Automated Spelling Correction for Dutch Internet Users with Intellectual Disabilities

Leen Sevens, Tom Vanallemeersch, Ineke Schuurman, Vincent Vandeghinste, Frank Van Eynde
Centre for Computational Linguistics
KU Leuven, Belgium
firstname@ccl.kuleuven.be

Abstract

We present the first version of an automated spelling correction system for Dutch Internet users with Intellectual Disabilities (ID). The normalization of ill-formed messages is an important preprocessing step before any conventional Natural Language Processing (NLP) process can be applied. As such, we describe the effects of automated correction of Dutch ID text within the larger framework of a Text-to-Pictograph translation system. The present study consists of two main parts. First, we thoroughly analyze email messages that have been written by users with cognitive disabilities in order to gain insights on how to develop solutions that are specifically tailored to their needs. We then present a new, generally applicable approach toward context-sensitive spelling correction, based on character-level fuzzy matching techniques. The resulting system shows significant improvements, although further research is still needed.

Keywords: Automated Spelling Correction, Intellectual Disabilities, Pictograph Translation, Alternative and Augmentative Communication

1. Introduction

The Internet has influenced our daily lives in various ways. Being able to stay in touch with family and friends via email or social media websites strengthens the feeling of belonging to a community, even at distances. Therefore, not being able to access or use information technology is a major form of social exclusion. There is a dire need for digital communication interfaces that enable people with Intellectual Disabilities (ID) to contact one another.

We are developing a Text-to-Pictograph and Pictograph-to-Text translation system for the WAI-NOT[1] communication platform. WAI-NOT is a Flemish non-profit organization that gives people with severe communication disabilities the opportunity to familiarize themselves with computers, the Internet, and social media. Their safe website environment offers an email client that makes use of the pictograph translation solutions. The Text-to-Pictograph translation system (Vandeghinste et al., 2015; Sevens et al., 2015a) automatically augments written text with Beta[2] or Sclera[3] pictographs and is primarily conceived to improve the *comprehension* of textual content. The Pictograph-to-Text translation system (Sevens et al., 2015b) allows the user to insert a series of Beta or Sclera pictographs, automatically translating this image sequence into natural language text where possible, hereby facilitating the *construction* of textual content.

The Text-to-Pictograph translation system consists of various sub-processes. During the preprocessing phase, basic spelling correction (see section 5.1.) is applied, as some users have the ability to write short messages without having to rely on the pictograph selection menu. However, these messages often contain severe spelling errors. While it is important to encourage people with ID to write their own messages if they have the ability to do so, the resulting text may pose several problems. First, even if the receivers of the ill-formed messages are (to some extent) able to read written text, they might not be able to understand these messages because of all these mistakes. Secondly, as noted by Sproat et al. (2001), text normalization is recommended before applying a more conventional Natural Language Processing (NLP) process. The Text-to-Pictograph translation tool, which translates the email into pictographs for people who have reading difficulties, may retrieve wrong pictographs or no pictographs at all for erroneously written words. Vandeghinste et al. (2015) evaluated the Text-to-Pictograph translation system and showed that there is clearly room for further improvement in the automated spelling correction process, as the scores for the upper bound (manual spelling correction) were significantly better than the scores for the basic, automated spelling correction process (see Figure 1).

Figure 1: Example of an erroneous Text-to-Beta translation

We present the first version of an automated spelling corrector that is specifically tailored to users with ID. After a discussion of related work (section 2.), we thoroughly analyze tweets and messages sent with the WAI-NOT system and show that users with ID make more and different spelling mistakes than users who do not have cognitive disabilities (section 3.). We then proceed to describe the system architecture. On the one hand, the system consists of a variant generation and filtering step that is partially based on discovering phonetic similarities. On the other hand, we apply character-based fuzzy matching as a novel approach to

[1] http://www.wai-not.be/
[2] https://www.betasymbols.com/
[3] http://www.sclera.be/

context-sensitive spelling correction (section 4.). Our evaluations show that improvements over the baseline in the Text-to-Pictograph translation tool were made (section 5.). Finally, we conclude and describe future work (section 6.).

2. Related work

The rapid dissemination of electronic communication devices has triggered the emergence of new forms of written texts (Kobus et al., 2008). Microtext, or chatspeak-style text, such as tweets or text messages, is characterized by the use of abbreviations, misspellings, phonetic text, colloquial and ungrammatical language, lack of punctuation, and inconsistent capitalization, among other things (De Clercq et al., 2013). Several linguistic models and algorithms have been proposed to deal with errors. We will focus on three popular models for the correction of microtext in particular, as proposed by Kobus et al. (2008): the Noisy Channel or Spell Checking model, the Machine Translation model, and the Speech Recognition model.

