Natural Language Engineering **1** (1): 000–000. Printed in the United Kingdom

Non-uniform Language in Technical Writing: Detection and Correction

Weibo Wang[1,2], Aminul Islam[3], Abidalrahman Moh'd[4], Axel J. Soto[5,6], Evangelos E. Milios[1]

[1] *Faculty of Computer Science, Dalhousie University, Canada*
[2] *Dash Hudson, Canada*
[3] *School of Computing and Informatics, University of Louisiana at Lafayette, USA*
[4] *Department of Mathematics and Computer Science, Eastern Illinois University, USA*
[5] *Institute for Computer Science and Engineering, CONICET–UNS, Argentina*
[6] *Department of Computer Science and Engineering, Universidad Nacional del Sur, Argentina*

(*Received 6 February 2020*)

Abstract

Technical writing in professional environments, such as user manual authoring, requires the use of uniform language. Non-uniform language refers to sentences in a technical document that are intended to have the same meaning within a similar context, but use different words or writing style. Addressing this non-uniformity problem requires the performance of two tasks. The first task, which we named Non-uniform Language Detection (NLD), is detecting such sentences. We propose an NLD method that utilizes different similarity algorithms at lexical, syntactic, semantic and pragmatic levels. Different features are extracted and integrated by applying a machine learning classification method. The second task, which we named Non-uniform Language Correction (NLC), is deciding which sentence among the detected ones is more appropriate for that context. To address this problem, we propose an NLC method that combines contraction removal, near-synonym choice, and text readability comparison. We tested our methods using smartphone user manuals. We finally compared our methods against state-of-the-art methods in paraphrase detection (for NLD) and against expert annotators (for both NLD and NLC). The experiments demonstrate that the proposed methods achieve performance that matches expert annotators.

1 Introduction

Technical writing, such as creating device operation manuals and user guide handbooks, is a special writing task that requires the description of a certain product or operation in an accurate manner. To avoid ambiguity and maximize the chances of being understood by most readers, technical writing requires consistency in the use of terminology and uniform language Farkas (1985). There are always demands from modern industries to improve the quality of technical documents in cost-efficient ways Soto et al. (2015).

Our goal in this paper is to avoid inner inconsistency and ambiguity of technical content by identifying non-uniform sentences. Two main subtasks are clearly involved: Non-uniform Language Detection (NLD) and Non-uniform Language Correction (NLC). NLD refers to the identification of sentences that are intended to have the same meaning or usage within a similar context but use different words or writing style. Meanwhile NLC refers to the selection of the sentence that is more formal, precise or easier to understand, and thus more proper for technical writing than other candidate sentences.

The NLD task goes beyond the recognition of similar sentences. Although non-uniform sentences tend to have similar wording, similar sentence pairs do not necessarily indicate a non-uniform language instance. For example, here are four similar sentence pairs cited from an iPhone user manual Apple Inc. (2015), where only two pairs are true non-uniform language instances:

(1) tap the screen to **show** the controls.
tap the screen to **display** the controls.

(2) start **writing** the name of the item.
start **entering** the name of the item.

(3) if the **photo** hasn't been downloaded yet, tap the download notice first.
if the **video** hasn't been downloaded yet, tap the download notice first.

(4) you can also turn bluetooth on or off in control center.
you can also turn **wi-fi and** bluetooth on or off in control center.

The pattern of differences within each sentence pair could be between two individual words, between one word and multiple words, or just one sentence having extra words that the other sentence does not have. Each pattern could be a true or false non-uniform language instance depending on the content and context. As we mentioned earlier, non-uniform language refers to sentences that are intended to deliver the same information and meaning but use different wording. In Example (1), both sentences describe the action of making the control unit visible, but used synonyms, i.e. '**show**' and '**display**'. Therefore, Example (1) is an instance of non-uniform language. Similarly, in Example (2), even though '**entering**' and '**writing**' are not synonyms, since the two sentences aim to describe the same operation but using different verbs to describe the input action, they should be considered as non-uniform language as well. In Example (3), despite of the fact that the only differing words between the sentences, i.e. '**photo**' and '**video**', are both media contents, since they are different objects referring to two different functionalities, they should not be regarded as non-uniform language. Example (4) is a false non-uniform language candidate because each sentence refers to different actions. Note that non-uniform sentences can differ in word length, which makes their identification more difficult than the case of equal word length sentences. Therefore, it is challenging to distinguish true and false occurrences of non-uniform cases based on text similarity algorithms only, and thus finer grained analyses need to be applied.

The NLC task, which aims to choose the most appropriate sentence between a non-uniform sentence pair is also challenging, since in most cases the sentences

can be both syntactically and semantically correct. For instance, in Example (1), which was identified as a non-uniform language case, we aim at selecting one of the two phrases and replace the other one. We refer to this process as *correcting* or *unifying* the sentence pair. The strategy to select the best sentence should be based on selecting the one that is simpler and more common for that context.

To address the problems of NLD and NLC, this paper proposes a pipeline of methodologies for detecting non-uniform language within a technical document at the sentence level, and correcting its instances. The schematic diagram of our approach is shown in Figure 1.

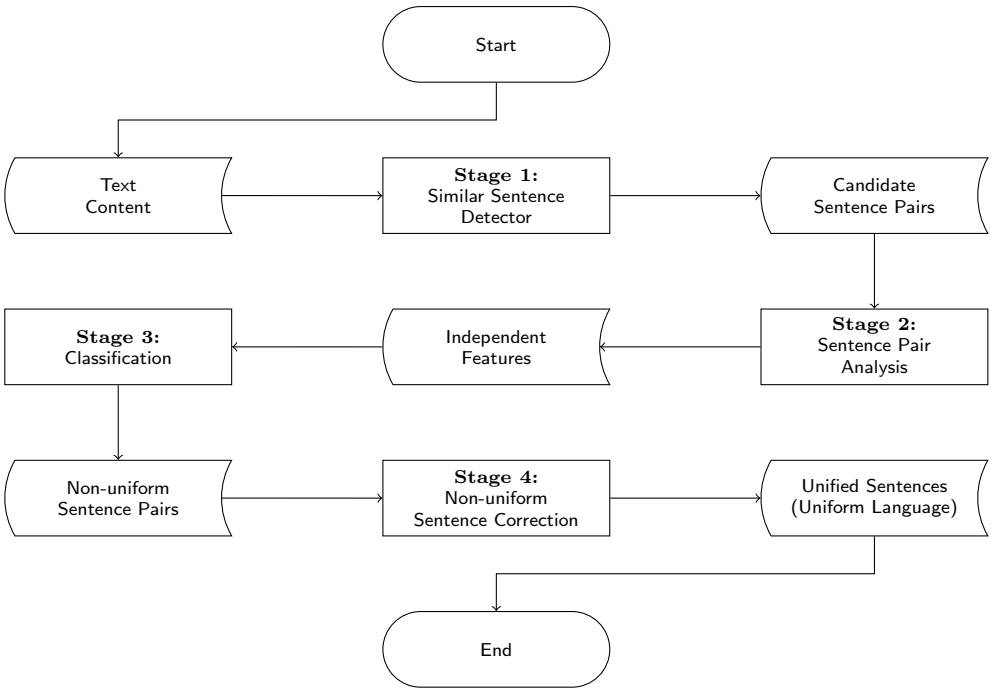

Fig. 1. Schematic diagram of our approach to NLD and NLC

It is worth mentioning that NLD is similar to plagiarism detection and paraphrase detection as all these tasks aim to capture similar sentences with equivalent meaning Das and Smith (2009). However, the goal of authors in plagiarism and paraphrasing is to change several words to increase the lexical differences between texts Androutsopoulos and Malakasiotis (2010), whereas in technical writing, the authors try to avoid such differences, which tend to be subtler. While true positive cases for both NLD and paraphrase detection can exist, they are not likely to happen in practice since textual differences in paraphrase detection tend to be much larger than in NLD.

To address the NLD task, we propose the application of Natural Language Processing (NLP) techniques at lexical, syntactic, semantic, and pragmatic levels. Our approach integrates different techniques such as a Part-of-Speech (POS) tagger Bird

et al. (2009), WordNet Miller et al. (1990), Google Tri-gram Method (GTM) Islam et al. (2012); Mei et al. (2015), and a Flickr-based metonymy tool[1]. In addition, given that recent methods based on recurrent neural networks using Long Short-Term Memory (LSTM) and word embeddings represent the state of the art in sentence similarity Neculoiu et al. (2016), we also integrate a deep siamese LSTM network Mueller and Thyagarajan (2016) into our NLD method. In this way, we apply textual analysis at different levels, and we treat the results as independent features that are finally integrated by applying a classification method based on a Support Vector Machine (SVM).

