


Normalization of shorthand forms in French text messages using word embedding and machine translation

Book Chapter**Author(s):**

Ghoshal, Parijat; [Rao, Susie Xi](#) 

Publication date:

2019

Permanent link:

<https://doi.org/10.3929/ethz-b-000388689>

Rights / license:

[In Copyright - Non-Commercial Use Permitted](#)

Originally published in:

Studies in Corpus Linguistics 90, <https://doi.org/10.1075/scl.90.17gho>

Normalization of Shorthand Forms in French Text Messages Using Word Embedding and Machine Translation

Parijat Ghoshal, Xi Rao¹

1 Introduction

Text messages are a form of computer-mediated communication (CMC). They have raised new discussions in linguistics about their spoken or written usage and their online or offline synchronicity. As the following quote points out, it remains a point of contention whether text messages are similar to spoken language: “The degree to which CMC can be associated with speech and/or writing [...] often depends on its level of synchronicity” (Van Compernelle 2010: 448).

Koch and Oesterreicher (2001: 3) have also mentioned two probable factors, namely concept and medium, through the interaction of which a genre can be considered “oral” or “written”. To summarize, we can expect variability from standard French if the language is used in close, online, informal and speech-like communication.

¹ Both authors contribute equally to the project. Authors are listed in the alphabetical order. The authors appreciate the valuable comments and support from Prof. Dr. Martin Volk and Natalia Korchagina from the Institute of Computational Linguistics at the University of Zurich and thank Heath Gordon and the anonymous reviewers for proofreading. All remaining errors are ours.

In our work, we approach the issue of normalizing shorthand forms in Belgian French text messages (Fairon 2006) using two approaches. In the first approach, we see normalization as a machine translation task and implement a statistical character-based translation system to generate character mappings from the non-standard text messages to standard French. In our second approach, we use bilingual word embeddings to capture similar items that can appear in the same contexts for non-standard and standard French. Both approaches require the use of parallel corpora for training purposes.

The paper is structured as follows. First, we give a brief overview of the endeavors in normalizing non-standard languages. Then we explain word embeddings and their state-of-the-art applications. Section 3 describes the corpus and preprocessing steps for our experiments. In Section 4, we describe our methodologies, tools and experiment setups. An extensive result analysis is conducted in Section 5, followed by the conclusion and an outlook for our future work in Sections 6 and 7, respectively.

2 Previous Work

Beaufort et al. (2010: 770) provide a brief overview of the variability that can occur in text messages, such as phonetic plays (“*mer6*” read “*merci*”, “thanks”), phonetic transcription (“*ki*” read “*qui*”, “who”), consonant skeletons (“*ktv*” read “*que tu veux*”, “as you wish”), misapplied, missing or

incorrect separators (“*ojourdhui*” read “*aujourd’hui*”, “today”), etc². The reasons for these deviations may be the constraints of the medium, which only allows 140 bytes per text message, as well as the layout of phone keypads.

Three methodologies have been applied to normalize text messages, namely, the hidden Markov model (HMM, aka noisy channel model), automatic speech recognition (ASR), and statistical machine translation (SMT).

Choudhury et al. (2007: 63-70) authored the earliest work on texting language (TL). They formally and linguistically analyzed the nature and type of TL. For their work, they used a word-aligned corpus of 1,000 English text messages (with 20,000 tokens), which was created manually. HMM was applied to decode the TL English to standard English. Essentially, HMM maps hidden phonetic information to characters. Their approach worked fairly well for unseen words; however, it could not handle self-looping (e.g. “*sooooo*” for “so”) or transposition (“*aks*” for “ask”). They evaluated their methodology on 1,228 unique tokens with an accuracy of 80%.

A hybrid approach, applying phrase-based SMT and ASR to normalize non-standard forms in text messages, was used by Kobus et al. (2008). The logic behind their use of ASR was that SMS can be seen as “an alphabetic/syllabic approximation of a phonetic form” (Kobus et al. 2008: 443). Phrase-based SMT was used to generate the mapping from the

² These examples are from the Belgian corpus.

original form to the standard, while ASR was able to find the most probable transformation from graphemes (original form) then to phonemes to graphemes (standard form). Dictionaries of grapheme-phoneme-mappings were created. This paper is of importance to our work, because it is an early work that tackled the issue of normalization using the Belgian sms4science³ (created in 2004) corpus. 3,000 unseen messages were tested against this hybrid approach with a word error rate (WER) of 11%. The authors did not report the number of unique original tokens in the test set and investigated both word (e.g. “d” to “de”) and phrase (e.g. “*oublié2tdir*” to “*oublié de te dire*”) normalization.

