Ortfon2 - tool for orthographic to phonetic transcription

Dawid Skurzok, Bartosz Ziółko, Mariusz Ziółko

Depatment of Computer Science, Electronics and Telecommunications
AGH University of Science and Technology
Kraków, Poland
(http://dsp.agh.edu.pl)

Techmo sp. z o.o.
Kraków, Poland
(http://techmo.pl)

{skurzok, bziolko, ziolko}@agh.edu.pl

Abstract

We present a new tool for orthographic to phonetic transcription for Polish. This tool was designed to be easy to use, efficient and flexible. Its interface is aimed to be natural for non-technical people and is presented in details. Some technical implementation details are discussed. We show speed and memory usage benchmarks and compare them to the old version.

1. Introduction

Grapheme to phoneme transcription is widely used in Text-To-Speech (Black et al., 1998) (Hunt and Black, 1996) and Automatic Speech Recognition (ASR) systems (Young et al., 2005) (Ziółko et al., 2011). There are two general methods of performing such transcription automatically: rule based and data driven. Rule based method are expert systems which are based on context decisions provided by linguists.

In case of some languages, like English, it is very irregular, so the pronunciation dictionaries are difficult to build. In case of others, including Polish, there are rules allowing to program it. However, it has to be stressed, that the number of rules is huge and using them is not obvious. The rules were published by Steffen-Batog (Steffen-Batóg, 1975) and later used in two systems: Polphone (Demenko et al., 2003) and Ortfon version 1.

There are many language-dependent transcription tools. For example, TOPH system (Ghneim and Aubergé, 1995) is a letter-to-phone software developed for French in nineties. TOPH is a language deterministic grammar with horizontal and vertical links. The software was designed mainly for spelling errors correction rather than for ASR like Ortfon2. Earlier, the topic of letter-to-phone transcriptions was described by Auberge in his PhD thesis (Aubergé, 1991). A system specialised for Russian pronunciation was also implemented (Krivnova, 1990), as a part of text-to-speech software. In case of English, due to a vast number of irregularities in pronunciation, approaches based on dictionaries like BEEP dictionary or http://www.photransedit.com/ seem to be more popular than a letter-to-phone converters. In case of Spanish, a system based on context-dependent rewrite rules and an exception mechanism is in use (Sandoval et al., 2008). It operates on word units.

One of our aims while designing a new transcription tool was to create both fast and flexible system while preserving easy and intuitive interface. Many existing tools for language processing are written using some scripting language. It significantly degradates speed efficiency and creates many problems during integration with systems created in compiled languages, especially in embedded system. Moreover some high secure environments e.g. internal police systems prohibit installation of any scripting language.

Our tool is designed to work as a part of Automatic Speech Recognition system. Such systems require high speed of data processing. Most of ASR systems are created using C or C++, some of them use other languages but only for research and initial algorithms design. We chose C++ as an implementation language to make our tool fast and easy to integrate with ASR system (Ziółko et al., 2015). To provide simple and easy interface for rules editors we created a small declarative language on top of a Comma Separated Values file format. File in this format can be created in every spreadsheet editor. Expressions of our language are just to be placed in the cells of spreadsheet. A natural table format of editors lay these expressions in readable manner. To use in Ortfon2, CSV file must be compiled and converted to an internal file format. Compilation process optimise much of the transcription rules to minimise its further processing.

In the next section of this paper we introduce formalism to define transcription rules, including defining a set of orthographic sequences in compact manner. In the next section we present some typical work flow of the text processing. Results of performance and efficiency evaluation will be presented in section 4. Finally, section 5 presents our conclusions from such approach on this work.

All examples in this paper uses transcription rules presented in (Steffen-Batóg, 1975). These rules describe Krakow-Poznań version of pronunciation.