To address the NLC task, an unsupervised method based on web-based statistical data is proposed. The method we propose here uses the Google N-gram corpus Brants and Franz (2009). Text simplicity and readability measures based on combining different linguistic features, such as De Clercq and Hoste (2016) and Coh-Metrix L2 Reading Index McNamara et al. (2014), are also applied.

Ground truth datasets for both NLD[2] and NLC[3], which were created from three smartphone user manuals are used for evaluation, and they are made publicly available. The experiments on these datasets demonstrate the effectiveness of the proposed approaches for the NLD and NLC tasks.

This paper is organized as follows. Next section discusses related work to the task of identifying non-uniform language. Section 3 and Section 4 describe the main methods used for the NLD and NLC tasks. The experimental work and results are presented in Section 5 and concluding remarks can be found in Section 6.

2 Related Work

Paraphrase Detection (PD) is closely related to NLD. Paraphrasing is a restatement using different words to make it appear different from the original text, and PD aims to detect pairs of texts that have essentially the same meaning. We reviewed studies in the PD area based on convolutional and recurrent neural networks Agarwal et al. (2018); Wang et al. (2018); Bhargava et al. (2017), Recursive Auto-Encoders (RAE) Socher et al. (2011), and Semantic Text Similarity (STS) Islam and Inkpen (2008). However, PD techniques do not perform well on the NLD task as they focus on variations at a coarser granularity. For instance, the four examples provided in the introduction section are recognized as paraphrases, even though only two of the pairs are real non-uniform language cases. Thus, standard PD techniques are unable to make accurate judgments on non-uniform language instances as they do not address the necessary level of detail for the NLD task. Recent approaches on recurrent neural networks for measuring sentence similarity Neculoiu et al. (2016); Mueller and Thyagarajan (2016); Chen et al. (2018) provide interesting directions to account for textual similarity that can be adapted to be used in the context of detecting non-uniform language.

[1] Flickr Service: https://www.flickr.com/services/api/flickr.tags.getRelated.html
[2] NLD data: https://goo.gl/6wRchr
[3] NLC data: https://goo.gl/7vZSJW

Near-duplicate text detection is another area related to NLD. It focuses on short text such as mobile phone short messages, or tweets, which are intended to have the same meaning but differ in terms of informal abbreviations, transliterations, and cyberspeak Gong et al. (2008). The detection and elimination of near-duplicate text are of major importance for other NLP tasks such as clustering, opinion mining, and topic detection Sun et al. (2013). However, the studies in this area focus on reducing the comparison time in large scale text databases and creating informal abbreviation corpora, rather than exploring the text similarity methods. Basic similarity methods, such as Longest Common Subsequence (LCS) are utilized, but they are not sufficient to address the NLD task. LCS accounts for matching words and their order between texts, so using LCS alone will give high recall and low precision for the NLD task. For the following negative example of non-uniform language, LCS returns a high similarity score:

(5) If the **photo** hasn't been downloaded yet, tap the download notice first.
If the **music** hasn't been downloaded yet, tap the download notice first.

Examples of this type are common in technical writing, so other measures are needed besides LCS to recognize NLD positives.

Near-duplicate document detection focuses on web documents that are identical in terms of written content but differ in a small portion of the document such as advertisements, counters and timestamps Manku et al. (2007). Such documents are important to be identified for web crawling and indexing of digital libraries. Since this area focuses on the variations between two documents, especially the variations on metadata rather than the written content within one document, existing solutions are not a good fit for NLD either.

Locality-Sensitive Hashing (LSH) is also worth mentioning here as it can find near-duplicates in linear time Gionis et al. (1999). Moreover, LSH has been used for technical writing to find reuse opportunities (i.e. duplicates or near duplicates) Soto et al. (2015). LSH can be a potential approach to be applied for NLD, or as a prefiltering of candidate senteces, in case a time complexity improvement were needed. However, LSH may miss cases of non-uniform language when semantic variance is present. Moreover, since optimizing time complexity is not a major objective here, but minimizing the chances of missing true instances of non-uniform language, we refrain from applying it in this paper.

Controlled language is another related topic worth mentioning here. Controlled languages have been described as restricted versions of natural languages; they constrain the words, phrases or syntactic constructions that may be used in the composition of a text Höfler (2012). Controlled languages are mainly used to achieve three goals: (1) to make it easier for humans to read and interpret a text, (2) to facilitate translation into other languages, (3) to allow for a direct mapping onto some formal semantic representation accessible to automated reasoning. The first goal above is also a major target of our NLD task. However, controlled language and NLD are different in the way they can be assessed. In controlled languages, a preset guideline rule set is required, and the evaluation is majorly performed by checking whether the text follows all the rules of the guideline. In our NLD task,

on the other hand, there is no such strict set of rules. In addition, non-uniform language cases depend on the occurrences of other sentences in the document.

Non-uniform Language Correction (NLC) is applied on detected instances of non-uniform language, where a sentence pair has minor differences. In such cases, detailed comparison is required to analyze the meaning and usage between the different wording, so that it can be determined which wording fits best given the context. Therefore, NLC has some commonalities with word sense disambiguation methods. Besides the differing word, its context needs to be analyzed to determine which surrounding words are more frequent. A popular resource that has been used in this area is the Google N-gram Corpus Nulty and Costello (2009). Google trigrams and unigrams have been used to determine the similarity between two texts Islam et al. (2012) and to detect and correct real-word spelling errors Islam and Inkpen (2009). In addition, the semantic relation that holds between two nouns in a noun-noun compound phrase such as "flu virus" or "morning exercise" was deduced using lexical patterns from the Google N-gram corpus.

NLC is also related to the areas of text simplicity and readability because an objective of NLC is to select the sentence that is easier to read and understand from a non-uniform language sentence pair. The most popular and classic methods in this area include the Gunnings Fog index Gunning (1969), the Flesch-Kincaid Reading Ease Index Kincaid et al. (1975), the Automated Readability Index Senter and Smith (1967) and the Coleman-Liau Index Coleman and Liau (1975). All the four tests evaluate the readability of a text by consistently breaking the text down into its basic structural elements, which are then combined using an empirical regression formula.

Recent work in NLP combines traditional text readability methods and language features related to text comprehension, cognitive processes, and other factors. Coh-Metrix Graesser et al. (2004) is a computational tool that measures cohesion and text difficulty at various levels of language, discourse, and conceptual analysis. Later on, Crossley et al. proposed L2 readability formula based on Coh-Metrix, which incorporated features that better reflected the psycholinguistic and cognitive reading processes Crossley et al. (2009). Meanwhile, Feng et al. explored various readability approaches and categorized them into different feature groups, i.e. discourse features, language modeling features, parsed syntactic features, POS-based features, and shallow features Feng et al. (2010). These feature groups are compared and a method based on the combination of these features was demonstrated to be one of the state-of-the-art approaches by 2010. One year later, another study that reviewed multiple traditional text readability methods showed that Coh-Metrix L2 Reading Index performed significantly better than traditional readability formulas, which suggests that the variables used in this index are more closely aligned to the intuitive text processing employed by authors when simplifying texts Crossley et al. (2011). In 2016, De Clercq and Hoste grouped 87 popular text readability methods into 10 feature groups and proposed a fully automatic readability prediction pipeline. The experiment has proven that their approach is on par with other approaches using gold-standard deep syntactic and semantic information De Clercq and Hoste (2016).

This paper represents an extension of Wang et al. (2016), where the NLD task was first presented. In this work, we improve our NLD method as well as present a new dataset and method for correction that aims to select the most suitable sentence when a non-uniform language sentence pair is detected.

3 Text Similarity Methods

This section describes the text similarity methods that we experiment with in this work. These algorithms have different characteristics, which intend to provide complementary information when measuring similarity.

Cosine similarity is one of the most popular text similarity algorithms. It measures the degree of similarity of two documents as the correlation between their corresponding bag-of-words vector representations, which can be quantified as the cosine of their angle. Despite the fact that it ignores the relative order of the words in the document, it offers a competitive baseline for text similarity.

GTM is an unsupervised approach based on the Google N-gram corpus for measuring semantic relatedness between texts Islam et al. (2012). The Google N-gram corpus contains English word n-grams (from uni-grams to 5-grams) and their observed frequency counts calculated over one trillion words from web pages collected by Google in January 2006. The text was tokenized following the Penn Treebank tokenization, except that hyphenated words, dates, email addresses and URLs are kept as single tokens. To find the relatedness between a pair of words, GTM takes into account all the trigrams that start and end with the given pair of words. Then, it normalizes their mean frequency using the uni-gram frequency of each of the words as well as the most frequent uni-gram in the Google Web N-gram corpus. The word relatedness metric is then used as a basis to measure document or sentence relatedness. To compute this sentence relatedness, each word from each sentence is compared against any other word in the other sentence, using the tri-gram word relatedness measure. Those pairs of words whose similarity is above a mean and standard deviation, are aggregated to compute the final document or sentence similarity. The specific formulae to compute GTM can be found in its publication Islam et al. (2012) whilst a web service to compute GTM is available online.[4]

Longest Common Subsequence (LCS) is another widely employed technique to measure similarity between texts. It measures the total length of the longest matching substrings in both texts, where these substrings are allowed to be non-contiguous as long as they appear in the same order Chin and Poon (1991); Irving and Fraser (1992). While the original algorithm was applied to find sub-strings of characters, a natural extension is to consider it for words, i.e. the longest common substring has to be composed by a sequence of full words only. The final similarity score can be obtained by dividing the number of words of the longest common subsequence by the length in words of the shortest text under comparison.

