Bermuda, a data-driven tool for phonetic transcription of words

Tiberiu Boroş, Dan Ștefănescu, Radu Ion
Research Institute for Artificial Intelligence, Romanian Academy (RACAI)
Calea 13 Septembrie, nr. 13, Bucureşti, România
E-mail: {tibi, danstef, radu}@racai.ro

Abstract

The article presents the Bermuda component of the NLPUF text-to-speech toolbox. Bermuda performs phonetic transcription for out-of-vocabulary words using a Maximum Entropy classifier and a custom designed algorithm named DLOPS. It offers direct transcription by using either one of the two available algorithms, or it can chain either algorithm to a second layer Maximum Entropy classifier designed to correct the first-layer transcription errors. Bermuda can be used outside of the NLPUF package by itself or to improve performance of other modular text-to-speech packages. The training steps are presented, the process of transcription is exemplified and an initial evaluation is performed. The article closes with usage examples of Bermuda.

Keywords: grapheme-to-phoneme, letter-to-sound, phonetic transcription, text-to-speech, data driven

1. Introduction

The last years have brought about a dramatic increase in the performance of human-computer interaction tools and techniques. This has naturally led to their successful application in Information-Technology and related fields. Consequently, accessibility to digital resources for elderly or disabled people is enabled by diverse methods such as better text organization and navigation, improved text input methods or better text reading using text-to-speech tools.

We present the *Natural Language Processing Unified Framework* (NLPUF) for text-to-speech (TTS) synthesis, which is part of the deliverables within the METANET4U project[1]. It comprises of a set of NLP tools and a speech synthesis module that can all be used together or as standalone packages. Its functionality consists of text normalization, phonetic transcription, homograph disambiguation, prosodic synthesis and speech synthesis, each of the functions being performed by different tools in the package. The speech synthesis component uses concatenative unit selection and can be easily integrated with other speech synthesis engines such as MBROLA (Dutoit et al., 1996) or HTS (Zen et al., 2007).

NLPUF is under development at the moment, but it is nearing completion. Before a TTS system can synthetize voice starting from arbitrary text, certain tasks have to be performed by the Natural Language Processing (NLP) module of the TTS system. The NLP module deals with text normalization, phonetic transcription, prosody analysis etc. Text normalization refers to the expansion of acronyms, abbreviations, numeric or mathematical expressions, etc., while prosody analysis tries to learn how to mimic speech phenomena such as rhythm, stress and intonation starting from text (Huang et al., 2001).

In this paper we focus only on the *phonetic transcription* (PT) for out-of-vocabulary (OOV) words and the way PT can be used to improve text accessibility. The phonetic transcription of words can be obtained using lexicons for known or common words in a target language, but there will always be OOV words (technical terms, proper nouns etc.) regardless of the lexicon's size. In this situation, the system needs a method to predict OOV words' pronunciation. This is one of the fragile steps of the pre-processing and analysis of text, because errors produced by incorrect transcription predictions can accumulate with errors from other modules (this is known as *error propagation*) on the way to the speech synthesizer (the part of the TTS that is responsible for the actual voice synthesis), leading to misreads of the original message.

Also, presence of foreign words inside the text (a common issue in any type of text: news, novels, technical articles etc.) increases the complexity of the problem. Thus, phonetic transcription of OOV words would greatly benefit from language identification, which is still an unresolved problem for very short texts (da Silva and Lopes, 2006; Vatanen et al., 2010).

In the case of NLPUF, phonetic transcription of OOV words is performed by *Bermuda*, a data-driven tool that uses Machine Learning (ML) techniques to best fit a phonetic transcription given an input word. As any other ML approach, it uses features, which in this case are based solely on the letters and groups of letters within the input word. While using more context sensitive data (part of speech, syllabification etc.) may provide better results in some cases, we intend to show that state of the art results can be obtained without using such data. Such an application is therefore much faster and does not require additional resources. Moreover, homograph disambiguation is not an issue here. Bermuda deals only with OOV words, which means it is impossible to predict that such words have two or more pronunciations that distinguish between their senses. The task of homograph disambiguation can only be performed on known words and it is handled by a different component in our framework.

