Project Specification: Annotated English

KJ Tsanaktsidis

FIT3036, Monash University

Abstract. 'Annotạtėd Ėnglish' is a proposal for clarifying the pronunciation of English words by defining unambiguous rules and adding diacritics representing exceptions to these rules. This is useful for people who can read English, but have difficulty with pronunciation. This proposal describes a tool for deducing the phonetic IPA transcription of an Annotated English word. The functionality of the tool is defined, and a high-level overview of the proposed software architecture is given. A project plan and a test plan outline how the project is to be delivered. It is hoped that the development of this tool will help with the adoption of Annotated English and prove useful to English readers looking to correct their pronunciation.

Word count: 1868 words

Table of Contents

1 Introduction

The English language has a truly horrid system of spelling compared with almost every other European language. Often, the way a word is written down has very little bearing on how a word is actually pronounced. The situation is especially bad for vowels [1]. This makes it difficult for people who may read English proficiently to actually speak it well.

Various proposals have been developed to reform the spelling of the English language to make words more closely indicate their pronunciation (for example, Interspell [2]). However, schemes such as this have a fundamental problem in that they require extensive retraining of generations of both native and non-native English speakers to stick.

Annotated English [3] is a system developed by José Hernández-Orallo to indicate the pronunciation of English words without changing their spelling. This is achieved by defining a set of unambiguous rules governing the pronunciation of English words. A set of diacritic symbols is also defined which can change the sound that letters would otherwise make. For example, in this scheme, the words 'Annotated English' would be written as 'Annot̨at̨ėd Ėnglish'. Appendix B of [3] describes a set of LaTeX commands which can render the required annotations. Using these commands, the text '`Annot\st{a}t\iot{e}d \iot{E}nglish`' produces the correct annotated output.

The advantage of this system is that the spelling of the text is not touched. A proficient English speaker can completely ignore all of the annotations and read normally. However, if a reader requires clarification on how a word should be pronounced, they can consult the annotations. Software which processes English text can also filter out the annotations and work as normal.

The pronunciation is deduced by working through 10 stages of processing outlined in [3], which gradually transform annotated text into IPA phonemes. IPA is a representation of a word's pronunciation using special symbols where each such symbol represents a single sound. Although several similar representations are possible for the same word [4], a given IPA string to a very good approximation has only one possible pronunciation. For example, the IPA representation of 'Annot̨at̨ėd Ėnglish' is ''ænəˌteɪd ˈɪŋglɪʃ'.

In order for Annotated English to be widely adopted, two pieces of software must exist. First, an encoder program needs to be available which takes as input a text and a pronunciation corpus (perhaps as described in [4]) and produces as output an annotated text. Secondly, a piece of software must exist which reproduces the IPA transcription of an annotated text, so as to assist a reader if required. In this project, we focus on building the latter.

2 Project Plan

2.1 Requirements

In this project, we wish to build a program which takes an annotated text as input and from this deduces the correct pronunciation of each word. The input will be read in the LaTeX source form outlined in the introduction and in [3]. This form will need to be parsed and then the sequence of rules outlined in [3] applied.

The program should then output the IPA representation of the words in the text. This should then be converted into X-SAMPA, a transcription of IPA using only 7-bit ASCII characters [5]. From X-SAMPA, it can be fed into a text-to-speech program to verify that the word sounds as it should.

2.2 Target Platform

Linux is the target platform for this project. No desktop environment is being targeted; instead, a command-line application will be built. Linux is appropriate for this project because of the first-class availability of an excellent finite-state transducer toolkit FOMA [6] as well as an X-SAMPA capable text-to-speech engine Pico [7]. Furthermore, a text-mode application is much simpler to build, and the typical Linux command-line environment is significantly richer in functionality than the Windows equivalent.

FOMA is a toolkit for producing text-processing applications. The programmer must produce a file containing replacement rules using regular expressions. At build time, the FOMA program compiles these rules into a representation of a finite-state transducer. This is an automata that uniquely specifies output text

for every valid piece of input text in an alphabet [8]. This allows the construction of an Annotated English interpreter by specifying and composing the individual rules as regular expressions.

The application itself will be coded in C. This choice has been made due to the availability of C-language interfaces to both FOMA and Pico, and the relative simplicity

There are no specific hardware requirements for this application, aside from that it must be able to run Linux. All testing will be performed on an x86 Intel 'Sandy Bridge' machine.

2.3 Risk Analysis

One of the greatest risks to this project is that the initial technology choices made may be unsuitable. Should this come to pass in the midst of the implementation phase, it is likely that significant amounts of code will need to be rewritten to suit the new language or libraries chosen. This will obviously consume precious time in research and coding.

Discovering unexpected complexity in the Annotated English implementation is also a significant risk to this project. If part way through the implementation of the ruleset a serious problem is encountered, it may be necessary to rework significant parts of the application to change the approach taken. Again, this will likely result in lost time and a failure to meet the deadline.