A deep LSTM siamese network Mueller and Thyagarajan (2016) was proposed

[4] Online GTM: `http://cgm6.research.cs.dal.ca:8080/DalTextWebApp`

as a siamese adaptation of a recurrent neural network for labeled data comprised of pairs of variable-length sequences with human-annotated similarity. The network is trained using pre-trained word embeddings and a thesaurus-based data augmentation is applied to minimize the effect of the particular wording/syntax in the training data. The network learns to project text strings into a fixed-dimensional embedding space by using only information about the similarity between pairs of strings, and provides text similarity based on this embedding space.

4 Non-uniform Language Detection and Correction

A framework consisting of four stages proposed to address the NLD and NLC tasks is shown in Fig. 1. The first stage extracts candidate sentence pairs that have high text similarity within a document. The second stage performs comprehensive analyses on each candidate sentence pair. The analyses are performed at lexical, syntactical, semantic, and pragmatic levels, where multiple NLP resources such as a POS tagger, WordNet, GTM, a Flickr-based metonymy tool, LSTM networks and pre-trained word embeddings are utilized. The third stage integrates all the previous analyses by applying a classification method based on SVM to classify the candidate sentence pairs as true or false cases of non-uniform language. The final stage performs further analysis on the detected non-uniform sentence pairs, and corrects the non-uniform language by choosing the "best" sentence. The selected sentence is expected to avoid informality, the use of words is expected to be common for its context, as well as it should favor sentences that are easy to understand. To achieve these goals, we propose a three-component approach, which uses a combination of methods based on near-synonym choice, removal of contractions and readability scores.

4.1 Stage 1: Detection of Similar Sentences

The main goal of this stage is to reduce the number of pairs of sentences that need to be analyzed in depth for detecting non-uniform pairs. Domain knowledge on non-uniform language indicates that some minimum level of textual similarity must exist in order for a sentence pair to be considered non-uniform language. To extract candidate sentence pairs, we firstly break down the technical document into sentences, and then use different text similarity measures to compare every sentence in the document against every other. We combine three of the similarity methods described in Section 3. GTM is an unsupervised corpus-based approach for measuring semantic relatedness between texts. LCS focuses on the word order of sentences. Cosine similarity on a bag-of-words representation accounts for matching of common words regardless of their order. We combine GTM, LCS, and Cosine Similarity to filter out candidate non-uniform sentence pairs based on semantics, sentence structure, and word frequency, respectively. Two sentences become a candidate pair only when LCS, Cosine Similarity, and GTM scores are all greater than the filtering thresholds.

The filtering thresholds were set by running experiments at the sentence level

on the iPhone user manual Apple Inc. (2015). Algorithm 1 is used to adapt the filtering thresholds for each average sentence length[5].

```
   Input  : Technical Document, Similarity Function (Sim)
   Output: Threshold-Length_List [(T_1, L_1), ...]
 1 begin
 2     S[n] ⟵ SentenceDetector(Technical Document)
 3     L ⟵ 2                                   /*Initial average length of a sentence pair*/
 4     T ⟵Initial similarity threshold
 5     Step ⟵Threshold increasing step
 6     while (L ≤ 10) do
 7         C ⟵ ∅                               /*Initialize the output sentence container.*/
 8         do
 9             T_low ⟵ T
10             T_up ⟵ T + Step
11             for (i=0; i<n; i++) do
12                 for (j=i+1; j<n; j++) do
13                     AvgL ⟵ (S[i] + S[j])/2
14                     if AvgL ∈ [L − 1, L) then
15                         if (T_low ≤ Sim(S[i], S[j])) and (Sim(S[i], S[j]) ≤ T_up) then
16                             C ⟸^add (S[i], S[j])
17                         end
18                     end
19                 end
20             end
21             T ⟵T+Step
22         while (Check(C))                    /*Check against human labels.*/;
23         Threshold-Length_List ⟸^add (T_low, L)
24         L ⟵L+1
25     end
26 end
```

Algorithm 1: Setting similarity thresholds

We utilize a sentence detector and a tokenizer[6] to divide the text of the manual into a sentence set of N sentence pairs (Line 2). We separately run Algorithm 1 three times to set the threshold sets for GTM, LCS, and Cosine. The thresholds are set based on the average lengths of both sentences of a sentence pair. The average length starts from 2 and is increased by one once the threshold for the current length is set. We discovered that once the sentence length goes above 10, the thresholds vary little. Therefore, we stop the algorithm when the threshold for pairs of average length equal to 10 is found (Line 6).

For each different average length, the algorithm starts by setting an initial similarity threshold and a step value (Line 4-5). This defines a threshold range, where its lower bound is T and the upper bound is *T+Step* (Line 9-10). Then, the algorithm loops over all the sentence pairs (Line 11-20) and add the pairs within the current threshold range into set C (Line 14-16). The similarity of sentence pairs above the previous threshold and below the current threshold are captured and analyzed (Line 15-16). If they consist of all false non-uniform language candidates, we repeat the loop with a higher threshold to filter more false candidates. Once we

[5] See the Example (4) in Section 1, where the sentences of a pair could be unequal in length, thus we compute the average length to represent the length of each candidate pair.

[6] OpenNLP: `https://opennlp.apache.org/documentation/1.5.3/manual/opennlp.html`

Candidate Sentence Pairs with POS Tags	**Ground Truth**
Link/NNP your/PRP device/NN to/TO iTunes/NNS **stores/NNS**	True
Link/NNP your/PRP device/NN to/TO iTunes/NNS **store/NN**	candidate
go/VB to/TO settings/NNS >/SYS general/JJ >/SYS accessibility/NN >/SYS **audio/NN**	False
go/VB to/TO settings/NNS >/SYS general/JJ >/SYS accessibility/NN >/SYS **video/NN**	candidate
Hold/VB the/DT power/NN button/NN for/IN two/NN seconds/NNS to/TO **shutdown/NN** the/DT device/NN	True
Hold/VB the/DT power/NN button/NN for/IN two/NN seconds/NNS to/TO **shut/VBN down/RB** the/DT device/NN	candidate
Hold/VB the/DT power/NN button/NN for/IN two/NN seconds/NNS to/TO **turn/VBN off/IN** the/DT device/NN	True
Hold/VB the/DT power/NN button/NN for/IN two/NN seconds/NNS to/TO **shut/VBN down/RB** the/DT device/NN	candidate

Table 1. POS analysis on candidate sentence pairs

discover that a true candidate is filtered by the current threshold, we stop increasing it and set the threshold to its previous value to minimize the chances of missing a potential candidate pair. The whole experiment is repeated for different sentence pair lengths and different similarity methods.

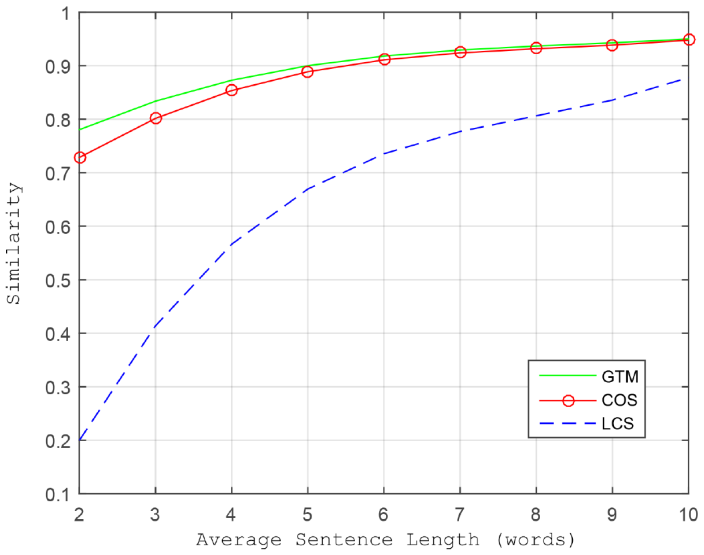

Fig. 2. Candidate filtering thresholds

It is worth mentioning that the thresholds are tunable based on the annotators' capacity to review the candidate pairs and the existing labels. The higher the threshold values are set, the less candidate pairs that this stage will produce. The idea is to set these threshold values as high as possible—as this would reduce the number of sentences to be checked—while aiming at not discarding true positive pairs. The final thresholds are shown in Fig. 2. For example, assume there are two sentences that are nine words long on average. The similarity scores of this pair have to be above all the GTM, LCS and Cosine thresholds (0.943, 0.836, and 0.932, respectively according to Fig. 2) to make it a candidate instance.

