Architecture for Text Normalization using Statistical Machine Translation techniques

V. López-Ludeña, R. San-Segundo, J. M. Montero, R. Barra-Chicote, J. Lorenzo

Speech Technology Group. E.T.S.I. Telecomunicación.
Universidad Politécnica de Madrid. Spain.
veronicalopez@die.upm.es

Abstract

This paper proposes an architecture, based on statistical machine translation, for developing the text normalization module of a text to speech conversion system. The main target is to generate a language independent text normalization module, based on data and flexible enough to deal with all situations presented in this task. The proposed architecture is composed by three main modules: a tokenizer for splitting the text input into a token graph (tokenization), a phrase-based translation module (token translation) and a post-processing module for removing some tokens. This paper presents initial experiments for numbers and abbreviations. The very good results obtained validates the proposed architecture.

Index Terms: text normalization, text to speech conversion, language translation, numbers, acronyms, abbreviations.

1. Introduction

Although Text to Speech (TTS) conversion is the area where more effort is devoted to text normalization, dealing with real text is a problem that also appears in order applications like machine translation, topic detection and speech recognition. In an ideal situation, there would be an unambiguous relationship between spelling and pronunciation. But in real text, there are not ordinary words like numbers, digit sequences, acronyms, abbreviations, dates, etc.

The main problem of a text normalization module consists of converting Non-Standard Words (NSWs) into regular words. This problem can be seen as a translation problem between a real text (including NSWs) and an ideal text where all the words are standard: there is a unique relation-ship between word spelling and its pronunciation.

2. State of the art

One of the main references focused on text normalization is (Sproat et al, 2001) [1]. In this reference, authors propose a very complete taxonomy of NSWs considering 23 different classes grouped in three main types: numerical, alphabetical and miscellanea. Sproat et al describe the whole normalization problem of NSWs, proposing solutions for some of the problems: a good strategy for tokenizing the input text, a classifier for determining the class associated to every token, some algorithms for expanding numeric and other classes that can be handled "algorithmically", and finally, supervised and unsupervised methods for designing do-main-dependent abbreviation expansion modules.

Additionally, it is also important to remark other references that have addressed specific problems included in the text normalization research line. Focused on abbreviations and acronyms, there are several efforts focused on extracting them from text automatically [2][3][4] and other efforts trying to model how they are generated [5][6]. Numbers [7] and proper names [8][9][10] have been also the target of other research works.

Nowadays, much effort on text normalization is focused on SMS language, interchanged through mobile phones and social networks like Facebook or Twitter [11][12][13].

Due to the important advances obtained in ma-chine translation in the last decade, there has been an increasing interest on exploring the machine translation capabilities for dealing with the problem of text normalization [14][15].

This paper proposes a general architecture based on statistical machine translation techniques for text normalization. The main target is to generate a language independent text normalization module, based on data (instead of on expert rules) and flexible enough to deal with all situations. This paper presents initial experiments for numbers and abbreviations for Spanish.

3. Architecture Description

Figure 1 shows the architecture diagram of the text normalization module proposed in this paper.

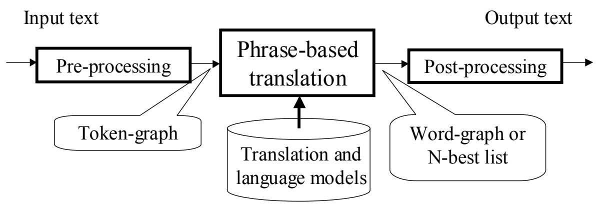

Figure 1. *Architecture diagram*

This architecture is composed by three modules: a pre-processing module that splits the text input into a token graph (tokenization), a phrase-based translation module (token translation) and a post-processing module for removing some tokens.

3.1. Pre-processing: sentence tokenization

At this first module, the input text is split into tokens. This process is carried out in two different steps.

At the first step, a preliminary token sequence is generated considering a small set of rules. As one of the main targets of this work is to provide a language independent architecture, the main rules should be language independent:

- The first rule supposes that blank characters provide an initial segmentation in tokens.
- The second rule subdivides initial tokens (sequence of characters between blank characters) considering some homogeneity criterions:
  - Tokens must have only alpha or numerical characters. If there is a change from alpha to number or vice-versa, the token must be subdivided.
  - Punctuations characters must be considered as independent tokens

Additionally to these language independent rules, it is possible to add new rules focused on one language or on a set of languages (Romances ones, for example). For example, in English, it is possible to consider subdividing a token if there is a change between lower and upper letters or vice-versa [1].