Yvon (2010) built on the approaches from Kobus et al. (2008) using the same samples from the Belgian French corpus, and claimed that weighted finite state machines (FSM) worked better than phrased-based SMT. An additional dictionary of abbreviation-standard forms was used before FSM. Finally, a statistical language model (3-gram) was used to enhance the results of FSM. The system performance in Yvon (2010) did not outperform that in Kobus et al. (2008).

Another work on the Belgian corpus was Beaufort et al. (2011), which performed a 10-fold cross validation. The ultimate goal of this work was to generate character alignments without using the phonetic similarities. Sentences were split based on the following standard: “the longest sequence of characters parsed without meeting the same separator on both sides of the

³ For more information, see <http://www.sms4science.org/?q=en> (access 13.5.2017).

alignment” (Beaufort et al. 2011: 774). The key contribution is finding that simple segmentation by spaces is not adequate to capture all the possible character mappings (e.g. “*j esper*” to “*j’espère*”). As we can see from Examples (1a) and (1b), word segmentation could be improved using their proposed strategy. Square brackets are markers for the segmentation boundaries; underlines show missing elements. The approach worked very well with an average WER of 9.3% reported on a test set that should be similar to that used by Kobus et al. (2008) according to the authors (Beaufort et al. 2011: 777).

- (1) a. [J esper_] [k__tu] [va_]
b. [J’espère] [que tu] [vas] (Beaufort et al. 2011: 774)

Pennell et al. (2011) looked at a character-based SMT approach for normalization of abbreviations in text messages. They proposed a two-step method to tackle this issue. The first step includes performing a character-based SMT (translation and language models with 5-grams) to get translation candidates. The next step used a character-based SMT with contextual information to choose the best translation candidate. They observed that by using context words they achieved a better accuracy score. This work is of particular importance as we used five of the seven types of abbreviations described in this work: deletions, substitutions, repetitions, swaps and insertions. Table 1 gives an overview of the categories of

shorthand forms as Pennell et al. (2011: 978) defines; the examples in Table 1 are selected from the Belgian French corpus.

Category	Normalized	Original
Deletions	bibliothèque	biblio
Substitutions	quelle	kel
Repetitions	max	maxxxxx
Swaps	nuit	niut
Insertions	ok	oki

Tab. 1. Categories and examples of five shorthand forms

On a dataset of tweets (4,661 tweets with 7,769 tokens), the best result achieved was a 69.7% accuracy using contextual information. The authors also tested their methodologies with the same dataset as Choudhury et al. (2007) and achieved an accuracy of 60.39%.

Li and Liu (2012) compared a two-step process using phonetic sequences with character-based SMT. In the two-step method, they translated non-standard texts into sequence of phonetic symbols in the International Phonetic Alphabet (IPA). Then the phonetic sequences were transformed into words with the help of a dictionary (mapping of phonemes and graphemes). Finally, they combined the aforementioned methodologies to choose the best translation candidate based on the probabilities. They observed that if the translation probability for the character-based SMT is low, then the candidates given by the two-step process with phonetic sequences worked better. They evaluated their methodologies using the same dataset as Choudhury et al. (2007) and concluded that their two-step approach with the phonetic sequences yielded an accuracy of 71.96%.

De Clercq et al. (2013) implemented a simple approach using user-generated contents of different types such as text messages, message board posts, and tweets. They simply performed a phrased-based translation of the corpus. The pairs from phrase table were then fed into a character-based translation system. They draw the conclusion that this combined method achieved the best WER (13% with unigram translation with text messages) than the aforementioned methods employed separately.

Distributional semantics are popular methods to represent words by the context surrounding them. An inspiring work by Sridhar (2015) highlights how effective word embeddings can be employed to learn normalization lexicon.

Distributional semantics originate from the conclusion in Firth (1957: 11): “You shall know a word by the company it keeps”. It means that words appear in the similar contexts have similar meanings. In Examples (2a) – (2d), “Berlin” and “Madrid” are both cities, while “Germany” and “Spain” are both countries. Thus, in a simplified manner, we demonstrate that names of countries and names of cities appear in similar contexts.

- (2) a. Berlin is a big city.
- b. Madrid is a big city.
- c. Germany is a big country.
- d. Spain is a big country.