[1] www.metanet4u.eu

2. The role of phonetic transcription in improving text accessibility

Phonetic transcription (PT) has an important role in any TTS system. One of the objectives of speech synthesis from text is to allow the user to fully understand the message that is being transmitted. While prosody highly contributes to understanding the message, PT also has a notable impact. Incorrect PT can render an entire fragment meaningless and mispronunciation can lead to annoying results (e.g. the same word is mispronounced again and again in a phrase or paragraph) even if the information may sometimes be transmitted regardless of small erroneous transcriptions. PT errors can also add up to the prosody errors and have a negative impact on the overall system performance.

Spelling correction or query alteration also link to text accessibility when taking into account that most relevant information found on the Internet is written in languages of international use and not all users are native speakers of such languages. Research has shown the possibility of using phonetic similarity as a feature for spelling correction (Li et al., 2006). A misspelled word can be corrected by using its PT. Table 1 shows an example where a misspelled word and its correct form produce identical PTs.

	Word	Phonetic transcription
Correct	Conceive	k ax n s iy v
Incorrect	Conceiv	k ax n s iy v

Table 1: PTs for words "conceive" and "conceiv" produced by Bermuda

In section 8 we show another example where web query alteration can benefit from the PT of words.

3. Related Work

Phonetic transcription in terms of *letter-to-phoneme* conversion (L2P) can be a simple task for languages where the relationship between letters and their phonetic transcription is simple (languages that are preponderantly characterized by having phonemic orthography, e.g. Romanian) but for other languages it poses a set of challenges. For example, current state of the art systems for English phonetic transcription of OOV words have an accuracy of 65% to 71% when used on the CMUDICT dictionary (Jiampojamarn et al., 2008).

There are a series of different methods and approaches to L2P conversion from context sensitive grammars to using classifiers or techniques specific to part-of-speech tagging.

A notable example of using a context sensitive grammar for writing L2P rules (pertaining to English and French) is given by Divay and Vitale (1997), although nowadays automatically inducing L2P rules is the main route followed by mainstream L2P research.

The *Expectation-Maximization* (EM) algorithm (Dempster et al., 1977) (or variants of it) is used to find one-to-one or many-to-many alignments between letters and phonemes in (Black et al., 1998; Jiampojamarn et al., 2008; Paget et al. 1998). The main idea of this algorithm is that, certain pairs of letters and phonemes are much more frequent than others and EM is employed in an effort to automatically detect the most probable alignments given a list of pairs of words and their transcriptions as training data.

Another approach for PT uses Hidden Markov Models (HMMs). Given the L2P rules (i.e. the probability of a phoneme being generated by a letter and the probability of occurrence of a phoneme sequence), the problem of automatic PT can be restated as follows: find the optimum sequence of hidden states (phonemes) that account for the given observation (the OOV word that has been suitably segmented for this task). Research of this approach has been done by Taylor (2005) and Jiampojamarn et al. (2008). One interesting conclusion of their research is that more accurate results are achieved if the phonemic substrings are paired with letter substrings. The reason for this is that phonetic transcriptions are context dependent: at any given moment, the phoneme to be generated is dependent on the adjacent phonemes. Moreover, it also depends on a contextual window of letters of the given word (Demberg, 2007).

4. A general view on Bermuda

Bermuda implements 2 methods for the L2P conversion task. The first one employs a *Maximum Entropy* (ME) classifier (*PTC*) to predict the phonetic transcription of every letter in the context (word) and uses a set of features similar to the MIRA and Perceptron methods presented by Jiampojamarn et al. (2008). The second one uses *DLOPS* algorithm described in Boroş et al. (2012). Furthermore, each of the methods has been improved by employing another ME classifier (*ERC*) designed to correct common L2P errors made by these two methods. In addition to the features used by PTC, ERC uses new features based on the already predicted phonemes which have become available. In other words, Bermuda chains the first layer prediction (PTC or DLOPS) to a second layer ME classifier for error correction (ERC). This leads to an accuracy increase of 2% to 7%.