Stage 1 could be skipped, if the NLD method is applied on an exhaustive manner on every pairwise sentence combination. However, by applying this filtering, candidate pairs could be reduced in reasonable scale in terms of the size of the corpus with compromising recall. As for precision, around 40% of the candidates we found in our dataset are true non-uniform language cases, where the remaining candidates are expected to be filtered in the second stage. In this way, we can focus our evaluation on the candidate cases (the difficult cases) rather than on every single pair, since most of the pairs in the documents are likely to be negative non-uniform instances, and hence including them in the analysis would lead to over-optimistic results.

4.2 Stage 2: Sentence Pair Analysis

In this stage, we aim to determine for the two sentences of a candidate pair whether they describe the same object or operation using different words or writing style (i.e. true non-uniform language) or they just appear similar but actually have different intended meanings. We describe here different text analysis approaches, where each of them is used afterwards as a feature in a classifier to determine whether sentence pairs are instances of non-uniform language or not.

4.2.1 Part-of-Speech Analysis

POS tags are assigned to each candidate pair using the NLTK Bird et al. (2009) tagger to gain a grammatical view over the sentences. As Table 1 shows, some differences in sentence content can be captured using POS tags, while for other sentences it is not possible. Thus, it is necessary to make further syntactic and semantic analyses to distinguish true candidates from false ones.

We categorized the different POS tags of the differing words, and depending on sentence lengths, into the groups shown in Table 2. The different POS tags are mapped to different categories, which are then used as one more feature of the sentence pair representation.

4.2.2 Character N-gram Analysis

In the character N-gram analysis, the relatedness between the different words of each candidate pair is calculated in terms of character unigram, bigram and trigram

Label	Description	Examples
1	Equal length, same POS tag	/**NN** vs. /**NN**, /**VB** vs. /**VB**
2	Equal length, plural noun with singular noun	/**NN** vs. /**NNS**
3	Equal length, different POS	/**NN** vs. /**VB**
4	Unequal length, extra article	/**NN** vs. /**DT/NN**
5	Unequal length, extra conjunction	/**NN** vs. /**CC/NN**
6	Unequal length, extra adjective	/**NN** vs. /**JJ/NN**
7	Other POS tag types.	/**NN** vs. **N/A**

Table 2. Categorization based on POS tags and sentence length differences

similarity. This similarity can be calculated using the Common N-gram distance (CNG) Kešelj and Cercone (2004):

$$d(f_1, f_2) = \sum_{n \in dom(f_1) \cup dom(f_2)} \left(\frac{f_1(n) - f_2(n)}{\frac{f_1(n)+f_2(n)}{2}}\right)^2. \quad (1)$$

In the equation above, n represents a certain character N-gram unit, $f_i(n)$ represents the frequency of n in sentence i *(i=1,2)*, and $dom(f_i)$ is the domain of function f_i. If n does not appear in sentence i, $f_i(n)$=0. The lower bound of CNG is 0 (when the two units to be compared are exactly the same), but there is no upper bound. CNG was demonstrated to be a robust measure of dissimilarity for character N-grams in different domains Wołkowicz and Kešelj (2013).

4.2.3 WordNet Lexical Relation Analysis

For a given candidate sentence pair, if the different wordings are synonymous to each other, there is a high likelihood that the two sentences try to convey the same meaning but using different expressions. On the other hand, if the different parts of a candidate pair are not related at the lexical level, then it is reasonable to assume that this pair is describing different objects/actions, and thus they might not be instances of non-uniform language.

WordNet is utilized here to analyze the lexical relationship within each candidate pair to determine whether they are synonyms to each other. To perform this analysis, we only used synset information from WordNet, and we only considered words as synonyms if they belong to a same synset. The rationale is that a similar sentence pair tends to be an instance of non-uniform language if the different words are synonyms, rather than having other relationships such as hypernymy, hyponymy, and antonymy. For example, given a similar sentence pair:

(6) if the **photo** hasn't been downloaded yet, tap the download notice first.
if the **video** hasn't been downloaded yet, tap the download notice first.

The sentence pair above is not a non-uniform language instance. However, the re-

latedness score between '**photo**' and '**video**' given by Wu-Palmer metric Wu and Palmer (1994) using WordNet is 0.6, which is fairly high compared to a random word pair. Yet we do not know how these words are related, e.g. "photo is a kind of video", "photo is a part of video", or "photo and video are examples of media content". Thus, we might make wrong judgments based on such a similarity score only. However, using synset information, we know that these words are not synonyms, and thus probably not suggesting a non-uniform language instance. Therefore, we considered as one more feature of our classifier whether mismatching words belong to the same synset or not.

4.2.4 GTM Word Relatedness Analysis

Besides text similarity, GTM also measures semantic relatedness between words Islam et al. (2012). To find the relatedness between a pair of words, GTM takes into account all the trigrams that start and end with the given pair of words. Then, GTM normalizes their mean frequency using unigram frequency of each of the words as well as the most frequent unigram in the Google Web 1T N-gram corpus Brants and Franz (2009) to capture the semantic relatedness among words.

4.2.5 Flickr-Based Concept Analysis

In some cases, word-to-word relatedness exists that goes beyond dictionary definitions, such as metonymy, in which a thing or concept is called not by its own name but rather by the name of something associated in meaning with that thing or concept Kövecses and Radden (1998). Metonymy detection is actually a task at the pragmatic level of NLP that can be applied for NLD in technical writing.

Flickr is a popular photo sharing website that supports time and location metadata and user tagging for each photo. Since the tags are added by humans and aim to describe or comment on a certain photo, the tags are somehow related from a human perspective. As a result, Flickr becomes a large online resource with the potential to find metonymy relationships in text.

Flickr made available statistical information about their dataset that can be used to query related concepts of a certain word or phrase online[7]. We utilized this resource to detect whether the different parts within a candidate sentence pair are related at the pragmatic level. A boolean value that indicates metonymy relationship is obtained and regarded as another feature of our sentence pair representation for our NLD analysis. Examples of relatedness that could be discovered in this stage are given in Table 3.

4.2.6 Deep LSTM Siamese Network Analysis

The use of recurrent neural networks for text similarity has recently received much attention. Therefore, we take the approach proposed by Mueller and Thyagarajan

[7] Flickr Service: https://www.flickr.com/services/api/flickr.tags.getRelated.html

Different Content	Is Metonymy
aeroplane, A380	True
film, hollywood	True
apple, iPhone	True
audio, grayscale	False

Table 3. Examples of metonymy determination using Flickr

(2016), as it has been considered to be the state-of-the-art model for semantic text similarity between pairs of variable-length sentences. This method uses a siamese LSTM-based network that encodes the meaning of a sentence in a fixed-length vector.

We train the model using the Stanford Natural Language Inference (SNLI) Corpus, which collects 570k labeled human-written English sentence pairs[8]. When it comes to word embeddings, we considered the four most popular resources: BERT Devlin et al. (2019), Word2Vec[9], fastText[10], and GloVe[11].

4.3 Stage 3: SVM Classification

All the metrics described above are regarded as features of our candidate sentence pairs. To make a comprehensive judgment based on these different signals, we apply a classification method based on SVM Vapnik (2013) using a Radial Basis Function (RBF) kernel. With regard to hyperparameters, a common practice is to set $\gamma = 1/m$, where m corresponds to the number of independent features, which in our case is 8. Therefore, we train our SVM with $\gamma = 0.125$ and the default values of the e1071 package [12] in R for the rest of the hyperparameters.

In order to account for the variance in the specific train-test split of the data, we split our labeled corpus using different ratios, 50%-50%, 55%-45%, 60%-40%, 65%-35%, 70%-30%, 75%-25%, and 80%-20%, and then train five models based on five random samples with the given split ratio. For each split ratio, we calculate the average performance and its standard deviation to account for the performance and stability of the models.

As we can see from Table 4, between 60%-40% and 65%-35% we obtain the smallest standard deviation. We repeated the above steps within the range from 60%-40% to 65%-35%, and we found that a 61.5%-38.5% split to train the SVM model, the results reach minimum standard deviation. We note that finding the

[8] SNLI Corpus: https://nlp.stanford.edu/projects/snli/
[9] Word2Vec: https://code.google.com/archive/p/word2vec/
[10] fastText: https://fasttext.cc/
[11] GloVe: https://nlp.stanford.edu/projects/glove/
[12] https://cran.r-project.org/web/packages/e1071/e1071.pdf

Training - Test Split Ratio	$\overline{F1}$	σ_{F1}
50% - 50%	83.24	2.98
55% - 45%	84.72	2.79
60% - 40%	87.42	1.95
65% - 35%	88.63	2.05
70% - 30%	86.24	3.77
75% - 25%	88.11	3.63
80% - 20%	89.36	5.18

Table 4. Stability of the NLD method based on different training-test split ratio for our three datasets

most stable split (and not the best performance) helps provide a reliable estimate of the generalization capacity of our NLD method.