The basic idea of word embeddings is to represent a word as vectors with real numbers in a vector space. There are many ways of creating vectors, amongst which the simplest method is the bag-of-words representation. As shown in Examples (2a) – (2d), our corpus consists of four sentences. The vocabulary size is nine, as we have nine unique tokens, i.e. “Berlin”, “Madrid”, “Germany”, “Spain”, “is”, “a”, “big”, “city”, “country”. The vector space can be constructed with each unique token as a dimension. The order of word dimensions is defined arbitrary but should be kept consistent for all the vector representations under the same vector space. Thus, the vector representations for “Berlin” and “Madrid” using one-hot encoding (more see Rong 2014:2) are: [1, 0, 0, 0, 0, 0, 0, 0, 0] and [0, 1, 0, 0, 0, 0, 0, 0, 0], respectively (see Table 1). Using this logic, the maximum length of vectors is equal to the number of unique words in the vocabulary. For a more extensive introduction of computational distributional semantics, see Chapter 20: Computational Lexical Semantics in Jurafsky and Martin (2014: 692-701).

	Berlin	Madrid	Germany	Spain	is	a	big	city	country
“Berlin”	1	0	0	0	0	0	0	0	0
“Madrid”	0	1	0	0	0	0	0	0	0

Tab. 2. Examples of one-hot encoded vectors

Word embeddings allow us to perform vector-based calculations. The similarity of words can be computed by the distance between two vectors in the same vector space. Common measure in distributional semantics is the cosine similarity score, which ranges from -1 to 1. The larger the cosine

similarity score is, the more similar the vectors are. For a given word embedding, we can calculate the nearest neighbors to that word by ranking their cosine similarity score in relation to that word. The top 1 candidate is usually taken as the best possible candidate.

Some other concepts to understand *word2vec* (Mikolov et al. 2013a, 2013b) are continuous bag-of-words (CBOW) and skip-gram. For CBOW, one tries to predict the target word based on its context words. In Example (3), if target word is “run”, given a context window of five, the context words are two words on the left and right of the target words, namely, “cats”, “usually”, “faster”, “than”. We move the context window alongside the whole sentence, so in the next iteration, the target word will become “faster”, with the context words “usually”, “run”, “than”, “humans”. Skip-gram is the opposite of CBOW, namely, that it tries to predict the context of a word. In our example, given a word “run”, skip-gram predicts the context “cats”, “usually”, “faster”, “than”.

(3) Cats usually run faster than humans.

word2vec is a computationally efficient way of calculating word embeddings using CBOW and skip-gram models. It utilizes negative sampling which is a way of randomly sampling co-occurrence in a corpus. For the co-occurrence with the word “cats” in our imaginary corpus, instead of extracting all the words that co-occur with “cats”, we only sample a small amount of words, e.g. “always”, “often”, “sometimes”, etc. Negative sampling increases the computational efficiency to calculate word

embeddings. As mentioned before, the length of word vectors is usually the length of vocabulary and these vectors are usually very sparse. In order to have a condensed representation of vectors, dimensionality reduction is necessary.

word2vec models implement neural networks with only one hidden layer (more see Rong 2014); therefore, a dimensionality reduction procedure takes place. The final embeddings of a given word is the row vector of the weight matrix between the input layer and the hidden layer.

multivec (Bérard et al. 2016) is an extension of *word2vec*. Whereas *word2vec* calculates the word vector representation in a single language, *multivec* maps bilingual representations across languages by calculating embeddings for a word in the source language as well as similar word in the same context in the target language. For this reason, *multivec* requires perfectly aligned parallel sentences as input.

Based on what we have discussed before, words with similar semantic meanings tend to appear in similar contexts (e.g. similar vector representations), we shall map the bilingual word embeddings for a small parallel corpus (Example (4)) as shown in Figure 1.

- (4)
- a. The king and queen are coming.
 - b. Der König und die Königin kommen.
 - c. Madrid and Barcelona are cities.
 - d. Madrid und Barcelona sind Städte.

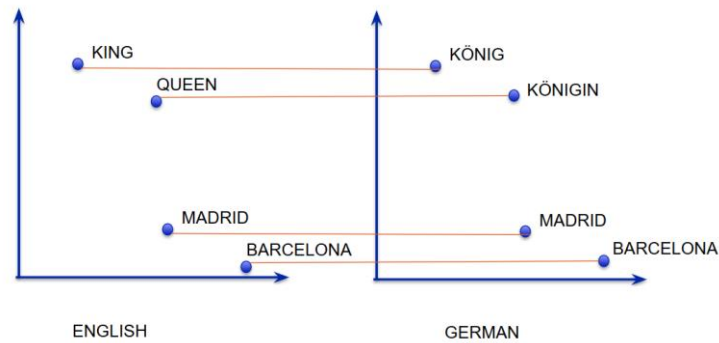


Fig. 1. Examples for bilingual word embeddings mapping of English and German. The graph on the left represents the word embeddings in the two-dimensional plane for the English words “king”, “queen”, “Madrid”, “Barcelona”, the graph on the right for their counterparts in German (see Figure 1). The lines represent the cross-linguistic mappings between the words that share the similar contexts.