The concept behind the *Noisy Channel* model, also called the *Spell Checking* model, is to consider a spelling error as a noisy signal that has been distorted somehow during transmission (Bassil and Alwani, 2012). The Noisy Channel model applies spelling correction on a word-per-word basis and is often limited to the correction of Out of Vocabulary (OOV) words. It relies on orthographic or phonemic surface similarity between two forms. Examples of the Noisy Channel approach for spelling correction are the rule-based system developed by de Neef and Fessard (2007), the system incorporating phonetic information developed by Toutanova and Moore (2002), and the Hidden Markov Model developed by Choudhury et al. (2007), which handles both graphemic variants and phonetic plays. Beaufort et al. (2010) note that the Noisy Channel model places excessive confidence in word boundaries.

The *Machine Translation* (MT) model considers the ill-formed text as the source language, and the correct text as the target language. Aw et al. (2006), for example, use phrase-based MT to tackle the spelling correction problem. It should be noted, though, that it is labor-intensive to construct an annotated corpus to cover ill-formed words and context-appropriate corrections (Han and Baldwin, 2011), especially since the lexical creativity in microtext is difficult to capture. Another issue is the fact that Statistical Machine Translation allows to handle many-to-many correspondences and applies methods to model the possible mismatch in word order (Kobus et al., 2008), while the normalization task is almost deterministic (Beaufort et al., 2010), with no change in word order. De Clercq et al. (2013) implement an MT-based approach and describe the first (and to our knowledge, only) proof-of-concept system for Dutch user-generated content normalization, but they do not consider users with ID.

The *Speech Recognition model* converts the input string into a phone lattice, followed by the creation of a word-based lattice using phoneme-to-grapheme rules, after which a language model is applied and a best-path algorithm is used (Beaufort et al., 2010). An example of this method is presented by Kobus et al. (2008). Han and Baldwin (2011) identify normalization candidates for an OOV word by decoding the pronunciation of all in-vocabulary words and retrieving all words that lie within a threshold character edit distance between the OOV word's pronunciation and the dictionary words' pronunciation.

Our spelling correction system can be considered as a combination of all three approaches, while also introducing new ideas. Although not only OOV words are considered, spelling variants are generated for individual tokens (Noisy Channel model). More specifically, these variants are generated (in the first place) by considering the ill-formed word as a result of phonetic confusion (Speech Recognition model). Finally, we match our new spelling hypotheses against a target language corpus of correctly written text (Machine Translation model). The system does not require large amounts of annotated data.

3. Error distribution: Comparison with tweets

Whenever microtext is considered in the literature, its description is often (if not always) limited to the analysis of SMS messages and tweets. Spelling correction for microtext is a young domain of research, due to the recent boom of social media websites, and its focus lies on users who do not necessarily have a cognitive disability. However, many people with cognitive disabilities resort to specialized communication platforms and apps, such as the WAI-NOT environment. The fact that the spelling correction tool possibly needs to deal with a completely new and different type of microtext should not be ignored. In order to verify this, we compared tweets written by people who supposedly do not have a cognitive disability with emails that were sent with the WAI-NOT email client.

	# OOV	# RWE	# Words	% Errors
WAI-NOT	481	183	8077	8.2%
Tweets	182	88	10964	2.5%

Table 1: Total amount of misspelled tokens. OOV = Out-of-Vocabulary tokens; RWE = Real-word errors

We selected a total of 1000 subsequent tweets from the Dutch Twitter feed, having excluded those messages that were not personal, such as news articles or advertisements. Additionally, a total of 1000 random WAI-NOT emails were selected after having thrown away 49 completely unreadable messages and 330 messages that consisted of pictographs only. We manually corrected all tweets and email messages, while analyzing the different types of errors that were made.[4]

Generally speaking (see Table 1), many more errors can be found in the WAI-NOT messages (8.2%) than in tweets (2.5%). Both OOV words and real-word errors were considered.

As shown in Table 2, the majority (52.1%) of spelling mistakes that are made by people with ID is caused by phonetic confusion, defined here as the orthographic approximation of a word's pronunciation (such as *wiekent* for *weekend*). Although this phenonemon can also be observed in tweets

[4]The corrected tweets and WAI-NOT emails are available on request. The emails may only be used for research purposes.