Secondly, some of the tokens (NSWs) are re-written in a different format in order to facilitate their posterior translation. At this step, before rewriting, it is necessary to classify each token as a standard word (W) or as a non-standard word (NSW). This classification can be done considering a dictionary of standard words in this language or considering a more complex classifier based on some features obtained from the target token and its context: character language model, vowels, capitals, etc. In this work, the machine translation module has to deal with this ambiguity without adding additional information.

If the token is classified as a NSW, it is split into letters including some separators at the beginning and at the end of the letter sequence. For example, UPM (Universidad Politécnica de Madrid in Spanish) is rewritten into # U P M #. This way of rewritten an alpha token tries to introduce a high flexibility to facilitate the text normalization process. Considering sequence of letters, some non seen abbreviations could be normalized using the translations of its letters individually.

Also, all the numbers are rewritten dividing the token into digits. Every digit is complemented with its position in the number sequence. For example: 2012 is rewritten as 2_4 0_3 1_2 2_1, where 2_4 means the digit 2 in the 4th position (beginning from the right). The Roman numbers are first translated into Arabic ones before rewriting.

As it will be shown in the next section, the translation module can deal with graphs of tokens as input. Thanks to this characteristic, it is possible to work with fuzzy decisions when classifying every token as standard word or NSW. Considering a token graph, both alternatives can be considered with different weight if necessary. Figure 2 shows an example of token graph for the sentence "Welcome to UPM2012".

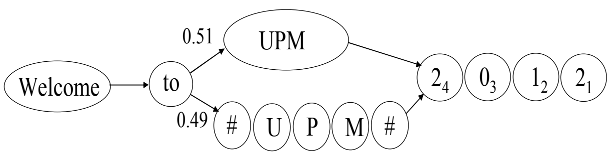

Figure 2. *Token graph for the sentence "Welcome to UPM2012"*

The token "UPM2012" is divided into two tokens: UPM and 2012. The first one, UPM, is rewritten considering two possibilities: as it is, and letter by letter. The second one is a number and it is rewritten digit by digit, including information about its position.

The main target of the standard vs. non–standard word classifier is to detect with high accuracy standard words in order to reduce the token graph complexity, avoiding alternative paths in these cases.

3.2. Token translation

The token translation is performed using a phrase-based system. The phrase-based translation system is based on the software released from Workshops on Statistical Machine Translation (http://www.statmt.org). The translation process uses a phrase-based translation model and a target language model.

These models have been trained according to these steps (Figure 3).

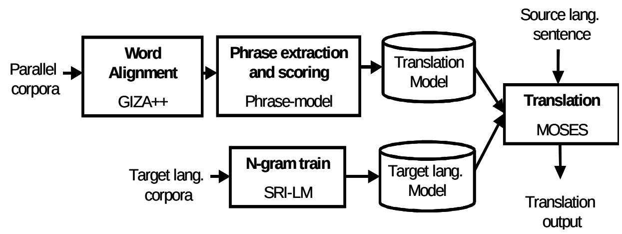

Figure 3. *Process for training the translation and target language models.*

The first step is word alignment computation. In this step, the GIZA++ software [16] has been used to calculate the alignments between source and target tokens. In order to establish these alignments, GIZA++ combines the alignments in both directions. As there are many standard words, they are the same tokens in source and target languages, being important reference points for the alignment.

The second step is phrase extraction [17]. All token phrase pairs that are consistent with the token alignment are collected. For a phrase alignment to be consistent with the word alignment, all alignment points for rows and columns that are touched by a rectangle have to be in the rectangle, not outside. The maximum size of a phrase has been increased to 20 in order to deal with token graphs including sequences of letter and digits properly.

Finally, the last step is phrase scoring. In this step, the translation probabilities are computed for all phrase pairs. Both translation probabilities are calculated: forward and backward.

The Moses decoder (http://www.statmt.org/moses/) is used for the translation process. This program is a beam search decoder for phrase-based statistical machine translation models. In order to obtain an N-gram language model needed by Moses, the SRI language modeling toolkit has been used [18].

3.2.1. Corpora necessity

In order to generate the token translation module, it is necessary to train a translation and a target language model.

For training the translation module is necessary to develop a parallel corpus including examples of all possible type of NSW described in the taxonomy presented at [1]. Some examples are:

- Abbreviations and Acronyms: “The UPM is …” and “The Universidad Politécnica de Madrid is ...”.
- Numbers: “more than 120 cars” “more than one hundred and twenty cards”.
- Dates and times: “On May 3rd, 2012” “on may third , two thousand and twelve”
- Webs and emails: “example@upm.es“ example at U P M dot E S”
- Money and percentages: “$3.4 billions” “three dot four billions dollars”
- Misspelling or funny spelling: “CU8er” “see you later”.