3 Corpus and Preprocessing

3.1 Corpus

We used the Belgian French data from the sms4science project, whose goal was to carry out linguistic analysis on communication with SMS. The corpus is available as a manually normalized *Excel* file. The file was structured so each row corresponds to a single SMS, and the two columns we selected for our analysis⁴. Each text message in the corpus is regarded a document.

⁴ The names of the columns we selected are *traduction_normalisee_sans_tag*, *message_non_normalise* where the rows correspond to the normalized and original versions, respectively.

The text messages were previously redacted by the creators of the corpus. All the sensitive data were removed, such as telephone numbers, email addresses, etc. We do not have access to the guidelines for normalization. However, based on manual evaluation, we were able to deduce the annotators used different strategies to normalize the texts. As a result, the quality of normalization is inconsistent; hence, it took a lengthy processing step to acquire perfect sentence alignments from the documents. The numbers of tokens and unique spelling variants are listed in Table 3.

	Original	Normalized
TOKEN	657,572	681,866
SPELLING VARIANT	46,413	26,687

Tab. 3. Tokens and spelling variants in the Belgian French corpus

3.2 Preprocessing

Based on the non-standardized nature of the data, we will see that it will have an effect on the pre-processing and quality of the results. The input for machine translation in *moses* (Koehn et al. 2007) and *multivec* systems (we refer to our two approaches as *moses* and *multivec* hereafter) requires perfectly aligned parallel sentences. In order to generate sentence mapping, we had to identify the sentence boundaries from text messages. A single text message can contain multiple sentences.

To start with, we removed the markups in the corpus for sensitive information (e.g. {???, EMAIL}, {???, NOM}), so that they do not influence the sentence segmenter and word tokenizer. Then we applied the Sentence Segmenter (*nlk.tokenize.sent_tokenize()*) from Natural Language

Processing Toolkit (NLTK in Python 2.7, Bird et al. 2009) on our corpus. We found some discrepancies of sentence alignments mainly because punctuations in the original texts can appear within the sentence as emoticons, abbreviations, emphasis, typos, etc.

- (5) [OK..pour.20h30-
21h,.il.y.aura.2.petit.en.plus.3.et.6.ans,.je.vais.faire.un.ciné.(2
2h30.][)et.retour.av.1H.du.matin,.je.vous.donne.forfait.40.€?.
merci.de.me.repondre.av.midi]
- (6) [OK pour 20h30-21h, il y aura 2 petits en plus 3 et 6 ans, je
vais faire un cinéma (22h30) et retour avant 1 heure du matin,
je vous donne forfait 40 €?][Merci de me répondre avant
midi]

As shown in Examples (5) and (6) for one pair of original and normalized text messages, due to the irregularities in the original texts, the segmenter, trained on standard French, was not able to identify the sentence boundaries in an adequate manner. The square brackets in Examples (5) and (6) show the sentence boundaries identified by the segmenter. We decided to manually correct the sentence boundaries. In the end, we obtained 94,982 sentences (aligned in the original and normalized forms) out of 30,000 SMS. For character-based machine translation, we further split the tokens by word boundaries, then each token was split into individual characters (e.g. “2 m l” to “d e m a i n”). Character alignment was performed with GIZA++ (Och and Ney 2003) in *moses*. We trained the machine translation system

with the sentences of the same length, which takes 73.5% of all the materials from the corpus. The numbers of tokens and unique spelling variants are listed in Table 3. With *multivec*, the inputs are pairs of aligned parallel sentences. The numbers of tokens and spelling variants can be found in Table 4.

	TRAINING		TEST	
	Original	Normalized	Original	Normalized
TOKEN	83,926	83,926	10,490	10,490
TYPE	13,563	9,069	3,360	2,439

Tab. 4. Tokens and spelling variants for *moses* training and testing

4 Methodologies, Tools and Experiments

4.1 Methodologies

As we discuss in the literature review, we can use the original and normalized text messages as parallel sentences in machine translation systems when we consider the original text message as source language and the normalized text message as target language. Since the text messages have monotonous word alignments, given a reasonable word segmentation strategy, character-based SMT can be applied to generate the character mappings between the two versions of text messages.

Using the logic that similar words tend to appear in similar contexts, we can use *multivec* as a method of translating from the original texts to the standardized texts, because the shorthand forms bear the same context, even if they appear in different spelling variants.

