Text Normalization as a Special Case of Machine Translation

Graliński Filip, Jassem Krzysztof, Wagner Agnieszka, Wypych Mikołaj

Abstract. The paper presents some problems of text normalization for an inflected language like Polish. It is shown that text normalization, being usually one of the first steps of text-to-speech synthesis, is a complex process, referring to several levels of analysis: morphological, syntactical and semantic. It is claimed here that text normalization may be treated in a similar way to Machine Translation: The tools and algorithms developed for Machine Translation may be used for text normalization, with the "spoken language" being treated as the target language.

Introduction

Text normalization is an automated process of the synthesis processor, which converts the written form (orthographic form) of the text into the spoken form [1]. Various approaches have been taken towards text normalization [1–5]. Normalization is often the first phase of text preprocessing in a text-to-speech system [6–10]. It consists in expanding abbreviations, converting names, numbers, acronyms, dates etc. into their spoken form. For example the string *$200* should be expanded in English into *two hundred dollars*.

Normalization of a Polish text is somewhat more complex and this is due to the inflected character of the language and the lack of efficient and widely available language analysis tools such as morphological or syntactic analyzers. As a result, so far the existing text-to-speech systems for Polish [7–9, 11, 12] have not succeeded in solving many text normalization related problems.

Some difficulties of text normalization in the Polish language may be exemplified by the sentence:

(0) dla (1) p. (2) dr. (3) J. (4) Kowalskiego (5) leg.(6) się (7) dow. (8) osob. (9) BAC1234567, (10) zam. (11) na (12) os. (13) B. (14) Chrobrego (15) 10 (16) m (17) 7, (18) 61-100 (19) Poznań

which should be expanded into:

(0) dla (1) pana (2) doktora (3) jot (4) kowalskiego (5) legitymującego (6) się (7) dowodem (8) osobistym (9) be a ce jeden dwa trzy cztery pięć sześć siedem, (10) zamieszkałego (11) na (12) osiedlu (13) bolesława (14) chrobrego (15) dziesięć (16) mieszkania (17) siedem, (18) sześćdziesiąt jeden sto (19) Poznań

The practical aim of the research has been to deliver the module for the normalization of Polish texts. The module is to be used in the text-to-speech synthesiser being developed to enable the spoken output of the translation system, TRANSLATICA(`http://translatica.pl, http://www.poleng.pl`). It is claimed here that the process of text normalization is very similar to that of machine translation from one natural language to another. In fact, normalization may be successfully treated by the same mechanisms and algorithms as those used in Machine Translation (MT) with the exception that the target language is "the spoken form" of the source language. The thesis has been verified by applying the mechanisms of an MT system TRANSLATICA to the normalization of Polish texts.

TRANSLATICA is a bidirectional MT system between Polish and English, and Polish and Russian. TRANSLATICA uses transfer approach and the translation process is based on a lexical database that has to be prepared beforehand. For a given input, the translation process consists of the following steps:

1. tokenization (i.e. division into words, punctuation marks, and other types of tokens),
2. sentence segmentation,
3. lexical analysis (including look-up for lexical phrases)
4. syntactic analysis for each sentence in order to obtain a set of possible parses,
5. syntactic disambiguation in order to choose the best parse,
6. semantic disambiguation in order to select the best of possible meanings
7. transfer into the representation of the target language,
8. syntactic transformation of the target representation,
9. inflectional synthesis of the output.

The paper shows that in order to normalize the text properly, similar steps for inflected languages like Polish should be taken. This hypothesis has been confirmed by experiments that consisted in applying the TRANSLATICA translation mechanisms to text normalization.

Preparation of the lexicon

Text normalization should be based on a lexicon very similar to that used in Machine Translation. It has to include the graphical forms of the entries, descriptions of their syntactical features, rules for the lexical transfer and equivalents in the output language (i.e. in the spoken language).

The logical structure of the TRANSLATICA lexicon is the following: each entry is identified by its basic form (infinitive for verbs, nominative singular for nouns, etc.) and a POS tag (a grammatical identifier).

