Dutch Automatic Speech Recognition on the Web: Towards a General Purpose System

Joris Pelemans, Kris Demuynck, Patrick Wambacq

Department ESAT, KU Leuven, Leuven, Belgium

{joris.pelemans,kris.demuynck,patrick.wambacq}@esat.kuleuven.be

Abstract

In this paper we present our state-of-the-art automatic speech recognition system for Dutch that we made available on the web. The free, online disclosure of our software aims at allowing non-specialists to adopt ASR technology effortlessly. Access is possible via a standard web browser or as a web service in automated tools. We discuss the way the web application was built and focus on usability criteria – especially interoperability. To overcome user differences we provide input conversion and basic parameter selection. Extensions of the current system and a path to robust, general purpose ASR are suggested.

Index Terms: robust speech recognition, web service, automatic model selection, speech processing tools

1. Introduction

Despite several decades of research Automatic Speech Recognition (ASR) is still no match for Human Speech Recognition (HSR) and it is likely that this won't change in the near future. Nevertheless, ASR has already proven its value in a lot of applications: children are using automatic tutors to improve their reading skills; doctors are gaining time and money using dictation software; disabled people are able to control the computer with their voice instead of the keyboard, ...

Also in the world of Human and Social Sciences (HSS) there exists a great need to search and access spoken data in a more efficient way. A lot of material is stored as audio only: recorded interviews, speeches, news broadcasts, ... Easy access to this material by historians, political scientists or linguists requires a complete or partial transcription. This can then be used to find certain keywords or speakers, or can be analyzed using standard text mining applications. Some of the material has already been transcribed manually but is not aligned with the speech which makes it cumbersome to process. In all these cases ASR technology can help, either by aligning the incomplete transcriptions with the audio or by providing automatic transcriptions which can be adopted as is or can be utilized as a baseline transcription to speed up the manual annotation process.

Whereas the current state-of-the-art technology works well for restricted tasks, so far it does not provide an adequate user experience for unrestricted tasks. To assure good performance, current recognition systems still require careful selection of components (signal preprocessing, acoustic models, language models and lexicon) and meticulous tuning of all parameters based on the acoustic and linguistic conditions of the audio. These choices are far from trivial and in most cases the interpretation of the different settings requires expert knowledge. In addition, the end user has to be able to install the software, keep

it up-to-date and provide a computing infrastructure that is powerful enough to run the software. All of this is a burden on the end user who does not want to be concerned with such technical details and would prefer to simply input an audio file and receive a transcription.

The CLARIN [1] initiative is a large-scale pan-European collaborative effort to create, coordinate and make language resources and technology available and readily usable for the whole European HSS community. CLARIN offers scholars the tools to allow computer-aided language processing, addressing the multiple roles language plays in the HSS. To this aim they created a list of usability criteria which all of the tools have to meet. The CLARIN project TTNWW [2] has the goal to develop and provide Speech & Language Technology (SLT) tools for Dutch to HSS researchers with little or no technical background in the field of ASR. These tools are meant for them to answer their current research questions better and provide the possibility to formulate new types of research questions.

Our research aims to put a first step towards achieving the TTNWW goals by presenting our state-of-the-art ASR system SPRAAK [3, 4] to the user in an environment that meets the CLARIN usability criteria. The software is embedded into a website which we developed using the Computational Linguistics Application Mediator (CLAM) [5]. It can be accessed with a web browser on <http://www.spraak.org/webservice/> or as a web service in automated tools and as such guarantees fast and easy access. In this initial system, the need for expert knowledge is reduced by providing input conversion as a separate web service and selecting system parameters based on user input.

This paper is organized as follows. Section 2 describes our state-of-the-art speech recognizer in more detail. In Section 3 we discuss necessary criteria for application usability according to CLARIN. Section 4 then explains how we choose to meet these criteria, focusing on the web service, input/output conversion and a general purpose ASR system. We end with some conclusive remarks and an overview of future work in Section 5.

2. Speech recognizer

The first installation of the web service is built around and has the same performance as the state-of-the-art recognizer that was made for the N-Best project [6] and follows the block diagram of Figure 1. This speaker independent large vocabulary continuous speech recognizer was built using the SPRAAK [3, 4] toolkit and has the capability to select components and adjust parameter settings on the fly, based on the observed conditions in the audio. In the current installment, two conditions are distinguished: studio quality broadband speech containing mainly prepared speech and telephone interviews which are expected to contain more spontaneous speech.

This research was supported by the CLARIN project TTNWW.

Figure 1: Schematic diagram of SPRAAK

For the acoustic models (AMs), 49 three-state acoustic units (46 phones, silence, garbage and speaker noise) and one single-state phone (short schwa) are modeled using our default tied gaussian approach, i.e. the density function for each of the 4k cross-word context-dependent tied states is modeled as a mixture of an arbitrary subset of gaussians drawn from a global pool of 50k gaussians. The mixtures use on average 180 gaussians to model a 36 dimensional observation vector of features which are obtained by means of a mutual information based discriminant linear transform (MIDA) on vocal-tract length normalized (VTLN) and mean-normalized MEL-scale spectral features and their first and second order time derivatives [4].