4.4 Stage 4: Non-uniform Language Correction

The goal of this stage is to unify all the non-uniform language cases by choosing the sentence that is more proper for technical writing for each non-uniform sentence pair. Based on interviews and collaborative work with technical writers, we set up the following criteria for sentence selection:

1. Formal writing should be preferred over informal. Contractions are the most common source of informality in technical writing, and thus they should be avoided.
2. The choice of words should read natural for the given context.
3. Reading simplicity should be preferred over more complicated grammar structures.

Based on the criteria above, in this stage we distinguish three main subtasks: contraction removal, near-synonym choice, and text readability comparison. The final decision is made by firstly replacing all contractions and then choosing the sentence with the best near-synonym choice score. If there is a tie in the near-synonym choice score or both sentences fail to find a match, we select the sentence with the best readability score as the best sentence.

4.4.1 Contraction Removal

Contraction removal is straightforward as contractions form a fixed set with a one-to-one mapping with no contracted forms. We created a contraction dictionary that includes all contracted expressions along with their uncontracted format (e.g. *won't* $\rightarrow$ *will not*, *don't* $\rightarrow$ *do not*, *can't* $\rightarrow$ *cannot*, etc.), so that contracted forms can be replaced accordingly.

4.4.2 Near-synonym Choice

Near-synonyms are words that have similar meaning or high relatedness, but differ in lexical nuances Islam and Inkpen (2010). For example, *error*, *mistake*, and *blunder* all mean a generic type of error, but *blunder* carries an implication of *accident* or *ignorance* Inkpen (2007). Methods for near-synonym selection fit our task because they look at lexical nuances as well as at the occurring context. This is, when trying to replace a word by a near-synonym, the selected word should fit well with regard to the other words in the sentence. This scenario is indeed what we need to address when deciding which sentence should be kept during NLC if only one word is differing between each sentence. For example, given a non-uniform sentence pair:

(7) You must be **logged** in to a YouTube account to use this feature.
You must be **signed** in to a YouTube account to use this feature.

We compare the only differing words, i.e. *logged* and *signed*, to determine which one fits better within the sentence. The target words *logged* and *signed* become our candidate choices. The task is to choose the highest scoring candidate word as the correct word to unify non-uniform language cases. To score the near-synonym candidate words, a two-phase method using the Google N-gram corpus is proposed.

We represent each sentence using the following notation:

$$S_k = (\ldots\ w_{i-4}\ w_{i-3}\ w_{i-2}\ w_{i-1}\ \boxed{w_i}\ w_{i+1}\ w_{i+2}\ w_{i+3}\ w_{i+4}\ \ldots)$$

where w_i represents the target word at position i, which can be chosen from the set of candidate words. In Example 7, w_i is "logged" for S_1 and "signed" for S_2. Let define s_1 to be the differing word for S_1 ("logged") and s_2 the differing word for S_2 ("signed"). We take into account at most four words before w_i and at most four words after w_i since 5-gram is the longest length in the Google N-gram corpus. Our task is to choose the differing word that best matches with the context. To address this task, we created n-gram units for each s_k, where $2 \leq n \leq 5$, and looked up these units in the corresponding Google N-gram corpus. All the target n-gram units and the corresponding n is listed in Table 5.

We determine the score for each candidate word considering the target position and all N-grams containing that word. In our task, we have two candidate choices for each non-uniform sentence pair at the target position, i, which are s_1 and s_2, and their n-gram frequencies $f_{(1,1)}, f_{(1,2)}, \cdots, f_{(1,j)}, \cdots, f_{(1,14)}, f_{(2,1)}, f_{(2,2)}, \cdots, f_{(2,j)}, \cdots, f_{(2,14)}$, where $f_{(k,j)}$ is the frequency of a N-gram of type number j (as indicated in Table 5) for candidate s_k, where $1 \leq j \leq 14$. In our case, we are looking for as many word matches as possible in the largest context to determine the best near-synonym choice. Therefore, we search starting from 5-grams. If no match is found in 5-grams, we search 4-grams, 3-grams and 2-grams likewise. For each N-gram, we compute a normalized frequency value for each $f_{(k,j)}$ by dividing by the maximum frequency among all

N-gram	Type No.	Target N-gram units
5-gram	1	$\boxed{w_i}\, w_{(i+1)}\, w_{(i+2)}\, w_{(i+3)}\, w_{(i+4)}$
	2	$w_{(i-1)}\, \boxed{w_i}\, w_{(i+1)}\, w_{(i+2)}\, w_{(i+3)}$
	3	$w_{(i-2)}\, w_{(i-1)}\, \boxed{w_i}\, w_{(i+1)}\, w_{(i+2)}$
	4	$w_{(i-3)}\, w_{(i-2)}\, w_{(i-1)}\, \boxed{w_i}\, w_{(i+1)}$
	5	$w_{(i-4)}\, w_{(i-3)}\, w_{(i-2)}\, w_{(i-1)}\, \boxed{w_i}$
4-gram	6	$\boxed{w_i}\, w_{(i+1)}\, w_{(i+2)}\, w_{(i+3)}$
	7	$w_{(i-1)}\, \boxed{w_i}\, w_{(i+1)}\, w_{(i+2)}$
	8	$w_{(i-2)}\, w_{(i-1)}\, \boxed{w_i}\, w_{(i+1)}$
	9	$w_{(i-3)}\, w_{(i-2)}\, w_{(i-1)}\, \boxed{w_i}$
3-gram	10	$\boxed{w_i}\, w_{(i+1)}\, w_{(i+2)}$
	11	$w_{(i-1)}\, \boxed{w_i}\, w_{(i+1)}$
	12	$w_{(i-2)}\, w_{(i-1)}\, \boxed{w_i}$
2-gram	13	$\boxed{w_i}\, w_{(i+1)}$
	14	$w_{(i-1)}\, \boxed{w_i}$

Table 5. List of n-gram units used for the selection of a candidate word

types in the current N-gram frequencies:

$$F(s_{(k,j)}) = \begin{cases} \dfrac{f_{(k,j)}}{\max(f_{(1,1)}, f_{(1,2)}, \cdots, f_{(1,5)}, f_{(2,1)}, \cdots, f_{(2,5)})} & \text{(5-gram)} \\ \dfrac{f_{(k,j)}}{\max(f_{(1,6)}, f_{(1,7)}, \cdots, f_{(1,9)}, f_{(2,6)}, \cdots, f_{(2,9)})} & \text{(4-gram)} \\ \dfrac{f_{(k,j)}}{\max(f_{(1,10)}, f_{(1,11)}, f_{(1,12)}, f_{(2,10)}, f_{(2,11)}, f_{(2,12)})} & \text{(3-gram)} \\ \dfrac{f_{(k,j)}}{\max(f_{(1,13)}, f_{(1,14)}, f_{(2,13)}, f_{(2,14)})} & \text{(2-gram)} \end{cases} \tag{2}$$

Finally, we represent the final score $F(s_k)$ for each candidate word s_k by selecting the maximum value among $F(s_{(k,j)})$:

$$F(s_k) = \begin{cases} \max(F(s_{(k,1)}), F(s_{(k,2)}), \cdots, F(s_{(k,5)})) & \text{(5-gram)} \\ \max(F(s_{(k,6)}), F(s_{(k,7)}), \cdots, F(s_{(k,9)})) & \text{(4-gram)} \\ \max(F(s_{(k,10)}), F(s_{(k,11)}), F(s_{(k,12)})) & \text{(3-gram)} \\ \max(F(s_{(k,13)}), F(s_{(k,14)})) & \text{(2-gram)} \end{cases} \tag{3}$$

Phase 1: Word Choice with Exact Match

First, we obtain 5-gram units from types 1 to 5 for both candidate s_1 and s_2, and calculate $F(s_1)$ and $F(s_2)$ using Equation 2 and 3, respectively. If we have $F(s_1) > F(s_2)$, we choose s_1, and vice versa. If we could not find any match for neither s_1 nor s_2, we obtain 4-gram units from types 6 to 9 for s_1 and s_2, and compare $F(s_1), F(s_2)$ likewise. If no match is found again, we apply the same method to search for 3-grams from types 10 to 12, and for 2-grams from types 13 and 14, respectively. If no n-grams of types 1 to 14 are found for neither candidate s_1 nor s_2, we perform Phase 2.

Phase 2: Word Choice with Relaxed Condition

In technical writing, sentences may contain special terminology, numerical parameters, or domain-specific words that could result in a zero match with the Google N-gram corpus. For example, in this non-uniform sentence pair:

(8) Use a dock connector to usb cable to connect the 30-pin **port** of the adapter to your computer.
Use a dock connector to usb cable to connect the 30-pin **jack** of the adapter to your computer.

the target word set is {"port","jack"}, but the adjacent word "30-pin" is not common enough in the Google N-gram corpus, therefore we could hardly find a match in N-grams.