For example the basic form *powieść* represents two lexicon entries: verbal (to lead something), and nominal (a novel). Two lexicon entries are also represented by the form *boks*. One of the entries (boxing – sport) represents the lexeme with singular forms only, the other (box – room) may have inflected forms in both numbers. The word *kontroler* (controller) forms two entries because one of the meanings (person) forms the plural *kontrolerzy* and the other meaning (computer device) forms the plural form *konrolery*.

Similar phenomena may be noticed in text normalization. For example the abbreviation "p." forms no less than 5 different entries in the normalization "bilingual" lexicon:

Table 1. Normalization of the abbreviation *p.* in a Polish text

Abbr.	Expansion	Meaning	POS-tag	meaning of POS-tag	Example of normalization
p.	pan	Mr	N:1	noun, masculine, human	dla p. Kowalskiego → dla pana Kowalskiego
	ni	Mrs	N:4	noun, feminine	dla p. Kowalskiej → dla pani Kowalskiej
	punkt	point	N:3	noun, masculine, inanimate	zdobył 5 p. → zdobył pięć punktów
	piętro	floor	N:5	noun, neutral	3-e p. → trzecie piętro
	patrz	see	S	sentence	p. [1] → patrz jeden

The POS-tag plays the crucial role in the syntactic disambiguation of the abbreviation. This will be described in section "Syntactic analysis".

In Machine Translation it is often the case that one lexicon entry may consist of a few equivalents in the target language. The same may apply to the normalization lexicon. The abbreviation *ks.* gives rise to only one lexical entry (denoted by the POS-tag *N:1*) but it may be expanded to either *ksiądz* (priest) or *książe* (prince). The rules for the disambiguation are similar to those of semantic disambiguation in Machine Translation: they need to take into account the meaning of the context.

The lexicon intended for Machine Translation should include all inflected forms of the entry words. The same applies to entries in the normalization lexicon. The lexeme represented by the basic form *dr* (doctor, PhD) should include inflected form of the abbreviation because in the majority of spoken contexts the abbreviations are inflected. Moreover, three ways of denoting the inflection occur in real texts: with a dot, with the ending preceded by the hyphen and with the ending without the hyphen. For example, the dative case of the expression: *dr Kowalski* may have the form: *dr Kowalskiego*, *dr. Kowalskiego*, *dr-a Kowalskiego*, *dra Kowalskiego*.

Lexical analysis

Lexical analysis in Machine Translation searches for tokens of the text in the lexicon of the system and "memorizes" the found information to use it in further steps. In case a token is not found, lexical analysis calls special procedures in order to pose hypotheses about the syntactic and semantic characteristics of the unknown token.

Lexical analysis plays a similar role in text normalization. For example, lexical analyzer finds the word *Kowalskiego* in the lexicon and memorizes the fact that it denotes a surname in genitive. The information about abbreviations is also loaded into memory, e.g. the information that the first two entries for the word *p.* usually precede a name, one entry being of masculine gender, the other – feminine.

A lexicon intended for Machine Translation should contain lexical phrases (terms, fixed expressions, idioms) and lexical analysis should identify their occurrences. The same takes place in the normalization process: entries such as *art. mal.*, *pod kier.*, *pod red.* should be included in the dictionary and then identified in the lexical analysis, so that they can be expanded respectively into: *artysta malarz* (painter artist), *pod kierunkiem* (managed by), *podredakcją* (edited by). This is important because the abbreviated words in default contexts are expanded differently: *art → artykuł* (article), *kier. → kierownik* (manager), *red → redakcja* (editors).

Let us notice that the mechanism for lexical phrases allows for the conversion of the string *ul. B. Chrobrego* (B. Chrobrego Street) into *ulica Bolesława Chrobrego* (Bolesława Chrobrego Street) provided that the lexicon is supplied with the full names of famous people.

Tokenization and sentence segmentation

One of the aims of the tokenization process is to determine the meaning of the punctuation marks, particularly dots. The dot may end the sentence – then it is treated as a separate token, it may serve to denote the abbreviation (e.g. *p.*) – then it is treated as a part of the abbreviation, or may have the both functions simultaneously (e.g. *itd.* – Eng. etc.).

The process of tokenization in normalization is as important as it is in Machine Translation. Tokenization takes into account the list of abbreviations prepared in the normalization lexicon. The abbreviations ending with the dot play a crucial role in the next step of text processing – sentence segmentation.