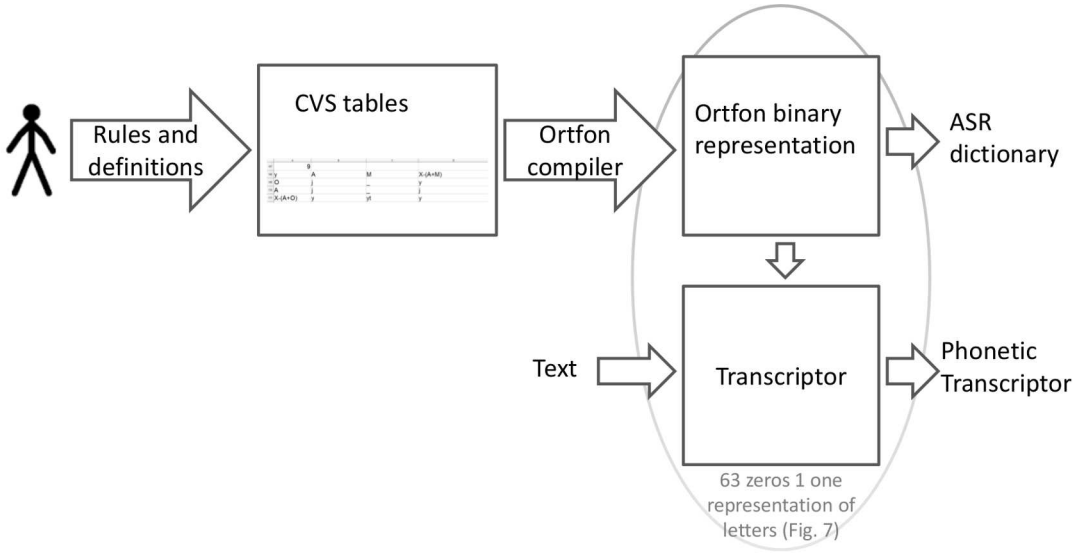

Figure 1: General diagram of Ortfon. The main difference with other ortophonetic transcription systems is using two representations of transcriptions - user friendly one and a special binary format consisting of 64 bits, where letters are represented in a flag mode to speed up all calculations and comparisons of expressions (Fig. fig:binaryRepresentation)

```
d{z,ż}i
RV
J+Ś+ZJ
(M'-ń)(X-J)
(1+T+R+M')K
{ą,ę,i,u,y,#}+(X-n){a,ienа,ieo}+(X-
{#,e,o})na+(X-i){ena,eo}+(X-p)
{ona,rze}+(X-{#,p,e})o+(X-#)p{o,rze}+
(X-z)e+(X-r){ze,ó}+(X-ś)ró
```

Figure 2: Example of orthographic expressions to be used in Ortfon

2. Transcription Rules

2.1. Representing orthographic sequences

As a basic building block of transcription rules we use the so called orthographic expression, i.e. formulas representing a set of letter sequences (fig. /ref-fig:OrthographicExpressions). Orthographic expressions are algebraic-like formulas with special definition of operators +, - and concatenation of expression elements. These constructs were introduced and formally described by (Steffen-Batóg, 1975). Most of the orthographic expressions use predefined sets of orthographic letters. Such sets are defined using upper case letter to distinguish from lower case which are used as standard, single letters. Such sets usually denote letters which will produce a phoneme with similar features during transcription (e.g. fricative, voiced, etc.). E.g. capital *A* stands for vowels: *a, ą, e, ę, i, o, ó, u, y*.

Sum of letters and sets of letters are defined in terms of algebraic sum of sets and use operator **+** or in case of summing just alphabetical letters, every addend is written separated with coma. E.g. *a+b+c* is equivalent to *{a, b, c}*.

Algebraic multiplication of elements in a formula means concatenation of expressions representing by these elements. We do not support any multiplication sign. Instead we provided braces to denote concatenation of sums. E.g. *a(b+c)* stands for sequences *ab* + *ac*, i.e. letter *a* followed by letters *b* or *c*; If *J* defines *{i, j}* then *Ja* can be unfolded as sequences *ia+ja*.