	Total # misspelled	# PW	% PW
WAI-NOT	664	346	52.1%
Tweets	270	95	35.2%

Table 2: Total amount of misspelled words that are a phonetic approximation of the correct word. PW = Phonetic words

(35.2%), Twitter users' phonetic spellings tend to be much more systematic. They usually concern deliberate mistakes in an attempt to mimic speech (such as the final *t* deletion in *da* or *nie* for *dat "that"* and *niet "not"*), or recurrent grammatical mistakes (such as *jou "you"* versus *jouw "your"* or *gebeurt "happens"* versus *gebeurd "happened"*). Han and Baldwin (2011) note that ill-formedness in regular messages is often intentional, whether due to the desire to save characters or keystrokes, due to the wish to belong to a social group, or due to convention. Phonetic mistakes in WAI-NOT messages are most likely undeliberate mistakes in an attempt to write a correct piece of text, and are therefore much more diverse. This idea is reinforced by the fact that a large part of the analyzed messages were addressed at teachers or caregivers, for whom one might do a deliberate effort.

	LD	# Words	Percentage
WAI-NOT			
	1	479	72.1%
	2	128	19.3%
	3	44	6.6%
	4	9	1.4%
	5	2	0.3%
	6	2	0.3%
Tweets			
	1	166	61.5%
	2	66	24.4%
	3	19	7%
	4	7	2.6%
	5	4	1.5%
	6	4	1.5%
	7	2	0.7%
	8	1	0.4%
	12	1	0.4%

Table 3: Overview of total amount of character operations required per erroneously spelled word. LD = Levenshtein distance

As an additional error measure, we counted the number of insertions, deletions, and substitutions needed to get from the original messages to their corrected counterparts (see Table 3). On the average, messages in WAI-NOT require 1.4 operations per erroneously spelled word, while tweets require 1.7 operations. This difference can be explained as follows. Relatively speaking, Twitter users are more likely to delete characters (75.6% of all required character operations are insertions) than WAI-NOT users (48.6%). This observation is most likely due to the 140-character limit for tweets or the wish to belong to a social group. Examples of deliberate abbreviations in tweets that require many character insertions are *wrschnlk* for *waarschijnlijk "probably"* and *mssch* for *misschien "maybe"*.

	# FL	# PN	# EN	# AB
WAI-NOT	10	0	6	0
Tweets	9	0	72	59

Table 4: Other factors that should be taken into consideration. FL = Flooding; PN = Phonetic numbers; EN = English words; AB = Abbreviations

There are other problems related to spelling errors that may need correction (see Table 4). Flooding, the constant repetition of one character, which occurs when emphasis is given by the user (such as *noooooo* or *coooool*), can be found in both genres, while we did not encounter any examples of numbers encoding phonetic values (such as *m8* for *mate* in English). English words were hardly used in the Dutch WAI-NOT messages (with the exception of *I love you*). Abbreviations (such as *m.b.t. "w.r.t."* for *met betrekking tot "with respect to"*) did not occur in these messages at all. Therefore, as long as our system focuses on users with ID, it should not be dealing with foreign language detection or abbreviation solving.

From this analysis, it can be concluded that text written by people with ID is indeed a different kind of microtext. Not only does it contain more errors and phonetic approximations, common abbreviations are lacking, and the users barely use any English words for which a Dutch alternative is available.

Spelling errors made by children who are still learning how to spell and people with Alzheimer's disease might be very similar to text written by people with ID. [5] This hypothesis will have to be tested.

4. System architecture

We describe our prototype version of a spelling corrector that is specifically tailored to Dutch text written by people with ID (see Figure 2). In the first phase, the input text to be corrected undergoes a number of preprocessing steps (section 4.1.). Next, spelling variants are generated for all OOV tokens and infrequent real words (section 4.2.); a trigram language model is used to narrow down the final amount of possibilities. The third step consists of using a character-based fuzzy matching technique for finding the best combination of spelling variants and, additionally, performing new character substitutions when a strong context match is found (section 4.3.). Finally, we describe the parameter tuning process (section 4.4.).

4.1. Preprocessing

The input text undergoes a number of preprocessing steps. The *word builder* module (Vandeghinste, 2002) takes every two adjacent tokens and checks whether they can be put together in order to form a real word. The word builder parameters (see section 4.4.) set different threshold frequencies for the (non-)acceptance of the newly created compound word.

The rule-based *tokenizer* splits the punctuation signs from the words, as the variant generation module works on the token level. Given that the hyphen/dash and the apostrophe

[5] We thank one of the anonymous reviewers for this suggestion.

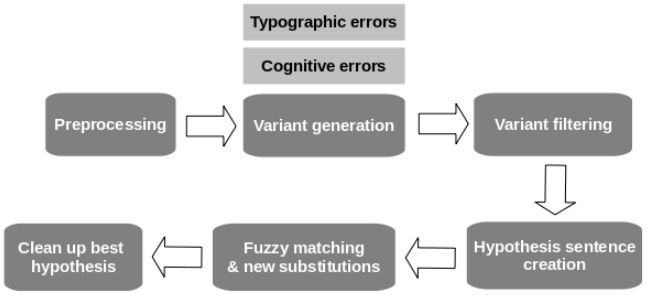

Figure 2: System architecture

often belong to the word, they are not dealt with by the tokenization process.