There are at least three types of abbreviations ending with a dot:
Type-1: abbreviations that do not end the sentence, e.g. *p.*, *zam.* (living in)
Type-2: abbreviations that usually end the sentence, e.g *itd.* (etc.), *itp.* (and alike)
Type-3: the abbreviations that may end the sentence, e.g. *ub. r.* (last year), *r.* (year), *ub. m.* (last month), *m.* (month).

The lexicon for both normalization and Machine Translation should include the information on the type of the abbreviation and the algorithm for sentence segmentation should take the information into account. One of the possible treatments of the abbreviations by the segmentation algorithm may be the following:

1. never divide the sentence after abbreviations of Type-1,
2. use standard rules for segmentation for abbreviations of Type-2 (e.g. divide after dot and before a capital letter),
3. use standard rules after Type-3 abbreviations but do not segment before numbers (like e.g. in the expression *w r. 2006* (in year 2006)).

In text normalization, it is crucial that the tokenization process should distinguish between types of numerical expressions. Numerical expressions such as telephone numbers, dates, time, ZIP codes, vehicle license numbers should be identified in the text and handled in a special way because their pronunciation very much depends on what they represent. For example, the numerical string *0(48)12 628 24 30* most likely represents a telephone number and should be expanded into the Polish equivalent of *zero, forty-eight, twelve, six hundred and twenty-eight, twenty-four, thirty* rather than be read digit by digit. The same applies to the string *22-02-2006*, which represents a date and should be pronounced *dwudziesty drugi lutego dwa tysiące sześć* (twenty-second of February two thousand six). The distinction between types of numerical expressions may be made by providing a symbolic representation (e.g. in terms of regular expressions) [14] of each numerical expression in question. Table 1 shows examples of various types of expressions as well as their representation by regular expressions.

The expression for telephone numbers says that such a number consists of at least seven digits with spaces or hyphens in between. The preceding context may be helpful: telephone numbers are usually preceded by an abbreviation *tel.* written with or without a dot. A telephone number (without the directory prefix) should be read: triple, pair, pair, e.g. *426 10 12* will be expanded into the Polish equivalent of *four hundred and twenty-six, ten, twelve.*

The disambiguation of Polish ZIP codes is easy. They always fit the pattern: two digits followed by a hyphen or space followed by three digits. They are read: pair, triple, thus the way of pronunciation is indicated by the delimiting punctuation.

The expansion of dates into text is somewhat more complicated and requires a list which includes full month names, abbreviated month names (e.g. *[Cc]ze* – the first three letters of the word *czerwiec*, Eng. June) and month numbers. Digits representing day and year should be expanded into ordinal numbers. The date may be preceded by the word *dnia* or *dniu* (day) and followed by *r.* or *roku* (e.g. *w dniu*

Table 2. Numerical expression in Polish texts

expression type	examples	symbolic representation (regular expression)
telephone numbers	(0-33) 82-82-826 0 801 555 555 0-(prefix)-52-32-60-725 (48-56) 660-71-77 0048/12/616-21-04 061 426 10 12	`(tel\. ?)?\+?([0-9]{0,4}[-/ ])? ...`
ZIP codes	61-255	`[0-9][0-9][- ][0-9][0-9][0-9]`
car licence numbers	WX 0025 PO 0223X PZA 121Y	`[A-Z]{2,3} [0-9]{3,4}[A-Z]`
dates	1 VII 2006 r. 12.XI.1999 9-VII-1999 dnia 24-07-1995	`(dni(a|u) )?([0-2]?[1-9]| ...`
pesel	49040501580	`[0-9]{11}`
NIP	889-10-16-508	`[0-9]{3}[- ][0-9]{2}[- ][0-9]{2}[- ][0-9]{3}`
time	godz. 21:35:00	`(godz\.?|g\.)?([01]?[0-9]|[2][0-4])([:.] ...`

20 października 1995 r. will be pronounced as the equivalent of *on the twentieth of October nineteen ninety-five*).