Subtraction requires a special comment because it may be unintuitive for some readers. In most cases subtraction is used to exclude some letters from a given set, for example if set *S* is defined as *{a, b, c}* then *S-a* means set *{b, c}*. It must be emphasized that e.g. *(a+b)S - bb* do not mean *a{a,c}*, but "*(a+b)S and not bb*". We decide to not implement support for such expressions so sequences longer than one element can not be subtracted. Although it may be useful in some cases, it would complicate matching engine. Moreover, a need for subtraction can be eliminated. To dispose such expression one must decompose it to minuend and subtrahend. Then move them to different columns (or rows) such that subtrahend is before minuend. In such layout subtrahend sequence will be checked first during a context matching test for given letter. If subtrahend will match given orthographic context phoneme for this expression will be used and analysis for given letter will be terminated. Subtraction of sequences was very rarely used in transcription tables (Steffen-Batóg and Nowakowski, 1993) we use.

There are special cases when sequence subtraction is implemented: when minuend is single set definition of length equals to 1. In such case, minuend is extended to the length of subtrahend by appending another copy of itself. E.g. *A - ab* is equivalent to *(A-a)(A-b)*.

Orthographic alphabet also contains neutral symbol (identity symbol) for concatenation *1*, which is used to shorten expression. E.g. *(1+a+b)c* is equivalent to *c+ac+bc*.

In transcription tables from (Steffen-Batóg and

	A	B	C	D
167		9		
168	y	A	M	X-(A+M)
169	O	j	_	y
170	A	j	_	j
171	X-(A+O)	y	yt	y

Figure 3: Example of table definition in CVS file. 9 is the index of the table, y is the considered letter, O, A, X-(A+O), M, X-(A+M) are contexts and j, y and yt are possible pronuniciation

Nowakowski, 1993) there are 2 cases when such construct was used, but this problem was solved by simple table rearranging.

To parse an orthographic expression and try to match it with a letters sequence, one needs a context-free grammar parser because of use of letters sets definitions. However, all expressions can be transformed easily to the regular expression form. Ortfon2 transforms all expressions to the sets of alternative sequences, and these sequences are used to check if given input sequence matches a prefix or a suffix of a given letter.

2.2. Tables format

For each letter of orthographic alphabet one must define a table or few tables with transcription rules [Fig. 3]. The first, single cell in the table denotes an order number of the table. Than the first cell in the second row is a letter for which this table may be used. First column in the table contains prefixes for this letter and first row consists of suffixes, both in form of orthographic expressions. Every other cell contains a phoneme, a sequence of phonemes, a neutral symbol "*1*" or an empty symbol "_". We decided to use a special symbol for empty cells to simplify the parsing process and to reduce a chance of typing an error. The empty symbol means that given prefix and suffix of letter is forbidden, i.e. it should not occur in the language. Every table has also a number specified by a user which can be used while debugging. Phonetic neutral symbol means that a given letter does not produces a phoneme.

2.3. Defining transcription rules

At the top of the file, one may want to define sets of letters for further usage in orthographic expressions. To define a set, one needs to put the word *def* in the first cell of the row [Fig. 4], a unique name in the second cell and an orthographic expression in the third cell. The *def* command can be placed anywhere in the file, so it may be used to define orthographic expression used in only one table. The order of the rows in the tables is relevant, because the system stop searching for a context after finding the first one which fits.

Another command is *phonetic* which defines phonetic symbol [Fig. 5]. To use it, one must place the word *phonetic* in the first cell of the row, name of the phoneme (ASCII) in the second cell and IPA and Slavic symbols (UTF-8) in the next cells. All phonetic symbols definitions must be placed in consecutive rows before first usage of any phoneme, so technically it must be before any table. We also implement simple comment format: every row started by "--" in the first cell is skipped by the parser.

	A	B	C	D
1	def	A	{a,ą,e,ę,i,o,ó,u,y}	
2	def	A'	A-{i}	
3	def	M	{m,n,ń,j}	
4	def	M'	M-{j}	
5	def	J	{i,j}	
6	def	R	{l,ł,r,w}	

Figure 4: Example of *def* usage. The second column are unique names of expressions introduced by the user. The third column are orthographic expressions (for example letters) attached to the unique name

	A	B	C	D
28	phonetic	et	ɛ̃	ẽ
29	phonetic	etd	ɛ̃	ẽ
30	phonetic	a	a	a
31	phonetic	ap	ą	ä

Figure 5: Example of *phonetic* usage. The second column are names of the phonemes (for example 'et'). The third column are IPA symbols related to that phoneme, and the last one symbols from the Slavic alphabet

We also provide a special command *merge* used mainly in ASR system development [Fig. 6]. It is used to merge a set of phonemes into one. Technically, it just replaces all phonemes from the set to another. Such operation may be performed by any text editor, but having this command makes developing of a transcription system more convenient. It keeps the tables more readable and consistent between versions. It can also be used to change the ASCII name of a single phoneme.