Although most messages sent by the users only contain one sentence, *sentence detection* is applied. Segmentation is based on full stops.

In the next step, upper-case letters are converted to lower-case letters. Names keep their capital first letter, so they will not be involved in the spelling correction process, as long as the name can be found in a database of first names. [6]

The constant repetition of one character or *flooding* is tackled by reducing any repeated sequence of characters to two characters.

Finally, we created a very small dictionary containing popular greetings (such as *hey*) for tokens that will have to be left out of the correction process.

4.2. Variant generation and filtering

Spelling variants are generated for all OOV tokens and infrequent real words according to a reference corpus. Our variant generation process focuses on phonetic, i.e. cognitive errors (section 4.2.1.). If no variants are generated, the system checks for basic typographic errors (section 4.2.2.). The final amount of variants is narrowed down by a trigram language lookup before proceeding to the next step (section 4.2.3.).

4.2.1. Generating variants for cognitive errors

Cognitive errors occur when the writer does not know how to spell a word, and often rely on the identical pronunciation of words (Toutanova and Moore, 2002). As shown by the error distribution (see Table 2), phonetic confusion causes the majority of spelling errors that are made by the target group.

Building conversion rules The approach described in this section is partially inspired by the finite-state framework for normalizing SMS messages developed by Beaufort et al. (2010).

First, we manually correct 1000 sentences written by WAI-NOT users.

We then align the uncorrected and corrected sentences on the character level, using Levenshtein Distance Alignment. [7] This metric (Levenshtein, 1966) computes the edit distance of two strings by measuring the minimum number of operations (substitutions, insertions, deletions) required to transform one string into the other. Delimiters, such as commas and spaces, are also aligned. Missing characters on either side of the alignment are indicated by inserting a hyphen (-).

In the next step, we create token pairs. A token pair is retrieved when the same delimiter is found at the same location in both the source/uncorrected and target/corrected language character string. From these token pairs, we extract all possible character 4-grams on the source language side and the characters they align with on the target language side. This process is repeated for three, two, and one character(s).[8]

Having obtained all character 4-gram, trigram, bigram, and unigram alignments, probabilities are estimated: For every character n-gram on the source side, we calculate the likelihood of obtaining a particular character sequence on the target side. The sequence obtained is usually identical, but sometimes different. For example, the character trigram "int" on the source/uncorrected language side remained "int" in 91% of the cases, but had been corrected into "ind" in the remainder of the sentences.

For the construction of our final rule set, we retain only those cases where we observe a 1% to 100% probability of changing a particular character n-gram into a different n-gram. The idea behind this is that rarely occurring alternations might actually have a typographic rather than a phonetic origin. By contrast, more commonly occurring mistakes are most likely due to phonetic confusion. This idea is also reflected in the final version of our inventory. For instance, the written sequences "int" and "ind", "pra" and "praa", "orie" and "orry" can indeed be pronounced the same way.

This inventory of commonly appearing alternations allows us to build a system of character rewrite rules, in which character 4-gram rules overrule trigram rules, trigram rules overrule bigram rules, and so on.

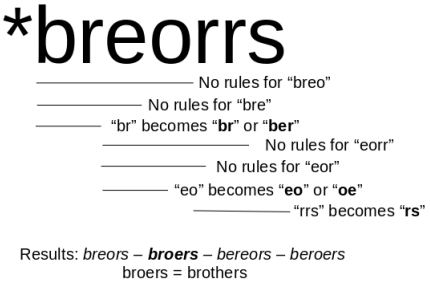

Figure 3: Example of how the conversion rules are applied

Applying conversion rules For every non-word and every real word that has a frequency lower than the *real word minimum frequency threshold* (see section 4.4.) in our frequency list,[9] the conversion rules are applied (see Figure 3).

[6] http://www.quietaffiliate.com/free-first-name-and-last-name-databases-csv-and-sql/

[7] http://rosettacode.org/wiki/Levenshtein_distance/Alignment

[8] In future work, we will evaluate whether five- or six-character pairs make the variant generation process more robust.

[9] The frequency list contains roughly eighty million words of Belgian Dutch newspaper text.

A four-character window slides over the token, starting with the first four characters of the token, and checks if a 4-gram rule can be found for that particular sequence. If a rule is found, all the conversion outputs (including the original sequence) are stored and the system will proceed to check the next four characters of the token. If no rule is found, the system backs off to the first three characters of the token and attempts to find a trigram rule for that sequence. If, even at the one-character level, no rules could be found, the original character is retained and the system proceeds to find rules for the next four-character sequence.

In the end, all conversion outputs are concatenated and both non-words and real words may have been formed. If a real word is formed with a frequency higher than the *variant frequency* (see section 4.4.), it will be retained as a variant for that token.