4.2 Tools and experiments

4.2.1 *multivec*

For *multivec* we use the entire corpus for training the model (corpus size see Table 3). The unit of analysis in *multivec* is word. The setup of *multivec* is as follows:

1. Dimensions in the vectors: 100
2. Context windows size: 5
3. Minimum absolute counts in corpus: 5
4. Learning rate: 0.05
5. Iterations: 5
6. Subsampling: 0.001
7. Negative sampling: 5
8. The remaining parameters: default

We observe that true casing works better than lower casing. Moreover, we achieve the following accuracy scores in the nearest neighbor list for the top 1 item and top 3 items: 55.3%, 59.0%.

4.2.2 *moses*

The unit of analysis in *moses* is character. In the preprocessing phase we took only sentences with the same length, because the simple segmentation by word boundaries can lead to mismatch in the original and normalized sentences (corpus size see Table 4). We tried language models of up to 7-

grams and found out that the normalization accuracy of shorthand forms stabilized at 7-grams.

The best result is given by taking the top 1 candidate in 10-best list, language model 7-gram, grow-diag-final-and. The average length of a token in our corpus is 7 (6.76 for *moses*, 5.7 for *multivec*), which might explain why the 7-gram language model scored the best.

5 Results Analysis

We took a test set of 10,490 shorthand forms for both *moses* and *multivec*. We looked at recall of two approaches and their overlaps in identifying shorthand forms. To evaluate accuracy, we took the overlapping part as the test set. *multivec* retrieved 5,809 instances, whereas *moses* found 5,283. We report only accuracy of the task, because the precision of retrieval is 100%. The accuracy score is equal to the recall of shorthand forms in the test set. The two approaches overlap in 3,987 retrieved instances, with discrepancies of 1,821 for *multivec* and 1,291 for *moses*, respectively. Spelling variants of a lemma refer to the variability of that lemma in different forms. For the lemma “*demain*”, there are plenty of spelling variants in our corpus such as “*dm1*”, “*dm 1*”, “*dem1*”, “*dem 1*”, “*2m1*”, “*2main*”, “*2min*”, etc.

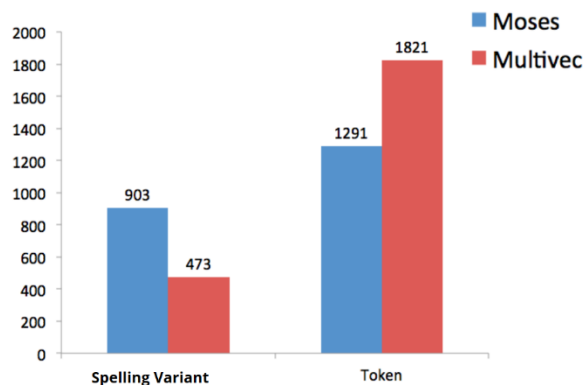


Fig. 2. Discrepancies of tokens and spelling variants

	moses	multivec
TOKEN (retrieved)	1,291	1,821
SPELLING VARIANT (retrieved)	903	473
TOKEN (training set)	83,926	657,572
SPELLING VARIANT (training set)	13,569	46,413
NORMALIZED TOKEN%	1.50%	0.28%
NORMALIZED SPELLING VARIANT%	6.65%	1.02%

Tab. 5. Normalized retrieved rate for tokens and spelling variants

We analyzed the retrieval discrepancies extensively at the token and spelling variant levels in the following sections. Because we trained the systems of *multivec* and *moses* with different sizes of training materials, normalization of retrieved instances against the size of training materials is needed in order to evaluate the output on the same basis (Table 5). As seen in Figure 2 and Table 5, *moses* recognizes a lot more cases than *multivec* on both the token and spelling variant levels. However, in the test set, we notice that *multivec* is able to correctly identify more tokens, while *moses* is better at identifying spelling variants (see Figure 2).

When we compare *moses* with *multivec*, *multivec* recognizes the most deletions (see Figures 3 and 4). Although *moses* has less training materials, it is better at recognizing substitutions and repetitions. There are not many cases of swaps and insertions in the test set; thus, it is difficult to judge the performances of the two systems on these categories.

We looked at the different categories of shorthand forms recognized in the test set and observed that *moses* and *multivec* have different preferences for normalizing different categories. Deletions are the most common category in the test set, followed by substitutions. The number of deletions is 3 times that of substitutions. The number of substitutions is 20 times that of repetitions. Swaps and insertions are much less observable in test set (see Figures 3 and 4).