Stress assignment and syllabification are not considered in the existing rules for Polish. However, they could be implemented due to syntax used in Ortfon.

3. User interface

Ortfon2 is divided into two applications: a parser which is used to read a CSV table file and compile it to internal binary format and an actual program that uses compiled tables and produces transcription. Ortfon2 can be also compiled with build-in tables for simplicity of deployment and usage.

The parser accepts only CSV files with tables as input, but users are allowed to use more complex file formats for their convenience. We use LibreOffice and its native Open Document Format as main tool for editing and storing tables, because of its good support of Unicode and IPA alphabet. Tables can be exported to CSV format just before rules compilation. Such an approach also allows us to use rich editing and marking tools like colouring, version

	A	B	C
119	merge	w,wd,wt	w
120	merge	l,ll,lj	l
121	merge	r,dr,r'	r
122	merge	m,md,m'	m

Figure 6: Example of *merge* usage. For example in the first row phonemes 'w', 'wd' and 'wt' are merged into one with name 'w'

```
Can sequence 'abc' belong to expression
'X(b+c)(a+b)' ?
abc = (001)(010)(100)
X(b+c)(a+b) = (111)(110)(011)
(001)(010)(100) &
(111)(110)(011) =
(001)(010)(000)
 yes  yes  no!
```

Figure 7: Our binary representation of letters as a sequence of several zeros and single one allows much faster arithmetic calculations than in case of a typical representation

control system and comments which is very useful in the development stage of transcription tables, especially when more than one person works with the tables.

4. Implementation

While previous version of Ortfon was just a transcription tool with build-in rules, Ortfon2 is a universal tool for creating transcription systems.

Ortfon2 is written exclusively in portable C++. It uses its own specialised parsing engine to compare text input with an orthographic expression. It may produce a transcription using an ASCII name for symbols, as well as UTF-8 encoded IPA and Slavonic symbols. A file with transcription tables in compiled form has around 100-150 kB, so its size is negligible. Ortfon2 accepts raw text as input, and produce transcription encoded in IPA, Slavonic symbol or ASCII characters string.

There are two major reasons for implementing own expression matching engine instead of using some standard regular expression library. Our engine besides of expression to text matching, allows also expression to expression matching which is used to create a compact hidden Markov model of a word. Second reason is that our library is smaller than general purpose libraries which may be important in embedded system.

Orthographic expressions are compiled to a set of sequences of letters sets. We took advantage that there are less then 64 letters in Polish alphabets to store these sets in a compact manner. Each set of single letters is stored as a 64 bit integer number. Each bit in that number corresponds to one letter. So a sequence of letters sets is stored as a sequence of numbers. For example if a = 001, b = 010, c = 100 than a(b+c) = (001)(110).

Such form of data layout allows also very fast testing if given letters belongs to a set. Such checking is a basic operation of the matching engine and in such data format it requires translation to one machine instruction. A dedicated regular expression match algorithm is optimized to most common form of orthographic expression.

Currently, Ortfon2 works only with Polish alphabet, but it is possible to extend it to work with any, user-defined alphabet. It is also possible to further speed up the expression matching engine, but it is not needed currently.

4.1. Evaluation

We used transcription rules from Steffen-Batogowa work (Steffen-Batóg, 1975) for the correctness evaluation. The quality of transcription was not measured traditionally by comparing results of automatic and hand-written transcription. Ortfon2 was created mainly for SARMATA ASR system (Ziółko et al., 2011), so the quality of transcription was measured by evaluating ASR system performance. The ASR using first version of Ortfon recognised correctly 95% of phrases, while using the second version recognised correctly 97%. The test was performed on 10 000 phrases. Every phrase was transcribed as whole, so letters on words borders had context of another word applied.