Note that our approach, although phonetically similar variants are generated, does not yet decode the pronunciation of words into phonemes.

4.2.2. Generating variants for typographic errors

Typographic errors are mostly related to the keyboard (Toutanova and Moore, 2002). If in the previous step no variants were generated for a non-word or a word that has a frequency lower than the *real word minimum frequency threshold* (see section 4.4.), we apply basic typographic error correction principles. We generate variants based on five different operations.

The first operation is the word splitting module. This is an insertion module for one space character. The system checks whether the erroneous or infrequent token can be split into two parts at any position. Frequency thresholds are determined by parameters (see section 4.4.).

The next operations are one-character deletion, insertion, substitution, or adjacent transposition at every position of the token. If a real word is formed with a frequency higher than the *variant frequency* (see section 4.4.), it will be retained as a variant for that token. If the original token was a real word, then that word will always be retained as one of the variants.

4.2.3. Filtering the variants

It is possible that, at this point, the system has generated multiple variants for a single erroneous word or infrequent real word. Especially when considering very short words, many real-word alternatives can often be created. The filter module narrows down the total amount of possibilities before proceeding to the next step. A trigram language model trained on a very large corpus (a combination of the Dutch part of Europarl, Corpus Gesproken Nederlands, Cross-Language Evaluation Forum, DGT-Translation Memory, and Wikipedia) is used to check whether the variant ever occurs within its context in the language model. As the token's direct context may also contain variants, all possible trigrams are checked until a match is found. If a match is found, the variant will be retained. If no trigram matches are found for any of the variants because the context does not provide enough information, all variants are retained.

4.3. Character-based fuzzy matching

The combination of all variants described above leads to the creation of a number of potentially correct sentences. Each one of these sentences is a hypothesis, one of which will receive the highest score through fuzzy matching. Fuzzy matching techniques, which have been developed in the context of translation memories (databases with source sentences and their translations used by professional translators), allow to find strings in a corpus that approximately (rather than exactly) match a string. We are applying this technique to a monolingual corpus. In the development of the spelling correction tool, we explored the new possibility of applying fuzzy matching techniques at the character level.

Each one of the hypotheses is split into individual characters. The space is replaced by a dummy character, the % sign, and should also be recognized as a character. As our corpus, we use the Spoken Dutch Corpus (Corpus Gesproken Nederlands, (Oostdijk et al., 2002)) since spoken language better reflects the language used in user-generated content (De Clercq et al., 2013). This corpus is also split into individual characters. During the fuzzy matching process, we use a filter called *approximate query coverage* (Vanallemeersch and Vandeghinste, 2015). Its purpose is to select candidate sentences in a corpus which are likely to reach a minimal matching threshold when submitting them to a fuzzy matching metric, in order to increase the speed of matching. Candidate sentences share one or more n-grams of a minimal length with the input hypothesis, and share enough n-grams with the input hypothesis to cover the latter sufficiently (according to some threshold). In our spelling correction model, the unigram is one single character. A very efficient search for sentences sharing n-grams with the input hypothesis can be done by means of a suffix array (Manber and Myers, 1993).[10]

A hypothesis that shares many and long character n-grams with candidate sentences from the corpus has a bigger likelihood of becoming the winning hypothesis than one that shares only few and short n-grams.[11]

The context-sensitivity of the fuzzy matching method allows us to deal with additional spelling errors, even if the correct variant had not been generated in the variant generation phase. If a high-scoring corpus match is found for two strings of characters and there is a gap of maximum three characters between those strings in both the corpus and the original hypothesis, those characters will be replaced in the hypothesis. For example, the hypothesis *kan je dat misschien nog aan jou moeder vragen "maybe you can ask your mother"* contains a common spelling mistake in Dutch. *Jou* is a personal pronoun, while *jouw* is a possessive pronoun. *Jouw* would be correct here. However, no variants were generated for *jou* in the variant generation process, as it is a highly frequent word. One of the matching strings in the corpus is *nu moeten we het nog aan jouw moeder vragen "now we still need to ask your mother"*. Looking at this

[10] We used the SALM toolkit (Zhang and Vogel, 2006) for building and consulting suffix arrays.

[11] For sake of brevity, we refer to Vanallemeersch and Vandeghinste (2015).

sentence and the hypothesis, there is a character overlap between *% n o g % a a n % j o u* and *% m o e d e r % v r a g e n*. The system finds the character *w* in the corpus, surrounded by the two substrings (a one-character gap). This character is inserted in the original sentence, hereby fixing the spelling mistake. We will include the maximal gap width as one of the parameters in future work.
The winning hypothesis is cleaned up. The spaces between the characters are removed, the % signs are converted into spaces and the first letter is capitalized.