Using a lexicon of 400k words, 5-gram word language models (LMs) with modified Kneser-Ney discounting are trained on 4 main text components: 12 Southern Dutch newspapers, 10 Northern Dutch newspapers and transcriptions of broadcast news and conversational telephone speech. The 4 LMs are interpolated linearly and perplexity minimization is done to find the optimal interpolation weights. Lexicon creation is handled by an updated version of the system described in [7]. Dutch has a decent amount of (regional) pronunciation variation which is addressed by using phonological rules to generate the likely pronunciation variants. This results in a median of 3.8 pronunciations per word or 1.13 variants per phone in the canonical word transcriptions.

Since Dutch compounds are always written as a single word, the word recognition results are post-processed for compounding. Two subsequent words are replaced by their compound if the following criteria are met: 1) the words are longer than 3 letters, 2) the words are not very rare, 3) the unigram count of the compound is higher than the bigram count of the individual words. This approach essentially extends the 400k lexicon to a 6M lexicon.

The main parameters of the system concern hypothesis pruning and combining language model and acoustic model. To combine the model scores, we employ our standard way of handling this problem [4], by having a LM scaling factor and a word startup cost. Beam search pruning is applied to control the amount of hypotheses in the search space [8]: a threshold indicates how much the score of a hypothesis can drop below the score of the most likely hypothesis; if most hypotheses have a similar score, a beam width parameter is applied to indicate how many hypotheses can be retained, keeping only the best ones.

3. CLARIN usability criteria

CLARIN [1] is committed to establish an integrated and interoperable research infrastructure of language resources and its technology. It aims at lifting the current fragmentation, offering a stable, persistent, accessible and extendable infrastructure and therefore enabling eHumanities.

Interoperable: The resources and services overcome format, structure and terminological differences, thus enabling everyone – even those without expert domain knowledge – to benefit from the applications. The user does not have to care about input/output formats or character encodings. The need for expert knowledge is limited and where possible converted into non-expert formulations.

Stable: The resources and services are offered with a high availability. The amount of time and effort between the user wanting to run the application and actually running it, is reduced as much as possible. The application itself puts minimal constraints on the user's computer. An application that demands a large amount of memory and a powerful processor severely limits the group of possible users. In an optimal scenario, the application is always available and does not require local installation.

Persistent: The resources and services are planned to be accessible for many years so that researchers can rely on them.

Accessible: The resources and services are accessible everywhere. Different access methods are offered, tailored to the needs of the communities making use of them.

Extendable: The infrastructure is open so that new resources and services can be added effortlessly.

4. Solutions

4.1. The web as accessible platform

For an accessible, easy-to-use application the web is a logical choice of platform. It's always available, can be accessed anywhere, does not require local installation and puts minimal constraints on the user's computer. The application might need a large amount of memory or a powerful processor, all of which are taken care of by the server on which the web application is run. This web server can - if necessary - distribute its tasks among other servers.

Building a stand-alone web application however is not all that interesting: integration into automated tools is not possible and combination with another application always requires the construction of a new application. Web services with a common mediator on top overcome these issues, making software integration straightforward. Moreover, the output of one service can automatically be used as the input to another and every service can be integrated into a workflow system where the user chooses which services should be connected. A single web portal can then be used to provide many web services, making it a convenient access point.

4.1.1. Web service

Instead of building a complete web service from scratch, we set it up with CLAM [5]. CLAM was developed to quickly and transparently transform Natural Language Processing (NLP)

applications into a web service, with which automated clients can communicate, but which at the same time also acts as a modern web application with which end users can interact. It takes a description of the system and wraps itself around it, allowing clients or users to upload input to the application, start it with specific parameters of their choice, and download and view the output of the application. While the application runs, users can monitor its status.

CLAM is entirely written in Python. It is set up in a modular fashion and is therefore easily extendable. It offers a rich API for writing clients and wrapper scripts.

The provided service is a RESTful [9] web service, meaning that it uses the HTTP verbs GET, POST, PUT and DELETE to manipulate resources and returns responses using the HTTP response codes. The principal resource in CLAM is called a project. Various users can maintain various projects, each representing one specific run of the system, with particular input data, output data, and a set of configured parameters.

4.1.2. CLAM Architecture

CLAM has a layered architecture, with at the core the application that is turned into a web service. The application itself can remain untouched and unaware of CLAM. The block diagram in Figure 2, reproduced from [5] with permission, illustrates the various layers:

- The service configuration specifies what parameters the system can take, and what input and output formats are expected under what circumstances.
- The system wrapper script acts as the glue between CLAM and the application. CLAM executes the wrapper script, and the wrapper script in turn invokes the actual application. Using a wrapper script offers more flexibility than letting CLAM directly invoke the application and allows it to be totally independent of CLAM.
- The key function of the CLAM Data API is parsing the CLAM XML Data file that the CLAM web service uses to communicate with clients. This data is parsed and all its components are made available in an instance of a CLAMData class.

