Language Processing*. Pearson Prentice Hall, 2008.

[10] N. Indurkhya and F. J. Damerau, *Handbook of Natural Language Processing*, 2nd ed. Chapman & Hall/CRC, 2010.

[11] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, “The Stanford CoreNLP natural language processing toolkit,” in *ACL*, 2014, pp. 55–60.

[12] G. A. Miller, “WordNet: A lexical database for English,” *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[13] H. Zhang, “The optimality of naive bayes,” in *FLAIRS*. AAAI Press, 2004, pp. 562–567.

[14] J. Friedman, T. Hastie, and R. Tibshirani, “Additive logistic regression: A statistical view of boosting,” *The Annals of Statistics*, vol. 38, no. 2, 2000.

[15] D. L. Gupta, A. K. Malviya, and S. Singh, “Performance analysis of classification tree learning algorithms,” *International Journal of Computer Applications*, vol. 55, no. 6, pp. 39–44, 2012.

[16] E. Loper and S. Bird, “NLTK: The natural language toolkit,” in *ETMTNLP*. Association for Computational Linguistics, 2002, pp. 63–70.

[17] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The Weka data mining software: An update,” *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, 2009.

[18] O. Keszöcze, M. Soeken, E. Kuksa, and R. Drechsler, “lips: An IDE for model driven engineering based on natural language processing,” in *NaturaLiSE*, 2013.

[19] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF: Eclipse Modeling Framework 2.0*, 2nd ed. Addison-Wesley Professional, 2009.