In order to address this issue, we follow Phase 1 with some small changes. To increase the chance of finding matches in the Google N-gram corpus, we relax the matching condition by assigning a wildcard to the first word of all the N-gram units. For example, in the given sentence pair above, any 5-gram unit in the form of '* connect the 30-pin jack/port', '* the 30-pin jack/port of', '* 30-pin jack/port of the', '* jack/port of the adapter', where * could represent any word, will be regarded as a match. Likewise, we apply relaxed matching for 4-gram, 3-gram, and 2-gram units.

4.4.3 Text Readability Score

Some non-uniform sentence pairs differ in more than one word. For instance, in the following non-uniform sentence pair:

(9) Save **an attached video** to your Camera Roll album: Touch and hold the attachment, then tap Save Video.
Save **a video from an email message** to your Camera Roll album: Touch and hold the attachment, then tap Save Video.

we decide which sentence to keep based on the third criterion, which is to penalize sentence complexity, so that the most simple and easy-to-read sentence is kept. We first considered traditional text readability as our baseline methods. In this stage, we considered the Flesch-Kincaid Reading Ease Test and the Coleman-Liau

Index (CLI), which are based on well-differentiated features, i.e. features based on syllables per word and features based on characters per word. However, preliminary assessments indicated that CLI seemed better suited for short technical text, so we decided to use it as our baseline method for measuring text readability.

After defining the baseline method, we explored more recent work in text readability. One of the state-of-the-art approaches in automatic text readability has been proposed by De Clercq and Hoste (2016). However, when applying their online method to our data [13], we found that it does not perform well on short text and it is not able to quantify readability differently in most of our candidate sentence pairs. Another popular approach is Coh-Metrix L2 Crossley et al. (2009), which has been evaluated in an extensive benchmark Crossley et al. (2011), and it was implemented as part of an automated text evaluation system McNamara et al. (2014). Therefore, as a second readability approach for our NLC task we incorporated the Coh-Metrix L2 index.

5 Experiments and Evaluation

In this section we present the datasets, experimental work and results, including a comparison with other baseline methods.

5.1 Datasets

We collected smartphone user manuals from iPhone Apple Inc. (2015), LG LG (2009) and Samsung Samsung (2011), which are available online. Then, we performed Stage 1 on the three different datasets, and identified 325 candidate sentence pairs, which we regard as our candidate dataset. Before applying the sentence analysis and classification stages, each candidate sentence pair in the dataset was labeled by three different annotators as *true* or *false*. The annotators worked separately to label the sentence pairs. Cases of disagreement were sent again to the annotators to double-check their judgment. Then, the ground truth for each instance is generated by annotators' voting. Some statistics from the manuals are shown in Table 6.

User manual	**Data volume (Pages)**	**Candidate pairs (True, False)**
iPhone	196	208 (102, 106)
LG	274	54 (16, 38)
Samsung	190	63 (32, 31)

Table 6. Dataset statistics

Before the SVM-based classification stage, we split the dataset, which consists

[13] https://www.lt3.ugent.be/tools/machine-learning-readability/

of 150 true positives and 175 true negatives, into a training set (DS_{train}), and a testing set (DS_{test}). Following the procedure discussed in Section 4.3, 61.5% of the data was used for training and the remaining for testing. Considering that the data distribution is nearly balanced in terms of true and false instances, there was no need to take stratified samples. This validation procedure was repeated ten times to account for rthe variance of the results.

5.2 Evaluation Methods and Results

In this section we discuss evaluation and results for both NLD and NLC tasks using the datasets mentioned in the previous section.

5.2.1 Evaluation of NLD Method

The performance of each annotator against the majority voting is evaluated in terms of Precision, Recall, Accuracy, and F-measure. These results along with the number of true/false, positive/negative cases for each annotator are presented in Table 7.

Parameters	**Expert 1**	**Expert 2**	**Expert 3**
True-positive	130	99	125
True-negative	161	164	166
False-positive	20	51	25
False-negative	14	11	9
Precision	**86.67**	66.00	83.33
Recall	90.27	90.00	**93.28**
Accuracy	**89.54**	80.92	**89.54**
F-Measure	**88.43**	76.15	88.03

Table 7. Evaluation of annotator performance against majority voting

To measure the agreement among annotators, the Fleiss' Kappa test Fleiss and Cohen (1973) is used. Fleiss' Kappa is an extension of Cohen's Kappa Cohen (1968). Unlike Cohen's Kappa, which only measures the agreement between two annotators, Fleiss' Kappa measures the agreement among three or more annotators.

In our case, we have 3 annotators (the annotator number n is 3), each annotator labeled 325 candidate pairs (the subject volume N is 325), and each candidate pair was labeled as either negative or positive (the value of category k is 2). The final Fleiss' Kappa Value is 0.545, which indicates a moderate agreement level (0.41-0.60) based on the Kappa Interpretation Model Fleiss and Cohen (1973). In other words, the performance of the annotators reveal that the NLD task is not simple,

since there are many cases that are ambiguous and hard to make correct judgments on, even for humans. For instance:

(10) If you keep entering **zhuyin** without spaces, sentence suggestions appear.
If you keep entering **pinyin** without spaces, sentence suggestions appear.

Both *zhuyin* and *pinyin* refer to a sound-based writing system for Chinese. From some online resources, zhuyin and pinyin are regarded as the same language system, which starts by the syllables 'bo', 'po', 'mo' and 'fo'. However, some resources point out some differences in history and usage between *pinyin* and *zhuyin* [14]. In general, it is hard to tell the difference in meaning between zhuyin and pinyin, and therefore, it is difficult to determine whether the sentence pair above is describing the same language feature (positive non-uniform language case) or describing different language features (negative non-uniform language case).

The best performance of annotators is highlighted and regarded as the upper bound performance (UB) of the NLD task on our dataset, as shown in Table 7. An unsupervised paraphrase detection system named STS Islam and Inkpen (2008), as well as a supervised PD system named RAE Socher et al. (2011), are utilized to generate the baselines of the NLD task. STS uses the similarity score of 0.5 as the threshold to evaluate their method in the PD task. RAE applies supervised learning to classify a pair as a true or false instance of paraphrasing. These approaches are utilized in our evaluation as baselines for the NLD task.

After defining the upper bound and baseline performances of the NLD task, we performed a series of experiments to determine which pre-trained word embeddings with our LSTM network among BERT, Word2Vec, fastText and GloVe. We performed a series of cross-validation for our LSTM network by varying the pre-trained word embeddings. In the end, we calculated both the average and standard deviation of recall precision, accuracy, and F1 score for each word embedding.

Word Embedding	$(\overline{R}, \sigma_R)$	$(\overline{P}, \sigma_P)$	$(\overline{A}, \sigma_A)$	$(\overline{F1}, \sigma_{F1})$
BERT	(76.8, 1.8)	(86.1, 2.0)	(81.6, 0.9)	(81.2, 0.7)
Word2Vec	(73, 1.1)	(73, 4.8)	(75.2, 1.8)	(73, 2.6)
fastText	(74.2, 2.1)	(80, 2.4)	(78, 2)	(77, 2.1)
GloVe	(73.4, 2.0)	(79.1, 2.0)	(77.2, 1.8)	(76.2, 1.8)

Table 8. Pre-trained word embedding selection

BERT achieves the best performance among the four methods, as shown in Table 8. Therefore, we use BERT word embeddings for the deep LSTM siamese network to calculate text similarity for all our candidate sentence pairs.

After having each feature calculated, we evaluated our proposed NLD method by

[14] https://en.wikipedia.org/wiki/Bopomofo

training the SVM classifier on DS_{train}, and then performing classification using the SVM classifier on DS_{test}. The result of our NLD method is shown in the last row of Table 9. The first row presents the upper bound performance and the following two rows present the baseline performances. To assess the importance of each feature utilized in the proposed framework, we performed a feature ablation study Cohen and Howe (1988) on N-gram, POS analysis, lexical analysis [15], Flickr, and LSTM siamese networks separately on the DS_{test}. These results are listed in the remaining rows of Table 9.

Method	**R**(%)	**P** (%)	**A**(%)	**F1**(%)
UB	93.28	86.67	**89.54**	**88.43**
STS	**100**	46.15	46.15	63.16
RAE	**100**	46.40	46.40	63.39
POS	77.78	72.77	78.40	76.52
CNG Uni-gram	11.11	35.29	52.80	16.90
CNG Bi-gram	44.44	61.54	64.00	51.61
CNG Tri-gram	50.00	62.79	65.60	55.67
GTM + WordNet	85.18	59.74	68.80	70.23
Flickr	48.96	94.00	74.00	64.38
LSTM Siamese	81.29	82.35	82.76	81.82
NLD method	88.16	87.58	88.62	87.87

Table 9. Evaluation of NLD method

Student's t-tests are applied after running NLD, STS, RAE, and UB methods on the F-measure metric. The tests reveal that the performance of NLD method is significantly better than STS and RAE, no significant differences could be found between UB and NLD methods. These results demonstrate that NLD method would represent an effective approach for NLD that is on the par with annotator judgment and improves on state-of-the-art approaches for related tasks.