4.4. System parameters

The system contains a number of parameters, which were tuned.
There are two *word builder penalties*. The first penalty concerns the frequency of the separate parts of the (potentially in-vocabulary) compound token. If both parts are real words and their frequency is high enough (post-tuning value: 120), they won't have to pass through the word builder module. The other penalty is the minimum frequency required to accept a newly built real word (post-tuning value: 210).
Similarly, there are two *word splitter penalties*. The first penalty concerns the minimum frequency of the token. If the frequency of this token is high enough (post-tuning value: 1760), it will not have to pass through the word splitter module. However, if the frequency is not high enough or if the token turns out to be a non-word, the system will attempt to split the token into two real word parts. The second penalty sets a minimum frequency for those two words (post-tuning value: 1680). If the frequency is high enough, the original word will be split.
The *real word minimum frequency threshold* determines how common a correctly spelled word should be in order to avoid going through the spelling variant generation process (post-tuning value: 100).
When real word variants are generated for a token, they need to have a minimum frequency, the *variant frequency*, in order to be accepted as a variant (post-tuning value: 220).
There are also three fuzzy matching penalties. The *n-gram penalty* decides on the minimum amount of contiguous characters that should occur as a sequence in the corpus sentence (post-tuning value: 8). The *minimum score penalty* sets the minimum matching score needed to retrieve a corpus sentence (post-tuning value: 0.2). Finally, the *highest frequency threshold* decides that, if a certain n-gram has a very high frequency, the fuzzy matching system will ignore it for fuzzy matching, for reasons of speed (post-tuning value: 100).
We created a tuning corpus by manually correcting 200 new WAI-NOT messages. We used the local hill climber algorithm as described in Vandeghinste et al. (2015), which varies the parameter values when running the spelling corrector script on the test set. The BLEU metric (Papineni et al., 2002) was used as an indicator of relative improvement. BLEU is a precision-oriented metric which compares the system output to one or more reference translations, by counting how many n-grams overlap, and correcting for brevity. We ran five trials of a local hill climbing algorithm. We did this until BLEU converged onto a fixed score after several thousands of iterations. Each trial was run with random initialization values, and varied the values between certain boundaries in order to cover different areas of the search space. From these trials, we took the best scoring parameter values.

	BLEU	NIST	WER	# CO
No corrector	0.64	8.62	12.37	699
Old corrector	0.62	8.07	19.51	816
New corrector	0.84	10.49	7.57	238

Table 5: Automated evaluations on 300 email messages. CO = Number of character operations

	Old	New
# Justified corrections of erroneous words	41	145
# Unjustified corrections of erroneous words	74	29
# Non-corrected erroneous words (# Real)	157 (76)	98 (70)
# Inappropriate changes to correct words	16	0

Table 6: Analysis of how the systems deal with erroneous words

5. Evaluation

We present the results of our evaluations. Section 5.1. evaluates the system on an unseen test set of WAI-NOT messages and compares its corrections with the corrections made by the system that was originally developed for the Text-to-Pictograph translation tool. Section 5.2. evaluates the system within the context of the Text-to-Pictograph translation pipeline.

5.1. Intrinsic evaluation

After having filtered unreadable messages and messages that consisted of pictographs only, we took 300 random emails from the WAI-NOT corpus and manually corrected them to the best of our ability. Our baseline is the original set of uncorrected messages. We also show the result of applying the spelling correction system that was originally developed for the Text-to-Pictograph translation tool. The original system applies one-character substitutions, deletions, and insertions to generate a list of variants and selects the most frequent variant according to the frequency list (see section 4.2.). This context-insensitive approach is compared to the output generated by the new system.
Table 5 shows the word-based BLEU, NIST (Doddington, 2002), and Word Error Rate (WER) scores. NIST is similar to BLEU, but gives less credit to high frequency non-informative n-grams. WER counts the number of words that are incorrect with respect to the reference translation(s) and is very well suited for the evaluation of NLP tasks where the input and output strings are closely related. We also automatically calculated the amount of character operations needed in order to get to the reference correction.
As shown in Table 5, the original spell checker does more things wrong than right. However, significant improvements were made using the new spell checker.
Table 6 presents an analysis of how both the original and the new system deal with erroneous words in the email messages. The new system is able to detect more erroneous forms (as the original system was limited to OOV

Condition	Precision	With proper names Recall	With proper names F-Score	Without proper names Recall	Without proper names F-Score
Sclera					
Baseline	89.2%	86.2%	87.7%	85.2%	87.2%
New system	92.6%	89.1%	90.8%	88.2%	90.3%
Rel.improv.	*3.7%*	*3.3%*	*3.5%*	*3.6%*	*3.6%*
Beta					
Baseline	85.9%	89.5%	87.6%	88.7%	87.3%
New system	89.8%	91.5%	90.6%	90.8%	90.3%
Rel. improv.	*4.5%*	*2.3%*	*3.4%*	*2.4%*	*3.5%*

Table 7: Manual evaluation of the Text2Picto translation engine

errors) and finds the appropriate correction for 83.3% of these words, while the old system only manages to correct 35.6% of the detected words. 71.4% of the unretrieved words in the new system are highly frequent real words. Many of these real-word errors can be contributed to grammatical confusion, such as the difference between *jou* and *jouw* (see section 4.3.). These errors lead us into the domain of grammar correction and are currently beyond the scope of our work.