Significant work has already been undertaken to mitigate these risks through so-called Spike Testing [9]. This involves producing throw-away code to verify that the concepts and technologies have been understood and work as intended. In this case, a number of the Annotated English rules have already been implemented with FOMA and tested against some of the examples in [3]. As a result of this work, we have a much better understanding of how to use FOMA, its features, and its limitations. Additionally, several problems with a naive implementation of Annotated English have been identified and solutions developed. Some of these discoveries have informed the high-level design of the application, outlined in the next section.

An additional test has also been performed on the capabilities of the Pico TTS engine. How exactly to make Pico speak out X-SAMPA phonemes has now been identified and verified, and the quality of the output sound has been assessed. This testing has directly informed the choice of Pico as the TTS platform being targeted.

Additional risk mitigation has been undertaken by emailing Dr. Hernández-Orallo to open a line of communication with him. Already, this has helped resolve one ambiguity and provide some useful advice on how to proceed. This channel might be used to gain assistance during the project should more seemingly intractable issues arise.

2.4 Project Schedule

The application must be ready for demonstration by Tuesday the 20th of April [10]. This gives about 6 weeks starting from the 8th of April. In this time, all development and testing of the project must be conducted.

Dr Hernández-Orallo has indicated [11] that he expects the Annotated English rules in [3] to require some tweaking due to unintended conflicts. This suggests that an iterative development approach should be undertaken. A first version of the software will be produced as quickly as possible, preferably within the first three weeks. This will allow ample time for a testing loop, the details of which will be outlined later in the Test Plan section. Rules with unintended side effects can be identified by the output of incorrect pronunciation from the tool; they can then be appropriately narrowed. The cycle of testing then continues until no obvious problems are found. This is illustrated in figure 1.

The most important aspect of the project is the Annotated English → IPA converter. This aspect will be prioritised in development, so that should it not be possible to implement other features such as TTS in the time allowed, a functional tool is still developed.

3 Software Design

3.1 Software Interface

As previously noted, the application to be built will have a command-line interface. Annotated English in LaTeXform will be read from standard input. Since all Linux shells will allow the contents of a file to be piped

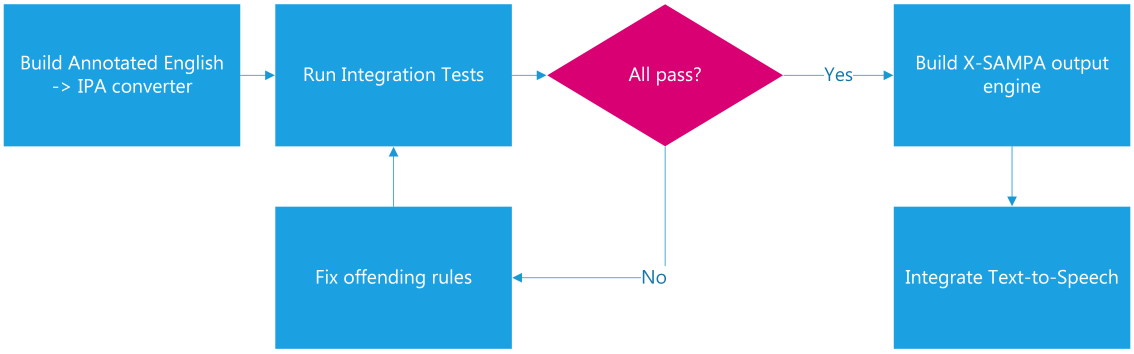

Fig. 1. Flowchart of the proposed development and testing methodology

to a program's standard input through input redirection, reading from a file does not need to be explicitly coded.

Whether IPA or X-SAMPA output is produced will depend on the value of a command-line argument to the program. This output will be written to standard output, where it can be written to a file via output redirection. An additional command-line switch will determine whether or not the TTS engine is invoked to pronounce this output.

3.2 Processing Pipeline

The software to be developed in this project is best modeled as a pipeline with several stages. Annotated English input is transformed to IPA by the sequential application of 10 rulesets in [3] by a transducer, then to X-SAMPA by a 1:1 transliteration process, and finally to speech sounds by the Pico TTS library. However, early testing has uncovered the need for a pre-processing stage before the finite-state transducer.

In some circumstances, Annotated English allows the nesting of LaTeX commands to place multiple annotations on a letter- for example, an accented letter can have a stress as in ánthólogy (`\pln{a}nth\st{\pln{o}}logy`). In a regular expression engine such as FOMA, it is impossible to detect whether or not something is in such a set of nested {} brackets. Matching brackets is the classic non-regular, context-free grammar [8].

A pre-processing step will be included to deal with such cases of nested annotations. This will principally consist of a pushdown automaton, a computational model capable of dealing with context-free grammars [12]. The `\group` command is purely a visual aid and can be removed altogether, whilst nested stresses can be replaced with special tokens.

The various steps in the pipeline will be wired together with the C-language API's of FOMA and Pico. The entire pipeline is illustrated visually in figure 2.