5.2.2 Evaluation of NLC Method

The Fleiss' Kappa test showed that our candidate sentence pairs contain several difficult cases that human annotators disagreed on, as indicated in the previous section. Such cases are controversial and therefore not appropriate for the evaluation of NLC. To evaluate NLC, we only selected non-uniform language cases, where all

[15] We combine GTM and WordNet by applying SVM regression on the two features to represent the lexical analysis result.

the human annotators labeled them as true candidates. Based on this criterion, 78 sentence pairs that are "definite" true NLD cases were selected for evaluating the NLC task.

In order to evaluate our proposed NLC method, five human annotators were invited to annotate the 78 sentence pairs. Each annotator labeled each sentence pair by choosing one of the following options:

1. Sentence 1 is more appropriate than Sentence 2.
2. Sentence 2 is more appropriate than Sentence 1.
3. It is hard to say which sentence should be selected.

For each sentence pair, if Sentence 1 (or Sentence 2) receives the majority of the votes (more than 3 votes), we regard Sentence 1 (or Sentence 2) as the correct one. Otherwise, we label the ground truth as "Hard to Say". 22 out of 78 sentence pairs were labeled as"Hard to Say", which shows that the NLC task is not a simple task either.

Analogous to the evaluation of NLD method, Fleiss' Kappa test is performed on the annotated dataset that is used to evaluate our NLC method. In this case, we have 5 annotators (the annotator number n is 5), each annotator labeled 78 candidate pairs (the subject volume N is 78), each candidate pair is labeled as either "Sentence 1", "Sentence 2", or "Hard to Say" (the value of category k is 3). The final Fleiss' Kappa Value is 0.59. Similar with NLD method, this Kappa test result for NLC method also achieves a moderate agreement level (0.41-0.60) based on the Kappa Interpretation Model Fleiss and Cohen (1973), which reveals the difficulty of the NLC task.

We performed our NLC method on the evaluation dataset along with an ablation study. Excluding the "Hard to Say" cases, we compared the results of the methodology described in Section 4.4 against the ground truth. The results are shown in Table 10.

Method	**R**(%)	**P** (%)	**A**(%)	**F1**(%)
Contraction Removal	11.54	9.09	5.36	10.17
CLI	71.42	60.61	62.50	65.57
Coh-Metrix L2	76.47	78.79	73.21	77.61
Near-synonym Choice	80.80	84.84	78.57	82.35
NLC method	93.94	86.11	87.5	89.86

Table 10. Evaluation of NLC method

We performed four methods to select a sentence for each sentence pair, shown in Table 10. By performing Contraction Removal only, we acquire very poor results. By applying the baseline text readability method CLI only, we can only achieve 62.50%

overall accuracy while by using Coh-Metrix L2 Reading Index we achieve 73.21%. Finally, by using the Near-synonym Choice method only, we achieve 78.57% overall accuracy. Yet by combining Contraction Removal, Coh-Metrix L2 Reading Index and Near-synonym Choice as the proposed NLC method, we are able to achieve an overall accuracy of 87.5%.

5.3 Discussion

The PD systems STS and RAE regard almost all the test cases as true non-uniform language cases, so the recall is 100% but the precision is very low, as shown in Table 9.

It is worth noting that by using character N-gram analysis alone, it is not possible to obtain good results. This is because the character N-gram analysis is unable to capture semantic relatedness, while the NLD task relies heavily on discovering such relatedness. The reason we applied the N-gram analysis is to use it as a supplementary method to catch differences such as between '**shut down**' (two words) and '**shutdown**' (one word), or for spelling variations.

POS analysis provides a syntactic perspective for the text instances. For instance, '**then(/RB)**' versus '**and(/CC)**', and '**store(/NN)**' versus '**stores(/NNS)**', the differences can be reflected in POS tags. Yet, POS analysis alone could not capture the difference between words such as '**writing(/VBG)**' versus '**entering(/VBG)**' since they share the same POS tag. These features make POS analysis outperform the character N-gram analysis, but not semantic-based approaches.

Lexical analysis (GTM and WordNet) achieves the best recall ratio since it can provide semantic relatedness, which is the most important aspect for the NLD task. Flickr is utilized as a supplementary resource to provide pragmatic relatedness.

In addition to these classical NLP methods, the LSTM siamese network trained on the SNLI corpus provides a good overall insight of text similarity and achieves the best F1 score. Since the model captures different characteristics of each sentence and takes into account lexical, structural and semantic aspects, it is not surprising to see this method achieving both good recall and precision.

By combining the different types of analyses above, the differences of each sentence pair are analyzed at different NLP levels, and thus the relatedness and difference from structural, grammatical, syntactic, semantic and pragmatic perspectives can be captured and integrated by the classification method. In this way, the combination of all these features yield better performance than any of the previous methods independently.

As for the NLC task, using CLI and Coh-Metrix L2 Reading Index alone did not achieve high performance because text complexity and readability is not a major factor that influences sentence selection. The major factor is whether the word fits the context and that is why using Near-synonym Choice method alone, which considers the context of the sentence, achieves 78.57% overall accuracy. By combining the Contraction Removal, Near-synonym Choice method and Coh-Metrix L2 Reading Index, our proposed NLC method is able to analyze whether words within the

sentence fit in a natural way and achieve a good overall readability, and therefore matches expert annotation better when deciding which sentence to use.

6 Conclusions

This paper discussed non-uniform language in technical writing and proposed to divide it into two main tasks. The first task, Non-uniform Language Detection (NLD), aims to detect non-uniform language for technical writing at the sentence level, while the second task, Non-uniform Language Correction (NLC), aims to decide which sentence among the detected non-uniform sentences is more appropriate for the context. We first proposed the NLD method by integrating different similarity algorithms at the lexical, syntactic, semantic, and pragmatic levels through an SVM-based classification method. We then proposed the NLC method, which takes into account semantic relatedness, context association, and text readability methods.

To evaluate the proposed NLD method, we created a dataset of candidate sentence pairs using several smartphone user manuals. Three annotators manually labeled all the candidate instances identified in the detection of similar sentences (stage 1). With the generated ground truth, a series of experiments using our implemented system were carried out. We compared our NLD method against state-of-the-art paraphrase detection methods. All the sentence pairs that were labeled as true non-uniform language cases by the three annotators in the previous task were used in the evaluation of NLC method. To determine which sentence is more appropriate for technical content in that context, five annotators labeled the true instances of non-uniform language. We finally compared our NLC method output against the ground truth generated by the human annotators. We also performed an ablation test using individual features on both NLD and NLC methods.

Our experiments show that our proposed methods achieve F1 scores above 87% and 89% for NLD and NLC tasks, respectively. Considering the agreement of annotator judgments as reflected by Fleiss' Kappa value, the NLD and NLC tasks are fairly difficult. Yet, the experiments reveal that the performance of our NLD and NLC methods is close to human annotation performance. Technical writing is an area that has not been extensively investigated and we consider that there are other research avenues that can be explored. One of them is the consideration of non-uniform language as a real-based value that is linked to the degree of non-uniformity that a sentence pair exhibits. In addition, intelligent user interfaces could be studied that incorporates the methods hereby provided for the assistance of technical writers.

Acknowledgments

This research work was supported by Innovatia Inc. and NSERC. We thank Andrew Albert, David Crowley, and Erika Allen who introduced the NLD task, and provided expertise that contributed to the preparation of this paper.

References

Agarwal, B., **Ramampiaro, H.**, **Langseth, H.**, **and Ruocco, M.** 2018. A deep network model for paraphrase detection in short text messages. *Information Processing & Management*, 54(6):922–937.

Androutsopoulos, I. and Malakasiotis, P. 2010. A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligent Research*, 38(1):135187.

Apple Inc. 2015. iPhone User Guide For iOS 8.4 Software. `https://manuals.info.apple.com/MANUALS/1000/MA1565/en_US/iphone_user_guide.pdf` [Accessed: 01-Dec-2015].

Bhargava, R., **Sharma, G.**, **and Sharma, Y.** 2017. Deep paraphrase detection in indian languages. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, ASONAM 17, 11521159, New York, NY, USA. Association for Computing Machinery.

Bird, S., **Klein, E.**, **and Loper, E.** 2009. *Natural language processing with Python*. O'Reilly Media, Inc.