The old system corrects some words that should not have been corrected in the first place. These erroneous corrections mostly concern common greetings and proper names that are not included in our list of first names and for which a low-frequency variant is generated. These problems are solved in the new system by the introduction of the small greetings dictionary and the fact that low-frequency variants will not be proposed for an unknown name.

Comparing our system with other systems is difficult, as they do not consider text written by users with ID (and most tools focus on English text). De Clercq et al. (2013), who created the first and only normalization tool for Dutch microtext, admit that words requiring different types of operations are difficult for their system, while our approach allows for multiple (phonetic) substitutions within a single word. De Clercq et al. showed that their system is best at resolving smaller words requiring only one or two insertions, while especially phonetic problems turned out to be an important obstacle for them. As our system is made for users with ID, phonetic alternations are the core of the variant generation process.

5.2. Extrinsic evaluation

We also manually evaluated the effects of the spelling correction system within the larger context of the Text-to-Pictograph translation tool. The baseline, which uses the old spelling corrector, is the system as described by Vandeghinste et al. (2015). The new system implements the spelling corrector as presented in this paper. We used the same systematic and objective approach to manual evaluation as Vandeghinste et al. (2015).

The evaluation set of 50 Dutch messages that have been sent with the WAI-NOT email system consists of 84 sentences (980 words). These were all translated into a sequence of Sclera or Beta pictographs using the Text-to-Pictograph translation tool. [12]

[12]Our gold standards are made available on request.

We have performed a manual annotation with one judge, who removed untranslated words that were considered not to contribute to the content. This allowed calculating the recall. For each of the translated words, she judged whether the pictograph generated was the correct pictograph, in order to calculate precision. Results are presented in Table 7. As proper names occur frequently in e-mail messages, we have calculated recall and F-score with and without proper names, in the latter case removing all proper names from the output. In the case where proper names are included, they are not converted into pictographs. Precision remains the same in both cases. In the WAI-NOT environment, proper names occurring in the contact lists of the users are converted into the pictures attached to these profiles, resulting in more personalized messages.

Figure 4: Example of a correct Text-to-Beta translation

An increase in precision and recall was obtained for both the Beta and the Sclera condition. Examples of erroneous words that previously could not be translated into pictographs are *grapeg* for *grappig "funny"*, *ikhoop* for *ik hoop "I hope"* and *heeeeel* for *heel "very"*. Examples of erroneous words that previously led to an erroneous pictograph translation were *wiekent* for *weekend "weekend"*, which was corrected into *wieken "wings"* (and translated into a pictograph showing a bird's wings, see Figure 1), and *moelijke* for *moeilijke "difficult"*, which was corrected into *mogelijke "possible"* (and translated into a pictograph showing the verb "can"). The new spelling corrector has managed to tackle these issues.

6. Conclusion and future work

We described the first version of an automated spelling corrector for Dutch text written by people with ID. The system can be extended to other languages, provided some corrected data is available in order to infer new phonetic rules for the variant generation step. Nevertheless, the current approach does not require massive amounts of training data.

The results show that the system already improves over the baseline, but there is ample room for enhancement.

In the first place, the variant generation process is not yet able to correct tokens in which both elements of phonetic confusion and typographic errors are present. While these are currently two completely unrelated steps within the variant generation process, the ideal scenario would be to find an efficient way to combine them without overgenerating. Additionally, phone lattices should be introduced in order to go deeper than purely orthographic variation patterns.

For the fuzzy matching step, we will add more and/or different corpora to the corpus and evaluate their influence on the system's performance. These corpora should be exempt from spelling errors and share as many characteristics with informal text or oral conversations as possible. The corpora should contain plenty of first-person and second-person forms.

Finally, we should consider performing a grammar check during the spelling correction process in order to detect real-word errors that are left out of the variant generation process because of their high frequency.

7. Acknowledgements

We would like to thank IWT and the European Commission's Competitiveness and Innovation Programme for funding Leen Sevens' doctoral research and the Able-To-Include project, which allows further development and valorisation of the pictograph translation tools.