As demonstrated in Table 6, *multivec* is able to find severely shorted forms (“we” for “weekend”, “kel” for “quelle”). *Moses* is capable of finding highly varied forms of spelling variants.

	moses	multivec
Deletions	“ojrd8” (“aujourd’hui”)	“we” (“weekend”)
Substitutions	“ki” (“qui”)	“kel” (“quelle”)
Repetitions	“groos” (“gros”)	“bizzz” (“biz”)
Swaps	“niut” (“nuit”)	--
Insertions	“dorlotter” (“dorloter”)	“oki” (“ok”)

*normalized word in bracket

Tab. 6. Examples from *moses* and *multivec* for five shorthand categories

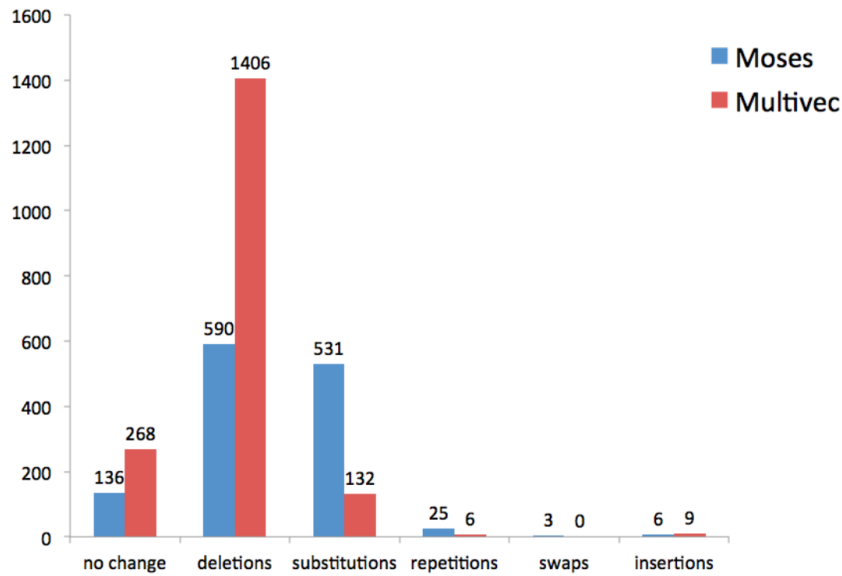


Fig. 3. Accurate normalization of token level for *moses* and *multivec* in five categories. When we compare *moses* with *multivec*, we see that the most common type that has been recognized by *multivec* is deletions. Although *moses* has less training materials, it is better at recognizing substitutions and repetitions. There are not many cases of swaps and insertions in the test set; hence, it is difficult to judge the performances of the two systems on these categories.

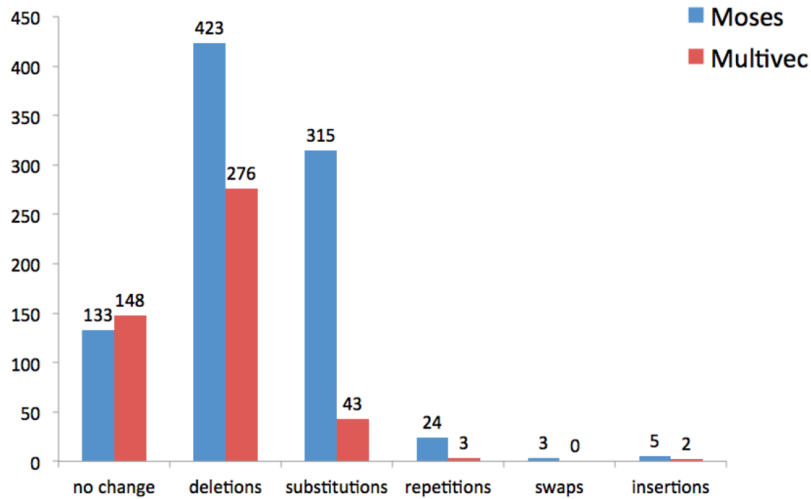


Fig. 4. Accurate normalization of spelling variant level for *moses* and *multivec* in five categories.

When it comes to the spelling variants, we see that across all the categories, *moses* captures most of the normalization cases accurately. In the test set, it highly outperforms *multivec* when it comes to substitutions. In general, *moses* has a higher rate of recall on different types, whereas *multivec* has high precision, as it recognizes words that appear in similar contexts all the time.

Original	Normalized	Approach	Frequency
auj		multivec	7
adj			4
aujourd'hui			1
aujourd'hui		moses	3
aujourdwi	aujourd'hui		1
ojourd'hui			1
aujourd			1
ojrd8			1
ojord			1

Tab. 7. Examples for “*aujourd'hui*” in different spelling forms

Original	Normalized
11h58	11h58
12:30	12h30
17.40h	17h40
18.30	18h30
19	19h
19h3o	19h30
20.49	20h49
8.30	8h30

Tab. 8. Examples for temporal expressions

Some other interesting observations are that *moses* is flexible on different spelling variants (see Table 7) and is good with temporal expressions (see Table 8). As we see in Table 7, for the normalization form “*aujourd'hui*”, nine different spelling variants were found in the test set, including the

lemma. Both systems have different preferences in “translating” from the original to the normalized forms.