This is the most important aspect when developing the text normalization module. The system performance depends strongly on the data used to train the translation model. Also, parallel corpus generation is a costly task that should be supported with automatic procedures to reduce this cost. With this idea, many efforts have been devoted to obtained appropriate corpora from raw text with a small supervision [9][3][7].

One interesting characteristic of Moses is the possibility of combining different phrase tables (translation models) during the translation process using different weights. Considering this possibility, an interesting analysis for future work will be to compare the possibility of training individual phrase tables for each type of NSW or generating a unique one.

One important aspect to consider is that the source language (in the parallel corpora) must be pre-processed in the same way as the input text with the difference that in this case, the parallel corpus does not have token graphs with two alternatives but only token sequences with the correct alternative.

About the target language model, it is important to consider the target side of the parallel corpora, but also, normalized sentences in different contexts. These additional sentences are interesting to learn the best normalization for a given NSW, depending on the context.

3.3. Post-processing

This module performs several actions in order to generate the normalized text to the speech synthesizer. One of the main actions has been to remove unnecessary tokens. For example, if after the translation module there are any # tokens (used for defining the limits of the letter sequences), they must be removed.

Additionally, given that the translation module can generate a token graph or a sequence of N-best token sequence, it would be possible to add new translation modules in order to improve the translation process by considering new language models for reordering the N-best token sequences or searching the output token graph.

4. Initial Experiments

In this paper, initial experiments are reported focused on numbers and abbreviations. About numbers, the main target is to define how the architecture can be adapted to deal with numerical numbers in general. The second objective is to deal with abbreviations (including acronyms), distinguishing when a token is a NSW (acronyms or abbreviations) or a standard word. For these experiments, a parallel corpus has been created considering an already developed text normalization module based on rules and word lists. The main idea is trying to learn these rules from data automatically.

For evaluating the performance of the translation system, the BLEU (BiLingual Evaluation Understudy) metric has been computed using the NIST tool (mteval.pl) and the WER (Word Error Rate). It is important to note that BLEU is an accuracy metric while WER is an error metric.

4.1. Experiments with numbers

For these experiments, three parallel corpora have been considered, 800 numbers for training, 1000 for validation and 4000 for testing. These data sets have been created randomly; guarantying that one number only appears in one of the sets. Table 1 includes some examples from the parallel corpora.

Original text	Normalized test
123.456,34	Ciento veintitrés mil cuatrocientos cincuenta y seis con treinta y cuatro
1.256,3	Mil doscientos cincuenta y seis con tres

Table 1. *Examples of numbers.*

Table 2 shows different experiments considering different codification strategies. In the first one, the digits are grouped in groups of three digits. In this case, there are many errors coming from the confusion between dots referring to millions or thousands. When considering the integer part completely, the results improve significantly. Finally, in the last experiment, a different codification strategy is considered for the decimal part. In this case, the position is coded from the right of the decimal part instead from the comma character: right to left instead of left to right.

System or experiment	BLEU (%)	WER (%)
Baseline: considering groups of three digits Example: 123.400,2 - $1_3\ 2_2\ 3_1\ .\ 4_3\ 0_2\ 0_1\ ,\ 2_1\ 3_2$	80.5	10.3
No considering groups of three digits Example: 123.400,2 - $1_6\ 2_5\ 3_4\ 4_3\ 0_2\ 0_1\ ,\ 2_1\ 3_2$	96.1	2.2
No considering groups of three digits and different codification for decimals. Example: 123.400,2 - $1_6\ 2_5\ 3_4\ 4_3\ 0_2\ 0_1\ ,\ 2_{-2}\ 3_{-1}$	**96.6**	**1.9**

Table 2. *Different codification strategy for numbers normalization*

In order to analyse the effect of the size of the training set, authors performed two additional experiments increasing and reducing the amount of data to train the translation model (Table 3):

Different amount of training data	BLEU (%)	WER (%)
400 numbers	92.6	4.4
800 numbers	96.6	1.9
1800 numbers	**97.5**	**1.5**

Table 3. *Experiments with different training sets*

As it is shown, a good compromise for the training set is around 1000, in order to get a WER lower than 2%.

4.2. Experiments with abbreviations

For these experiments a parallel corpus with 5225 sentences has been divided in training (4054 sentence), tuning (500 sentences), and testing (671 sentences). Every sentence contains one abbreviation (or acronym). Table 4 includes some examples from the parallel corpora.