The PESEL numbers (national identification numbers) are made up of 11 digits with no spaces or delimiting punctuation in between. They are read: 4 pairs followed by a triple, e.g. *80052705300* will be expanded into the Polish equivalent of *eighty, zero five, twenty-seven, zero five, three hundred*.

Pairs of digits separated by dots or colons and expressing time are expanded into ordinal numbers and read pair by pair, e.g. *21:35:00* will be expanded into *dwudziesta pierwsza trzydzieści pięć* (English literal translation: twenty first thirty five).

Syntactic analysis

Syntactic analysis makes it possible to link words into phrases (sentence components), which then allows to determine inflected forms of abbreviations.

Let us follow the steps of the syntactic parser used to analyze examples given in Table 1.

dla p. Kowalskiego The parser "knows" from the lexical analysis that *Kowalskiego* is a surname and that the abbreviation *p.* should be linked into one sentence component with a surname only if it represents the entry "translated" into *pan* or *pani*. The analyzer chooses the masculine entry *pan* because of the agreement of the genders between the abbreviation and the name.

dla p. Kowalskiej – the same procedure will choose the entry translated into *pani*.

zdobył 5 p. The abbreviation does not precede a name – it should not be interpreted in one of the two ways mentioned above. The parser attempts to link it with the preceding numeral (*5*) and therefore it would choose a nominal interpretation: either *punkt* or *piętro* – excluding *patrz*. The syntactic analyzer will not find a clue which interpretation to choose – this is left for semantic disambiguation.

3-e p. The ending *-e* defines the numeral *3-e* as an ordinal numeral in neutral gender. The only interpretation of the abbreviation that occurs in neutral gender is *piętro* and this interpretation will be linked to the numeral by the syntactic parser.

p. [1] The parse looks for a grammar rule that will make it possible to form a phrase out of *p. [1]*. In the absence of such a rule the parser will assume *p.* to be interpreted as a separate component, which corresponds to the expansion *patrz*.

Syntactic disambiguation

In case of any ambiguity (e.g. such as caused by various interpretations of an abbreviation), the syntactic parser returns all possible parses – even unlikely ones. It is up to the syntactic disambiguation to choose the parse which is most likely to be meant by the author. The algorithm for syntactic disambiguation in TRANSLATICA is described in [13]. In short, the algorithm consists in scoring the alternative parse subtrees during syntactic analysis and then selecting the highest–scored parse tree in the disambiguation phase.

The same method may be applied to normalization. Syntactic disambiguation may be helpful in finding the priority of one "spoken form" above the other, e.g. to choose the interpretation of *m 7* (see the example of a normalized sentence) into *mieszkania siedem* (Eng. lit. apartment seven) over *mieszkania siódmego* (Eng. lit. apartment seventh), although both interpretations are theoretically correct in Polish.

Semantic disambiguation

In Machine Translation, semantic disambiguation allows (among other things) for the determination of one of possible equivalents of the word listed in the lexicon. This is usually executed by examining the context, i.e. the text surrounding the word in question. Similar ideas may be proposed for normalization. If a context surrounding the expression *5 p.* is connected with sport or games of any type, the semantic disambiguator will choose the interpretation *pięć punktów* (five points), whereas the "building" context will force the disambiguator to choose *piąte piętro* (fifth floor).

Transfer

In Machine Translation, the transfer phase is used for conversion of a parse tree of a source sentence (expression) into a parse tree of an equivalent sentence.

In text normalization, the transfer phase may be used for the "translation" of numerals. The numbers should be interpreted in one of the following ways:

A ordinal numbers,
B cardinal numbers read in a standard way,
C numbers read digit by digit, e.g. identity card number, national economy register number,
D numbers read pair by pair, e.g. numerical expressions representing time, national identification number,
E numbers read triple by triple e.g. 9-digit telephone numbers,
F combined way, e.g. zip-codes, telephone numbers, tax identification number.

The tokenization process should determine types C, D, E, F. Syntactical steps should determine whether a number is cardinal or ordinal (Type A or B). Once the type of the number is determined, it is up to the transfer phase to expand it properly.