We aim to show how Bermuda can be used outside of the NLPUF package, as a stand-alone application, to improve performance in other modular TTS packages.

5. Phonetic Transcription as an Alignment Problem

All data-driven L2P systems require letter to phoneme alignment before a model for phonetic transcription can be created. This section presents a method for obtaining such an alignment that is easy to implement. PT can be viewed as a translation process from the written form of the word (the "source language") to its phonetic representation (the "target language") (Laurent et al. 2009). Because aligning between words and phonetic transcriptions is similar to training a translation model, it is possible to use a well-known tool, explicitly designed for this kind of task: GIZA++ (Och and Ney, 2003).

GIZA++ is a free toolkit designed for aligning items in a parallel corpus, often used in the Statistical Machine Translation (SMT) field. Given pairs of unaligned (at word level) source and target sentences, it outputs word alignments within each pair. GIZA++ treats the word alignment task as a statistical problem and, as such, it can be applied to other problems that can be described in similar terms. Rama et al. (2009) showed that GIZA++ can be successfully used to preprocess training data for letter to sound conversion systems.

6. Bermuda training

Before any phonetic transcription can be produced, the system has to be trained. Bermuda accepts two types of files (plain aligned files and GIZA++ output files) as input for the training process.

Each line in the plain aligned files contains a word paired with its PT. Every symbol or set of symbols used for either the encoding of the word (characters/letters) or the encoding of the PT (phonemes) are <SPACE> separated. The paired elements are separated by a <TAB> character. The number of tokens of the elements in each pair must be equal. The word characters, which in reality do not have a corresponding symbol in the PT, are marked with the empty phoneme: "-", designed to preserve the equality (lines 4 and 5 of figure 1). If one word character emits more than one corresponding symbol in the PT, the character "." is used to link together the symbols (line 5 of Figure 1). In some cases, in which more word characters participate in forming a single sound, it is standard practice to associate only the last letter of the word with the PT and assign the empty phoneme to the other letters.

```
a b a n d o n<TAB>ax b ae n d ax n
a b a s i c<TAB>ax b ey s ih k
a b a t e r<TAB>ax b ey t ax r
a b a t t e d<TAB>ax b ae - t ih d
a b u s e r<TAB>ax b y.uw z ax -
```

Figure 1: Plain text training file

One training method for Bermuda is by using the alignment output of the GIZA++ toolkit. We run GIZA++ for a primary letter to phoneme alignment with default parameters (10 iterations of IBM-1, HMM, IBM-3 and IBM-4 models). To do this, the data has to be split into two files, one corresponding to the words (source file) and the second one corresponding to their phonetic transcription (target file). Every word in the source file must be on a single line, and its letters have to be separated by <SPACE>. Every line in the source file has a corresponding line in the target file.

source.txt

```
f l u                          (line 1)
c a u s e                      (line 2)
t w a s                        (line 3)
s h i r e                      (line 4)
a b a n d o n                  (line 5)
```

target.txt

```
f l uw                         (line 1)
k ao z                         (line 2)
t w oh z                       (line 3)
sh ia                          (line 4)
ax b ae n d ax n               (line 5)
```

Before running GIZA++ we make sure it is compiled to be used outside of the Moses MT Toolkit. The following two lines should run successfully on the source and target files:

```
plain2snt.out target.txt source.txt
GIZA++ -S uk_beep.src.vcb -T target.vcb
-C source_target.snt -p0 0.98 -o output
```