3.3 Performance

Because of the prototype nature of this software, performance is not an especially relevant issue for this project. That said, it is expected that the running time of the software will be linear in the size of the input text. This is because the language accepted by FOMA transducers is strictly regular, and such regular languages can be parsed in `O(N)` time [12].

4 Test Plan

As previously mentioned, it is anticipated that the rules in [3] will need to be modified somewhat in response to their unintended side-effects. It is therefore more likely that the rules themselves will need to change than that they are implemented incorrectly. Under these circumstances, unit testing is of limited usefulness.

Instead, this project will feature a focus on end-to-end integration tests. The Annotated English publication contains a large number of example annotated words. A cross-section of these will be chosen and their IPA pronunciations looked up in a corpus such as that from the Oxford English Dictionary [13]. These

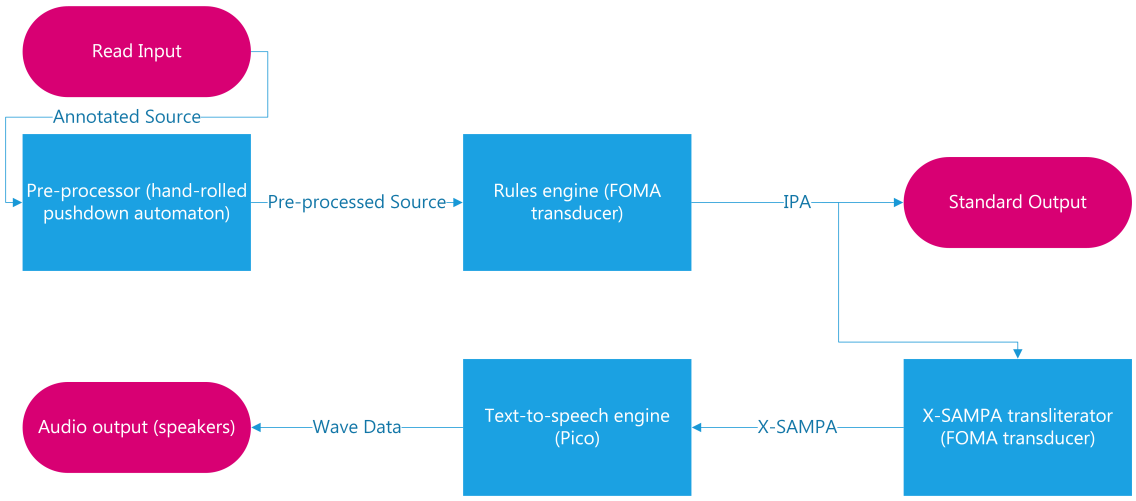

Fig. 2. Planned data flow in the software. The pre-processing step is implemented with a pushdown automaton, whilst the other text processing steps are implemented as FOMA transducers

values can then be compared in an automated manner with the output from the program, and inconsistencies debugged.

Once these tests are passing, a manual, auditory test of the TTS output of the software will be conducted. This will verify that the annotated words sound as they should.

5 Conclusion

In this document, a program for deducing Annotated English [3] pronunciations has been described. The high-level design for the system has been outlined, as well as a test plan and a project plan. The risk mitigation steps already taken put the project on the right track for success.

References

[1] Marsha Bell. *Spelling it Out: the Problems and Costs of English Spelling*. 2012.

[2] Valerie Yule. "The Design of Spelling to Match Needs and Abilities". In: *Harvard Educational Review* 56 (3 1986), pp. 278–298.

[3] José Hernández-Orallo. "Annotated English". In: *CoRR* abs/1012.5962 (2010).

[4] A. Ghoshal et al. "WEB-derived pronunciations". In: *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*. 2009, pp. 4289–4292. DOI: 10.1109/ICASSP.2009.4960577.

[5] John C. Wells. "SAMPA computer readable phonetic alphabet". In: *Handbook of Standards and Resources for Spoken Language Systems*. Ed. by D. Gibbon, R. Moore, and R. Winski. Berlin: Mouton de Gruyter, 1997.

[6] Mans Hulden. *FOMA: Finite-State Compiler and C Library*. https://code.google.com/p/foma/. Accessed April 1 2014. 2012.

[7] SVOX AG. *SVOX Pico: Speech Output Engine SDK*. http://dafpolo.free.fr/telecharger/svoxpico/SVOX_Pico_Manual.pdf. Accessed April 1 2014. 2009.

[8] E. Roche and Y. Schabes. *Finite-State Language Processing*. Massachusetts: MIT Press, 1997.

[9] James Shore and Shane Warden. *The Art of Agile Development*. Sebastopol CA: O'Reilly Media, 2007.

[10] Monash University. *FIT3036 Unit Guide*. http://www.infotech.monash.edu.au/units/archive/2014/s1/fit3036.html. Accessed April 3 2014. 2014.

[11] José Hernández-Orallo. Private Communication. 2014.

[12] Michael Sipser. *Introduction to the Theory of Computation*. 1st. International Thomson Publishing, 1996.
[13] Oxford University Prress. *Oxford English Dictionary*. `http://www.oxforddictionaries.com/`.

Word count: 1868 words