We noticed that *multivec* recognizes always the same items, e.g. “*auj*” (7 times), “*ajd*” (4 times). *Moses* recognizes six different spelling variants, and except for “*aujourdhui*” (3 times), the other five spelling variants are hapax legomena. “*auj*” and “*ajd*” are extremely difficult to for *moses* to normalize, as long-distant deletion of character sequences is involved.

We also noticed that *moses* is good at finding temporal expressions (see Table 8). In these forms, the variation lies in separators between hours and minutes, e.g. “*h*”, “*:*”, “*.*”. *Moses* is able to normalize extremely varied form such as “*17.40h*” to its normalized form “*17h40*”.

Moses is highly flexible with out-of-vocabulary words (words that have not been “seen” in its training materials) and is good at identifying different spelling variants. During the training of *multivec*, it omitted out-of-vocabulary words. Nonetheless, *moses* has the advantages of treating every normalization as a translation task and tries to find the most probable character mappings from the original form to the normalized form.

However, *Moses* can bring in noises in character mapping, because it calculates the path with the highest joint probabilities. For example, the word “*l’université*” was wrongly mapped to “*l’universitait*” by taking the path with the highest probability. The probabilities of character mapping are taken from the phrase table and path 1 is the most probable path amongst all the mapping strategies. Path 2 is an example of another possible way to

normalize the word “*l’université*”, and since it has a much lower probability compared to path 1, *moses* opted for path 1 (see Table 9).

Path 1	l’uni → l’uni (0.75)	v → v (0.86)	s → s (0.71)	i → i (0.86)	té → tait (0.66)	l’universitait (0.26)
Path 2	l’uni → l’uni (0.75)	ver → ver (0.28)	si → si (0.6)	té → tait (0.66)		l’université (0.08)

Tab. 9. *Moses* phrase table and the different paths of normalization

To summarize, using *multivec* is looking for existing items for a dictionary. If the item is unlisted in the dictionary, *multivec* cannot solve the normalization tasks. On the contrary, *moses* treats its normalization task as applying character mapping rules to the non-standard forms. It attempts to come up with the most plausible path.

6 Conclusion

Using *multivec* to normalize shorthand forms has its advantages as this process is always consistent and runs extremely fast. Moreover, it is able to retain contextual information; thus, highly abbreviated items can be deciphered by *multivec*. Unfortunately, this method is highly intolerant of unseen items in training materials. Consequently, it requires large input of parallel materials.

Moses is highly flexible when it comes to translations of out-of-vocabulary items. Moreover, it is able to retrieve multiple spelling variants. As discussed before, *moses* chooses the path of the highest joint probability. Thus, it can occur that during the character mapping process, noises (an incorrect mapping with a higher probability) are inserted into the output.

Furthermore, *moses* requires a long training time and we have to train the language model accordingly, i.e. we have to run different different models with different parameters until reaching a satisfactory result, which can lead to even longer training and tuning time.

As we do not have access to the evaluation sets in the previous studies on the Belgian French corpus (see Kobus et al. (2008), Yvon (2009) and Beaufort et al. (2011)) and we only investigated word normalization in our experiments, we cannot compare our results directly with those in the state-of-the-art study (i.e. Beaufort et al. (2011)). Our contributions are as follows:

1. To the best of our knowledge, we are the first study that applied the character-based SMT to normalize the Belgian French text messages.
2. We applied word embeddings that encode contextual information to increase recall and precision of normalization.
3. We identified the efficacy and necessity of combining the two approaches in normalization: character-based SMT (*moses*) and word embeddings (*multivec*).
4. Our detailed result analyses provide further insights into future work.

7 Future Work

As we did not include contextual information into *moses*, as it could be quite challenging to encode contextual information in the character system,

which requires precise segmentation of word boundaries. We suggest that in the future, we could combine the two approaches as the word embeddings from *multivec* already entails the contextual information of a word. It is plausible to identify the candidates for character-based SMT by *multivec*, and then use the parallel pairs as input for machine translation systems.

Another point where we could improve is incorporating parallel sentences of different lengths, aka phrase normalization, where one side in the pair is a multi-word expression (e.g. “*ktv*” to “*que tu veux*”). We also envisage using more materials from the sms4science project, namely the French sms4science corpus collected in France. Finally, in the realm of embeddings, we would like to extend our methodologies of creating embeddings to units on the sub-word level. *fastText* (Bojanowski 2016) provides the theoretical framework for applying this methodology to any language. We assume that mapping the original and normalized texts on the level of sub-word unit can reduce the number of out-of-vocabulary items, as well as omit noises as in *moses*. Moreover, as the sub-word embeddings entail morphological and contextual information the mapping quality could be improved. We will address the aforementioned open issues in our future experiments.