One frequent mistake that GIZA++ makes is the forced NULL alignment on the phonemic side. Since an unaligned phoneme must be generated by one of the close-by letters, we devised a simple correction algorithm that looks at the letters that emitted the previous and the next phonemes and links the unaligned phoneme to the letter with which it was most frequently aligned to. In case of ties, it chooses the letter on the left side. Let's take for example the word *absenteeism* (Figure 2). Between the phonetic symbols aligned to S and M that is 'Z' and 'M' respectively, we have the unaligned (or NULL aligned) symbol 'AH'. In this case, we correct the alignment by assigning the phoneme 'AH' to the letter "S" because 'AH' between 'Z' and 'M' was most frequently aligned with 'S' (next to 'M').

The correction algorithm also inserts the empty phoneme for every NULL aligned letter. In Figure 2, the letter at position 8 (bold font) does not emit any symbol and so, we insert the empty phoneme in the PT at the appropriate position.

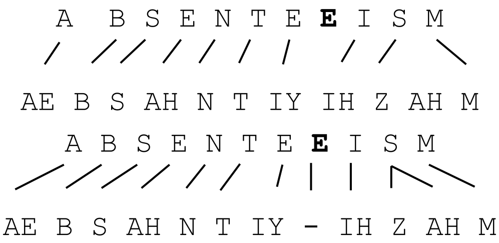
A B S E N T E **E** I S M
AE B S AH N T IY IH Z AH M
A B S E N T E **E** I S M
AE B S AH N T IY - IH Z AH M

Figure 2: Alignment correction

Figure 3 represents an overview of our training process (comprising of letter to phoneme alignments and building models for the primary ME classifier, the DLOPS method and the second layer classifiers). DLOPS is a data-driven method used for generating PTs of OOV words by optimally adjoining maximal spans of PTs found in a given dictionary, corresponding to adjacent parts of the input word (Boroş et al., 2012). This is the case when GIZA++ is used for initial letter to phoneme alignment. If Bermuda receives plain text aligned files, the first two steps are ignored and Bermuda skips directly to training the first-layer methods. After the initial training of the first layer methods, Bermuda runs through the entire training corpora and produces PTs for every word using the two primary prediction methods (PTC and DLOPS). A new set of training data is compiled based on the predictions made by the two methods and the real PTs in the training data. The second layer classifier (ERC) learns to correct the common mistakes of the two first-level methods, improving their accuracy.

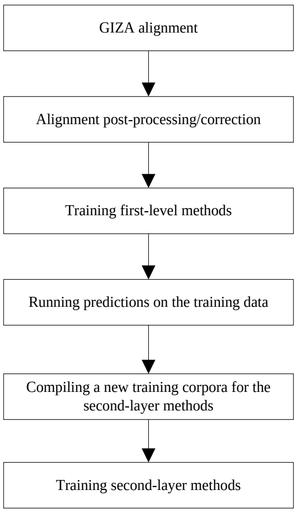
GIZA alignment
Alignment post-processing/correction
Training first-level methods
Running predictions on the training data
Compiling a new training corpora for the second-layer methods
Training second-layer methods

Figure 3*:* Training process

7. Usage and Testing

The current version of the system has been tested on two English dictionaries (BEEP UK – 250k words and CMUDICT US – 130k words) and on a Romanian dictionary extracted from the Romanian Speech Synthesis (RSS) Database (Stan et al., 2011). The training corpus was ten-folded and we ran Bermuda on every set while training on the other nine. The results show maximum performance for the PTC+ERC method as follows: CMUDICT 65%, the BEEP 71% and about 93% on the Romanian dictionary (the PT data for this dictionary has not been manually validated yet).

System	Word Accuracy on BEEP
PTC+ERC	71.31%
PTC	68.16%
DLOPS+ERC	66.40%
DLOPS	64.04%

Table 1: Word accuracy figures for the methods implemented by Bermuda

Table 1 shows an increase of about 2% to 3% in precision when chaining the second layer (ERC). These results are similar to those obtained by state of the art methods.