The speed test comparison with previous version of Ortfon was conducted. Ortfon2 transcribes more than 36 thousands of words per second, while the first version of Ortfon transcribes less then 8 thousands words per second, which is 4.5 times slower. Moreover, the first Ortfon uses much simpler rules of transcription. It uses only 37 phonemes which makes its transcription tables smaller and should actually result in shortening of the processing time. The speed gain of Ortfon2 over old Ortfon is mostly due to the implementation of orthographic expression matching engine. First Ortfon uses general purpose regular expression library PCRE (Perl Compatible Regular Expressions), while Ortfon2 uses its own engine tailored exclusively for this purpose. Ortfon2 uses approximately half of memory used by its predecessor, but the absolute amounts are negligible: Ortfon2 uses 1.7MB and the first version uses 3.5MB of memory.

5. Conclusions

We plan to implement omni-context transcription generation which may be useful in ASR systems. Such process would generate a set of transcriptions of a word, theoretically placing a given word in context of any other word in the language. Our implementation of orthographic expression allows fast expression-to-expression matching which may be used while implementing such feature.

Ortfon2 is a very successful programing attempt to make a phonetic transcription system based on existing knowledge of phonetics. Its implementation is much more effective than previous one and it is not possible to make it any faster with existing programming technologies. It has also another advantage, which is easy and user friendly method for introducing additional linguistic data to its database. Ortfon is a ready for use tool which was applied in ASR system. It is available on a license.

Acknowledgments
This work was funded by National Science Centre allocated on the basis of the decisions DEC-2011/03/D/ST6/00914 (Bartosz Ziółko and Dawid Skurzok) and DEC-2011/03/B/ST7/00442 (Mariusz Ziółko). We would like also to thank Jan Wicijowski, the author of the first version of Ortfon, which we used as a baseline and inspiration.

6. References

Aubergé, V., 1991. La synthese de la parole: des regles aux lexiques. *PhD Thesis, University of Grenoble*.

Black, A.W., K. Lenzo, and V. Pagel, 1998. Issues in building general letter to sound rules. *The Third ESCA Workshop in Speech Synthesis*.

Demenko, G., M. Wypych, and E. Baranowska, 2003. Implementation of grapheme-to-phoneme rules and extended SAMPA alphabet in Polish text-to-speech synthesis. *Speech and Language Technology, PTFon, Poznań*, 7(17).

Ghneim, N. and V. Aubergé, 1995. Optimising tools for the french letter-to-phone grammar toph with a view to phonographic spelling correction. *Proceedings of ROCLING*:55–63.

Hunt, A. and A. Black, 1996. Unit selection in a concatenative speech synthesis system using a large speech database. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 1996. ICASSP-96*, 1:373 – 376.

Krivnova, O.F., 1990. Automatic synthesis of Russian speech. *Proceedings of the XIV International Congress of Phonetic Sciences*, 1:507–510.

Sandoval, A. M., D.T. Toledano, R. de la Torre, M. Garrote, and J.M. Guirao, 2008. Developing a phonemic and syllabic frequency inventory for spontaneous spoken castilian spanish and their comparison to text-based inventories. *Proceedings of LREC*:1097–1100.

Steffen-Batóg, M., 1975. *Automatyzacja transkrypcji fonematycznej tekstów polskich (Eng. Automation of phonetic transcription of Polish texts)*. Warszawa: PWN.

Steffen-Batóg, M. and P. Nowakowski, 1993. An algorithm for phonetic transcription of ortographic texts in Polish. *Studia Phonetica Posnaniensia*, 3.

Young, S., G. Evermann, M. Gales, Th. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, 2005. *HTK Book*. UK: Cambridge University Engineering Department.

Ziółko, B., T. Jadczyk, D. Skurzok, P. Żelasko, J. Gałka, T. Pędzimąż, I. Gawlik, and S. Pałka, 2015. SARMATA 2.0 automatic Polish language speech recognition system. *Interspeech, Dresden*.

Ziółko, M., J. Gałka, B. Ziółko, T. Jadczyk, D. Skurzok, and M. Mąsior, 2011. Automatic speech recognition system dedicated for Polish. *Proceedings of Interspeech, Florence*.