References

Beaufort, Richard, Sophie Roekhaut, Louise-Amélie Cougnon and Cédric Fairon. 2010. “A hybrid rule/model-based finite-state framework for

normalizing SMS messages”. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*: 770-779.

Bérard, Alexandre, Christophe Servan, Olivier Pietquin and Laurent Besacier. 2016. “Multivec: a multilingual and multilevel representation learning toolkit for NLP”. *The 10th edition of the Language Resources and Evaluation Conference*: 4188-4192.

Bird, Steven, Edward Loper and Ewan Klein. 2009. *Natural Language Processing with Python*. O’Reilly Media Inc.

Bojanowski, Piotr, Edouard Grave, Armand Joulin and Tomas Mikolov. (2016). “*Enriching word vectors with subword information*”. <<https://arxiv.org/abs/1607.04606>>, access 13.5.2017.

Choudhury, Monojit, Rahul Saraf, Vijit Jain, Sarkar Sudeshna and Anupam Basu. 2006. “Investigation and modeling of the structure of texting language”. *International journal on document analysis and recognition* 10 (3): 157-174.

De Clercq, Orphée, Sarah Schulz, Bart Desmet, Els Lefever and Véronique Hoste. 2013. “Normalization of Dutch user-generated content”. *Proceedings of 9th International conference on Recent Advances in Natural Language Processing*: 179-188.

Fairon, Cécrick, Jean R. Klein and Sébastien Paumier. 2007. *Le langage SMS: étude d'un corpus informatisé à partir de l'enquête «Faites don de vos SMS à la science»*. Presses univ. de Louvain.

Firth, John R. 1957. "A synopsis of linguistic theory 1930–1955". *Studies in linguistic analysis*. Oxford: Blackwell: 1–32.

Jurafsky, Daniel and James H. Martin. 2009. *Speech and language processing (1st. ed.)*. Pearson.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin and Evan Herbst. 2007. "Moses: Open source toolkit for statistical machine translation". *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*: 177-180.

Kobus, Catherine, François Yvon and Géraldine Damnati. 2008. "Normalizing SMS: are two metaphors better than one?". *Proceedings of the 22nd International Conference on Computational Linguistics* 1: 441-448.

Koch, Peter and Wulf Oesterreicher. 2001. *Gesprochene und geschriebene Sprache. Französisch, Italienisch, Spanisch*. Berlin/New York: de Gruyter.

Li, Chen and Yang Liu. 2012. "Normalization of text messages using character-and phone-based machine translation approaches". *Proceedings of 13th Annual Conference of the International Speech Communication Association*: 2330-2333.

Mikolov, Tomas, Kai Chen, Corrado Greg and Jeffrey Dean. 2013a. "Efficient Estimation of Word Representations in Vector Space". *The*

Workshop Proceedings of the International Conference on Learning Representations. < <https://arxiv.org/abs/1301.3781>>, access 13.5.2017.

Mikolov Thomas, Sutskever Ilya, Kai Chen, Greg Corrado and Jeffrey Dean. 2013b. “Distributed Representations of Words and Phrases and their Compositionality”. *Advances in Neural Information Processing Systems*. <<https://arxiv.org/abs/1310.4546>>, access 13.5.2017.

Och, Franz Josef and Hermann Ney. 2003. “A Systematic Comparison of Various Statistical Alignment Models”. *Computational Linguistics* 29 (1): 19-51.

Pennell, Deana L. and Yang Liu. 2011. “A Character-Level Machine Translation Approach for Normalization of SMS Abbreviations”. *Proceedings of the 2011 International Joint Conference on Natural Language Processing (IJCNLP)*: 974-982.

Rong, Xin. (2014). “word2vec parameter learning explained”. <<http://https://arxiv.org/abs/1411.2738>>, access 13.5.2017.

sms4science project. (2004). <<http://www.sms4science.org/?q=en>>, access 13.5.2017.

Sridhar, V. K. R. (2015). “Unsupervised Text Normalization Using Distributed Representations of Words and Phrases”. *Proceedings of the 2015 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL)*: 8-16.

Van Compernelle, Rémi. A. 2010. “The (slightly more) productive use of ne in Montreal French chat”. *Language Sciences* 32 (4): 447–463.

Yvon, François. 2010. “Rewriting the orthography of SMS messages”. *Natural Language Engineering* 16 (02): 133-159.