Once training files are available, Bermuda can be trained using the following lines:

```
bermuda.exe -gizatrain <giza A3 filename>
[-test]
bermuda.exe -plaintrain <plain aligned
filename> [-test]
```

If the *–test* option is specified, Bermuda splits the training corpora using the tenfold method. The data is divided into 10 files (folds), each having approximately 10% of the original corpus. After the split is performed, the tool shows the accuracy obtained on each of the 10 folds while sequentially training on the other 9. Accuracy is measured for each method in particular, so the user will be able to know which one to use in the final implementation.

The following command is used for running Bermuda:

```
bermuda.exe -run -m<1…4>
```

The second argument selects the method that will be used when predicting the PT of a given word. 1 corresponds to DLOPS method, 2 is used for PTC, 3 DLOPS+ERC and 4 means PTC+MRC. After the data for the specified method is loaded, the queries for the PT can be entered. Each letter must be space separated as in the following example:

```
Q:> a b s e n t e e i s m
       AE B S AH N T IY IH Z AH M 0.82%
       AE B S EH N T IY IH Z AH M 0.07%
       …
```

The example above displays results obtained using DLOPS method. This is the only method that currently shows the confidence level for each phonetic transcription variant.

Bermuda also has a custom evaluation method which takes as input a file with the same structure as the plain aligned training corpus and calculates its accuracy based on the data inside. This can be called using the following command:

```
bermuda.exe -customtest <filename>
```

8. Current state and future work

This tool is currently available for online testing and can be downloaded from RACAI's NLP Tools website[2]. The online version is trained for both Romanian (using a proprietary lexicon) and for English (using UK BEEP dictionary). It can produce phonetic transcriptions using any model specified (DLOPS, PTC, DLOPS+ERC or PTC+ERC). The phonetic representation is based on the symbols (e.g. "@" for the Romanian letter "ă") employed by each individual training lexicon, but we plan on mapping these symbols to the International Phonetic Alphabet (IPA) in order to have a unified phonetic transcription system. Referring back to section 2, IPA transcription could improve current query suggestion systems. For example, users would be able to enter queries based on their native perception of the pronunciation of words (write queries in their native language based on their phonetic perception). The system would then be able to map the PT to that of any other language, thus finding the correct spelling suggestion. We call this type of query input *perceptive search* and we plan on doing further research in this area as well. We need to mention that Bermuda can be used to map back phonemes to words by inversing the lexicon files, a task which implies a different technique in order to cope with homophones.

[2] http://nlptools.racai.ro/

9. Conclusions

We have presented a data-driven tool for L2P conversion, which is part of the NLPUF package but can also be used individually. Training and usage of this tool are fully covered in this paper.

Sections 2 and 8 show the role of phonetic transcription in improving text accessibility starting from its integration in TTS systems, spelling correction and/or alteration based on phonetic similarity and the possibility of using letter to phoneme conversion and phoneme to letter conversion for implementing perceptive search.

Our future plans include further development and fine-tuning work on the current methods and a complete set of tests for experimental validation using baselines provided by other L2P systems (e.g. using the same dictionaries as other systems). We also want to map the available dictionaries to IPA and to implement and test a perceptive search method based on Bermuda.

This tool will be free and available for download once the final tests are performed.

10. Acknowledgments

The work reported here was funded by the project METANET4U by the European Comission under the Grant Agreement No 270893.

11. References

Baayen, R., Piepenbrock, R., and Gulikers, L. (1995). The CELEX lexical database. In *Linguistic Data Consortium*, University of Pennsylvania, Philadelphia.

Black, A., Lenzo, K. and Pagel, V. (1998). Issues in building general letter to sound rules. In *ESCA Speech Synthesis Work-shop*, Jenolan Caves.

Boroş, T., Ştefănescu, D. and Ion, R. (2012). Data driven methods for phonetic transcription of words. In the *13th Annual Conference of the International Speech Communication Association* (submitted).