Original text	Normalized test
El BBVA subió los precios (The BBVA bank increased the prices)	El be be uve a subió los precios
UGT no negociará más (UGT will not negotiate more)	U ge te no negociará más

Table 4. *Examples of sentences with abbreviations.*

Table 5 shows the results for the experiments with abbreviations.

Abbreviations Experiments	BLEU (%)	WER (%)
Baseline	96.1	2.9

Table 5. *Experiments with abbreviations*

The main errors come from those examples that appear in the test set but not in the training set. In these cases, the system leaves the abbreviations as they are generating errors.

5. Conclusions

In this paper, authors have presented initial efforts for developing a text normalization module to be included in a text to speech conversion system. During the design of this module the main characteristics considered have been language independence, based on data instead of on expert rules and a high level of flexibility to deal with all situations presented in this task.

The architecture proposed in this paper is based on a phrase-based translation system (Moses), considering its main possibilities: dealing with word-graphs at the input and combination of different translation models. This architecture is composed by three modules: a tokenizer for splitting the text input into a token graph (tokenization), a phrase-based translation module (token translation) and a post-processing module for removing some tokens.

Initial experiments with numbers and abbreviations have reported very good results validating the architecture proposed in this paper.

6. Acknowledgements

The work leading to these results has received funding from the European Union under grant agreement n° 287678. It has also been supported by TIMPANO (TIN2011-28169-C05-03), ITALIHA (CAM-UPM), INAPRA (MICINN, DPI2010-21247-C02-02) and MA2VICMR (CAM, S2009/TIC-1542) projects.

7. References

[1] Sproat, R., Black, A., Chen, S., Kumar, S., Ostendorf, M., and Richards, C., 2001. Normalization of non-standard words. Computer Speech and Language, 15(3), 287-333, 2001.

[2] Yeates. S., 1999. Automatic Extraction of Acronyms from Text. New Zealand Computer Science Research Students' Conference. 1999.

[3] Larkey, Leah, Paul Ogilvie, Andrew Price and Brenden Tamilio, 2000. Acrophile: An Automated Acronym Extractor and Server, In Proceedings of the ACM Digital Libraries conference, pp. 205-214, 2000.

[4] Chang, J.T., Schütze, H., and Altman. R.B., 2002. Cre-ating an Online Dictionary of Abbreviations from MEDLINE. JAMIA.

[5] Cannon, G., 1989. Abbreviations and acronyms in English word-formation. American Speech, 64:99-127.

[6] Pennell D., and Liu Y., 2011a. Toward text message normalization: Modeling abbreviation generation. Proceedings of the IEEE. pp. 5364-5367.

[7] Sproat, R., 2010. Lightly Supervised Learning of Text Normalization: Russian Number Names, IEEE Workshop on Spoken Language Technology, Berkeley, CA, 2010.

[8] Bikel, D., Schwartz, R., and Weischedel, R., 1999. An algorithm that learns what's in a name. Machine Learning, 34(1/3):221--231, 1999.

[9] Collins, M., and Singer, Y., 1999. Unsupervised Models for Named Entity Classification. EMNLP/VLC-99.

[10] Jonnalagadda and Topham. 2010. NEMO: Extraction and normalization of organization names from Pub-Med affiliations. J Biomed Discov Collab. Vol 5 pp 50-75.

[11] Brody, S., Diakopoulos. N., 2011. Cooooollllllll!!!!!!!! Using Word Lengthening to Detect Sentiment in Mi-croblogs. EMNLP 2011.

[12] Han B., and Baldwin, T., Lexical normalisation of short text messages: Makn sens a #twitter. ACL 2011.

[13] Kaufmann, M., 2010. Syntactic Normalization of Twitter Messages. Int'l Conference on NLP.

[14] Aw, et al. 2006. A phrase-based statistical model for SMS text normalization. ACL 2006.

[15] Pennell D., and Liu. Y., 2011b. A Character-Level Ma-chine Translation Approach for Normalization of SMS Abbreviations. IJCNLP.

[16] Och J., Ney. H., 2003. "A systematic comparison of various alignment models". Computational Linguistics, Vol. 29, No. 1 pp. 19-51, 2003.

[17] Koehn P., F.J. Och D. Marcu. 2003. "Statistical Phrase-based translation". Human Language Technology Conference 2003 (HLT-NAACL 2003), Edmonton, Canada, pp. 127-133, May 2003.

[18] Stolcke A. 2002 "SRILM – An Extensible Language Modeling Toolkit". ICSLP. 2002. Denver CO, USA.