We also thank the people from WAI-NOT for their valuable feedback and the integration of our tools on their website.

8. Bibliography

Aw, A., Zhang, M., Xiao, J., and Su, J. (2006). A phrase-based statistical model for SMS text normalization. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 33–40, Sydney, Australia. Association for Computational Linguistics.

Bassil, Y. and Alwani, M. (2012). Context-sensitive Spelling Correction Using Google Web 1T 5-Gram Information. *Computer and Information Science*, 5(3).

Beaufort, R., Roekhaut, S., Cougnon, L.-A., and Fairon, C. (2010). A Hybrid Rule/Model-based Finite-state Framework for Normalizing SMS Messages. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*, pages 770–779, Uppsala, Sweden. Association for Computational Linguistics.

Choudhury, M., Saraf, R., Jain, V., Mukherjee, A., Sarkar, S., and Basu, A. (2007). Investigation and Modeling of the Structure of Texting Language. *International Journal of Document Analysis and Retrieval: Special Issue on Analytics of Noisy Text*, 10:157–174.

De Clercq, O., Schulz, S., Desmet, B., Lefever, E., and Hoste, V. (2013). Normalization of Dutch User-Generated Content. In *Proceedings of the 9th International Conference on Recent Advances in Natural Language Processing (RANLP 2013)*, pages 179–188, Hissar, Bulgaria.

de Neef, E. G. and Fessard, S. (2007). Evaluation d'un système de transcription de SMS. In *Actes du 26e Colloque international Lexique Grammaire*, Bonifacio, France.

Doddington, G. (2002). Automatic Evaluation of Machine Translation Quality Using N-gram Co-occurrence Statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, pages 138–145, San Diego, California, USA.

Han, B. and Baldwin, T. (2011). Lexical Normalisation of Short Text Messages: Makn Sens a #Twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1 (ACL-HLT 2011)*, pages 368–378, Portland, Oregon, USA. Association for Computational Linguistics.

Kobus, C., Yvon, F., and Damnati, G. (2008). Normalizing SMS: Are Two Metaphors Better Than One? In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1 (COLING '08)*, pages 441–448, Manchester, UK. Association for Computational Linguistics.

Levenshtein, V. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10.

Manber, U. and Myers, G. (1993). Suffix Arrays: A New Method for On-line String Searches. *SIAM Journal on Computing*, 22:935–948.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL 2002)*, pages 311–318, Philadelphia, PA, USA. Association for Computational Linguistics.

Sevens, L., Vandeghinste, V., Schuurman, I., and Van Eynde, F. (2015a). Extending a Dutch Text-to-Pictograph Converter to English and Spanish. In *Proceedings of 6th Workshop on Speech and Language Processing for Assistive Technologies (SLPAT 2015)*, Dresden, Germany.

Sevens, L., Vandeghinste, V., Schuurman, I., and Van Eynde, F. (2015b). Natural Language Generation from Pictographs. In *Proceedings of 15th European Workshop on Natural Language Generation (ENLG 2015)*, pages 71–75, Brighton, UK. Association for Computational Linguistics.

Sproat, R., Black, A. W., Chen, S., Kumar, S., Ostendorf, M., and Richards, C. (2001). Normalization of Non-standard Words. *Computer Speech and Language*, 15(3):287–333.

Toutanova, K. and Moore, R. C. (2002). Pronunciation Modeling for Improved Spelling Correction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL 2002)*, pages 144–151, Philadelphia, PA, USA. Association for Computational Linguistics.

Vanallemeersch, T. and Vandeghinste, V. (2015). Assessing Linguistically Aware Fuzzy Matching in Translation Memories. In *Proceedings of the 18th Annual Confer-*

ence of the European Association for Machine Translation (EAMT 2015), Antalya, Turkey.

Vandeghinste, V., Schuurman, I., Sevens, L., and Van Eynde, F. (2015). Translating Text into Pictographs. *Natural Language Engineering*, pages 1–28.

Vandeghinste, V. (2002). Lexicon Optimization: Maximizing Lexical Coverage in Speech Recognition through Automated Compounding. In *Proceedings of the Third International Confererence on Language Resources and Evaluation (LREC 2002)*, Las Palmas, Spain.

9. Language Resource references

Oostdijk, N., Goedertier, W., Eynde, F. V., Boves, L., Martens, J.-P., Moortgat, M., and Baayen, H. (2002). Experiences from the Spoken Dutch Corpus Project. In *the Third International Confererence on Language Resources and Evaluation (LREC 2002)*, pages 340–347, Las Palmas, Spain.

Zhang, Y. and Vogel, S. (2006). Suffix Array and Its Applications in Empirical Natural Language Processing. Technical report, Carnegie Mellon University, Pittsburgh, PA, USA.