It may look that a transfer mechanism is a too powerful tool to be used for mere conversion of numbers into literals. It will not be worthwhile to develop such a mechanism especially for the task. But once the tool is at hand, it makes the expansion very easy. It suffices to create simple transfer rules. Here is an example of a rule written in the Perl-like TRANSLATICA transfer language that reads two-digit numerals (function *teen* returns expansions of *-teen* numbers, function emphten returns expansions of *-ty* numbers and function *digit* returns expansions of digits).

```
If (Word =~ /^([1-9])([0-9])$/)
{
    If (Word =~ /^1[0-9]$/)
    {
        Return teen(Word);
    }
    $ten = $1;
    $unit = $2;
    Return ten($ten) . ' ' . digit($unit);
}
```

The transfer phase returns basic forms of numerals. Inflection is left for inflectional synthesis.

Syntactic transformation

Syntactic transformation is a step in Machine Translation that assures the syntactic correctness of the output. It may consist for example of changing the order of the components in the output sentence or assuring the grammatical agreement between sentence components. For example, in translation into English the phase will assure the consistency of tense between the main clause and the subordinate clause.

In normalization, syntactic transformation of the output makes it possible to link the "distant" words, like *Kowalskiego* and *zam.* (living in) and assure that the abbreviation would be expanded to the same case and gender as the name, in this example: genitive masculine.

Inflectional synthesis

The inflectional synthesis of the output generates inflected forms of the "translated" expressions. This means, that for each expanded word, the synthesis phase should generate its form according to the information gathered in previous steps. For example, the syntactic transformation determines the abbreviation *zam.* as being the genitive singular of the word *zamieszkały*. The synthesis will generate the form *zamieszkałego*.

Conclusion

The above ideas are being implemented in a text-to-speech application. One of the aims of developing the application has been to enable the spoken output of the translation system, TRANSLATICA. It has turned out that one of the necessary modules of the speech synthesis, text normalization, may be easily developed using the mechanisms intended for Machine Translation.

References

1. Speech Synthesis Markup Language (SSML), 2004, Version 1.0. W3C Recommendation, 7 September 2004, `http://www.w3.org/TR/speech-synthesis/`
2. Sproat, R. (1996), *Multilingual text analysis for text-to-speech synthesis,* Journal of Natural Language Engineering, **2**(4):369–380.
3. Xydas G., Karberis G., Kouroupetroglou G. (2004). *Text Normalization for the Pronunciation of Non-standard Words in an Inflected Language,* In: Proceedings of the 3rd Hellenic Conference on Artificial Intelligence (SETN04), Samos, Greece, May 5–8, 2004.
4. Yarowsky D. (1993), *Text normalization and ambiguity resolution in speech synthesis,* In: The Journal of the Acoustical Society of America, Vol. **94**, Issue 3, p. 1841.
5. Reichel U. D., Pfitzinger H. (2006), *Text Preprocessing for Speech Synthesis,* In: TC-STAR Workshop on Speech-to-Speech Translation, June 19-21, 2006, Barcelona, Spain.
6. Black, A. Sproat R., Chen S. (2000), *Text normalization tools for the Festival speech synthesis system* `http://festvox.org/nsw`
7. Realspeak `http://www.nuance.com/realspeak/`
8. Ivona `http://www.ivo.pl/`
9. Acapela `http://www.acapela-group.com/`
10. The IMS German Festival Manual (2001), Institute for Natural Language Processing, University of Stuttgart, `http://www.ims.uni-tuttgart.de/phonetik/synthesis/`
11. Oliver D. (1998), *Polish Text to Speech Synthesis,* Master Thesis, University of Edinburgh 1998.
12. Marasek K. (2003) *Synteza Mowy: przegląd technologii i zastosowań za szczególnym uwzględnieniem języka polskiego,* Polsko-Japońska Wyższa Szkoła Technik Komputerowych, Warszawa 2003.
13. Jassem K., Graliński F., Wypych M. (2003), *Statistical and Heuristic Approach to Meaning Disambiguation in POLENG MT System,* In: Demenko G. (ed.) Analiza, synteza i rozpoznawanie mowy w lingwistyce, technice i medycynie, SZCZYRK 2003, Polskie Towarzystwo Fonetyczne.
14. Mikheev A.(2004), *Text segmentation,* In: Mitkov R. (ed.) The Oxford Handbook of Computational Linguistics, Oxford University Press 2004.