Brants, T. and Franz, A. 2009. Web 1T 5-gram, 10 european languages version 1. *LDC2009T25. Web Download. Linguistic Data Consortium, Philadelphia*.

Chen, Q., **Hu, Q.**, **Huang, J. X.**, **and He, L.** 2018. CA-RNN: using context-aligned recurrent neural networks for modeling sentence similarity. In *Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 265–273. AAAI Press.

Chin, F. Y. L. and Poon, C. K. 1991. A fast algorithm for computing longest common subsequences of small alphabet size. *Journal of Information Processing*, 13(4):463–469.

Cohen, J. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213.

Cohen, P. R. and Howe, A. E. 1988. How evaluation guides AI research: The message still counts more than the medium. *AI magazine*, 9(4):35.

Coleman, M. and Liau, T. L. 1975. A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60(2):283.

Crossley, S., **Salsbury, T.**, **and McNamara, D.** 2009. Measuring l2 lexical growth using hypernymic relationships. *Language Learning*, 59(2):307–334.

Crossley, S. A., **Allen, D. B.**, **and McNamara, D. S.** 2011. Text readability and intuitive simplification: A comparison of readability formulas. *Reading in a foreign language*, 23(1):84–101.

Das, D. and Smith, N. A. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pp. 468–476. Association for Computational Linguistics.

De Clercq, O. and Hoste, V. 2016. All mixed up? finding the optimal feature set for general readability prediction and its application to english and dutch. *Computational Linguistics*, 42(3):457–490.

Devlin, J., **Chang, M.-W.**, **Lee, K.**, **and Toutanova, K.** 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Pro-*

ceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics, pp. 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Farkas, D. K. 1985. The concept of consistency in writing and editing. *Journal of Technical Writing and Communication*, 15(4):353–364.

Feng, L., **Jansche, M.**, **Huenerfauth, M.**, **and Elhadad, N.** 2010. A comparison of features for automatic readability assessment. In *Proceedings of the 23rd international conference on computational linguistics: Posters*, pp. 276–284. Association for Computational Linguistics.

Fleiss, J. L. and Cohen, J. 1973. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and Psychological Measurement*, 33(3):613–619.

Gionis, A., **Indyk, P.**, **and Motwani, R.** 1999. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pp. 518–529, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Gong, C., **Huang, Y.**, **Cheng, X.**, **and Bai, S.** 2008. Detecting near-duplicates in large-scale short text databases. In *Advances in Knowledge Discovery and Data Mining*, pp. 877–883. Springer.

Graesser, A. C., **McNamara, D. S.**, **Louwerse, M. M.**, **and Cai, Z.** 2004. Coh-metrix: Analysis of text on cohesion and language. *Behavior research methods, instruments, & computers*, 36(2):193–202.

Gunning, R. 1969. The fog index after twenty years. *Journal of Business Communication*, 6(2):3–13.

Höfler, S. 2012. Legislative drafting guidelines: How different are they from controlled language rules for technical writing? In *International Workshop on Controlled Natural Language*, pp. 138–151, Berlin, Heidelberg. Springer.

Inkpen, D. Z. 2007. Near-synonym choice in an intelligent thesaurus. In *HLT-NAACL, Rochester, NY, April 22-27, 2007*, pp. 356–363.

Irving, R. W. and Fraser, C. 1992. Two algorithms for the longest common subsequence of three (or more) strings. In *Proceedings of the Third Annual Symposium on Combinatorial Pattern Matching*, CPM '92, pp. 214–229, London, UK, UK. Springer-Verlag.

Islam, A. and Inkpen, D. 2008. Semantic text similarity using corpus-based word similarity and string similarity. *ACM Transactions on Knowledge Discovery from Data*, 2(2):10:1–10:25.

Islam, A. and Inkpen, D. 2009. Real-word spelling correction using google web 1t n-gram data set. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pp. 1689–1692. ACM.

Islam, A. and Inkpen, D. 2010. Near-synonym choice using a 5-gram language model. *Research in Computing Sciences*, 46:41–52.

Islam, A., **Milios, E.**, **and Kešelj, V.** 2012. Text similarity using google trigrams. In **Kosseim, L. and Inkpen, D.**, editors, *Advances in Artificial Intelligence: 25th Canadian Conference on Artificial Intelligence, Canadian AI 2012*, pp. 312–317, Berlin, Heidelberg. Springer.

Kešelj, V. and Cercone, N. 2004. CNG method with weighted voting. In *Ad-hoc Authorship Attribution Competition. Proceedings 2004 Joint International Conference of the Association for Literary and Linguistic Computing and the Association for Computers and the Humanities (ALLC/ACH 2004)*.

Kincaid, J. P., **Fishburne Jr, R. P.**, **Rogers, R. L.**, **and Chissom, B. S.** 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, Naval Technical Training Command Millington TN Research Branch.

Kövecses, Z. and Radden, G. 1998. Metonymy: Developing a cognitive linguistic view. *Cognitive Linguistics (includes Cognitive Linguistic Bibliography)*, 9(1):37–78.

LG 2009. LG600G User Guide. `https://www.manualslib.com/manual/92956/Lg-Lg600g.html#product-LG600G` [Accessed: 15-Dec-2015].

Manku, G. S., **Jain, A.**, **and Das Sarma, A.** 2007. Detecting near-duplicates for web crawling. In *Proceedings of the 16th International Conference on World Wide Web*, pp. 141–150. Association for Computer Machinery.

McNamara, D. S., **Graesser, A. C.**, **McCarthy, P. M.**, **and Cai, Z.** 2014. *Automated evaluation of text and discourse with Coh-Metrix*. Cambridge University Press, Cambridge.

Mei, J., **Kou, X.**, **Yao, Z.**, **Rau-Chaplin, A.**, **Islam, A.**, **Moh'd, A.**, **and Milios, E. E.** 2015. Efficient computation of co-occurrence based word relatedness. `DemoURL:http://ares.research.cs.dal.ca/gtm/` [Accessed: 01-Dec-2015].

Miller, G. A., **Beckwith, R.**, **Fellbaum, C.**, **Gross, D.**, **and Miller, K. J.** 1990. Introduction to WordNet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244.

Mueller, J. and Thyagarajan, A. 2016. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pp. 2786–2792. AAAI Press.

Neculoiu, P., **Versteegh, M.**, **and Rotaru, M.** 2016. Learning text similarity with Siamese recurrent networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pp. 148–157, Berlin, Germany. Association for Computational Linguistics.

Nulty, P. and Costello, F. 2009. Using lexical patterns in the google web 1t corpus to deduce semantic relations between nouns. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, SEW 09, 5863, USA. Association for Computational Linguistics.

Samsung 2011. Samsung 010505d5 cell phone user manual. `http://cellphone.manualsonline.com/manuals/mfg/samsung/010505d5.html?p=53` [Accessed: 01-Dec-2015].

Senter, R. and Smith, E. A. 1967. Automated readability index. *Wright-Patterson Air Force Base. AMRL-TR-6620*, 3.

Socher, R., **Huang, E. H.**, **Pennington, J.**, **Ng, A. Y.**, **and Manning, C. D.** 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS11, 801809. Curran Associates Inc., Red Hook, NY, USA.

Soto, A. J., **Mohammad, A.**, **Albert, A.**, **Islam, A.**, **Milios, E.**, **Doyle, M.**, **Minghim, R.**, **and Ferreira de Oliveira, M. C.** 2015. Similarity-based support for text reuse in technical writing. In *Proceedings of the 2015 ACM Symposium on Document Engineering*, DocEng '15, Lausanne, Switzerland, pp. 97–106. Association for Computer Machinery.

Sun, Y., **Qin, J.**, **and Wang, W.** 2013. Near duplicate text detection using frequency-biased signatures. In *Web Information Systems Engineering–WISE 2013*, pp. 277–291, Berlin, Heidelberg. Springer.

Vapnik, V. 2013. *The nature of statistical learning theory*. Springer Verlag, New York, 2nd edition.

Wang, W., **Mohd, A.**, **Islam, A.**, **Soto, A. J.**, **and Milios, E.** 2016. Non-uniform language detection in technical writing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1892–1900, Austin, Texas. Association for Computational Linguistics.

Wang, X., **Li, C.**, **Zheng, Z.**, **and Xu, B.** 2018. Paraphrase recognition via combination of neural classifier and keywords. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE.

Wołkowicz, J. and Kešelj, V. 2013. Evaluation of n-gram-based classification approaches on classical music corpora. In **Yust, J.**, **Wild, J.**, **and Burgoyne, J. A.**, editors, *Mathematics and Computation in Music: 4th International Conference, MCM 2013*, volume 7937, pp. 213–225, Berlin Heidelberg. Springer.

Wu, Z. and Palmer, M. 1994. Verbs semantics and lexical selection. `DemoURL:http://ws4jdemo.appspot.com/?mode=w&s1=&w1=photo&s2=&w2=video` [Accessed: 01-Dec-2015].