Bosch, A., and Canisius, S. (2006). Improved morpho phonological sequence processing with constraint satisfaction inference. In *Proceedings of the Eighth Meeting of the ACL-SIGPHON at HLT-NAACL*, pp. 41–49.

CMU. (2011). Carnegie Mellon Pronuncing Dictionary. http://www.speech.cs.cmu.edu/cgi-bin/cmudict

Content, A., Mousty, P., and Radeau, M. (1990). Une base de données lexicales informatisée pour le français écrit et parlé. In *L'Année Psychologique*, 90, pp. 551–566.

Da Silva, J. F., and Lopes, G. P. (2006). Identification of document language is not yet a completely solved problem. In *Proceedings of CIMCA'06*, pp. 212–219.

Dempster, A.P., Laird, N. M. and Rubin, D.B. (1977). Maximum likelihood from in-complete data via the em algorithm. In *Journal of the Royal Statistical Society*: Series B, 39(1), pp. 1–38.

Demberg, V. (2007). Phonological constraints and morphological preprocessing for grapheme-to-phoneme conversion. In *Proceedings of ACL-2007*.

Divay, M. and Vitale, A. J. (1997). Algorithms for grapheme-phoneme translation for English and French: Applications. In *Computational Linguistics*, 23(4), pp. 495–524.

Dutoit, T., Pagel, V., Pierret, N., Bataille, F. and van der Vrecken, O. (1996). The MBROLA Project: Towards a set of high-quality speech synthesizers free of use for non-commercial purposes. In ICSLP'96, pp. 1393–1396.

Huang, X., Acero, A., and Hon, H. W. (2001). *Spoken Language Processing*. Upper Saddle River, NJ: Prentice-Hall.

Jiampojamarn, S., Cherry, C. and Kondrak, G. (2008). Joint processing and discriminative training for letter-to-phoneme conversion. In *Proceedings of ACL-2008: Human Language Technology Conference*, pp. 905–913, Columbus, Ohio.

Laurent, A., Deleglise, P. and Meignier, S. (2009). Grapheme to phoneme conversion using an SMT system. In Proceedings of the *10th Annual Conference of the International Speech Communication Association*.

Li, M., Zhang, Y., Zhu, M. and Zhou, M. (2006). Exploring distributional similarity based models for query spelling correction. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pp. 1025–1032.

Och, F. J. and Ney, H. (2003). A Systematic Comparison of Various Statistical Alignment Models. In *Computational Linguistics*, 29(1), pp. 19–51.

Pagel, V., Lenzo, K. and Black, A. (1998). Letter to sound rules for accented lexicon compression. In *International Conference on Spoken Language Processing*, Sydney, Australia.

Rama, T., Singh, A. K. and Kolachina, S. (2009). Modeling Letter-to-Phoneme Conversion as a Phrase Based Statistical Machine Translation Problem with Minimum Error Rate Training. In *Proceedings of the 2009 Named Entities Workshop, ACL-IJCNLP 2009*, pp. 124–127, Suntec, Singapore.

Stan, A.,Yamagishi, J., King, S. and Aylett, M. (2011). The Romanian Speech Synthesis (RSS) corpus: building a high quality HMM-based speech synthesis system using a high sampling rate. In *Speech Communication*, 53 (3), pp. 442–450.

Taylor, P. (2005). Hidden Markov Models for grapheme to phoneme conversion. In *Proceedings of the 9th European Conference on Speech Communication and Technology*.

Vatanen, T., Jaakko Väyrynen, J. and Virpioja, S. (2010). Language Identification of Short Text Segments with N-gram Models. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation LREC'10*.

Zen, H., Nose, T., Yamagishi, J., Sako, S., and Tokuda, K. (2007). The HMM-based speech synthesis system (HTS) version 2.0. In *Proceedings of the 6th ISCA Workshop on Speech Synthesis*, pp. 294–299.