All communication with either an end user or an automated client is done over HTTP which allows to put several web services in a workflow where one web service's output can serve as input for a second web service, thus creating a chain of services. Standard HTTP authentication protocols can be implemented to ensure the security of each user and his project. For more technical information about CLAM, we refer the reader to [5].

4.2. Input/output conversion

Using the CLAM web service as an interface between user and speech recognizer immediately meets many of the stated CLARIN criteria: the application is stable, persistent, accessible and extendable. The interoperability criterion however has not been fully met: a speech recognizer on the web can still be difficult to use if it requires expert knowledge about input and output formats and/or system models and parameters.

Audio recordings can be digitized in many ways. A large number of audio file formats exist and for each one there are a plethora of ways in which the audio samples can be encoded: sampling rate, bits per sample, number of channels and encoding algorithm. Rather than providing direct support for all of these, we set up another web service which is dedicated to the

Figure 2: CLAM architecture

conversion of multimedia files to our desired input format. The service uses existing conversion tools to convert the most common audio and video formats to WAV linear PCM 16kHz 16bit mono.

The same can be done for output formats. At the moment we still limit the output to 3 common transcription formats: NIST ctm, SubStation Alpha subtitles and plain text. In the future we will build another web service that does conversion from and to the most common output formats.

4.3. Towards general purpose ASR

ASR output is very dependent on acoustic and linguistic conditions such as environment, channel, dialect, word usage, ... It is therefore difficult to build a general purpose, condition independent speech recognizer that performs well on all user data. In this section we discuss the influence of these conditions on the system models and parameters.

4.3.1. Model selection

Current state-of-the-art ASR technology does not yet cope well with reverberation, noise, out-of-vocabulary (OOV) words, spontaneous and accented speech, deviant word usage, ... For these challenging conditions, specialized models and often even systems are needed, some of which are developed in our group and can be readily integrated into our general purpose system: [10] gives an overview of different dereverberation algorithms, [11] uses the missing data technique for noise robustness, [12] converts phonemes to graphemes for OOV word transcription and in [13] a layered approach is adopted for a.o. explicit handling of large pronunciation variation, typical to spontaneous and accented speech. Other recent research shows promising results for Dutch LM adaptation using automatic socio-situational setting classification [14] and reconstruction of very noisy data by online GMM adaptation [15]. After integration, our general purpose system will either base its selection decision on user input or automatically detect the conditions in the signal, after which it will utilize the selected model or system.

Rather than waiting for the complete integration of the different models and systems, we want to make our initial setup available as soon as possible. This allows the user to start experimenting with our system and as such reward us with valuable feedback and audio input for optimization. In the current setup only two conditions are distinguished, which trigger different models: studio quality broadband speech containing mainly prepared speech and telephone interviews which are expected to contain more spontaneous speech.

4.3.2. Parameter selection

Even after selecting the optimal models for the task at hand, the system performance is still very dependent on the different parameters. In our system this is observed most clearly in pruning: more difficult tasks require more hypotheses, hence less pruning. It is not obvious however how much pruning should be done to achieve optimal performance. A similar observation can be made when looking at model score combinations: since a more noisy audio signal yields less reliable acoustic hypotheses, it requires a higher relative scaling of the language model scores. The optimal magnitude of the scaling is again unclear.

As a very first, suboptimal step towards condition independence, we confront the user with basic questions concerning the difficulty of his task:

1. Does the audio contain background noise?
2. Is the audio recorded with suboptimal equipment?
3. Does the audio contain spontaneous or accented speech?
4. Does the audio exhibit deviant word usage?

The questions are then mapped to the parameters as follows: if none of them is answered positively, we consider the task to be undemanding and adopt the pruning settings yielding a real-time (1xRT) performance in the N-Best evaluation benchmark [6]. If 1 or 2 of the questions are answered positively, we choose the medium pruning settings of the 3xRT performance. And finally, if 3 or 4 of the questions are answered positively, we allow many hypotheses and prune only the few least likely ones, identical to the 9xRT performance. All of the options use the N-Best optimized model score combination parameters, except for the case of noisy audio input where the language model scores are scaled marginally.

This approach is obviously naive and will likely use suboptimal parameters for many given tasks. At the time of writing we are testing our initial setup on several corpora to optimize the system parameters for different conditions. If we can show that similar conditions yield similar optimal parameters, we can ask more specific questions and use the obtained parameter settings to adjust the current ones. In later work we can then attempt to measure the conditions directly and more reliably from the signal, so the need for user input decreases even further.

5. Conclusion

We incorporated an initial version of our state-of-the-art speech recognizer into a web service that meets the CLARIN usability criteria. It is (1) interoperable: the most common audio and video formats can be used as input by a dedicated conversion web service; (2) stable: the web service is available around the clock and puts minimal constraints on the user's computer because processing is taken care of by the web server(s); (3) persistent: user projects are stored and shielded from other users by HTTP authentication; (4) accessible: the service can be accessed with both a web browser and automated tools; and (5)

extendable: experience has proven that adding services is effortless. Several services have been built and we plan to build more in the future e.g. output conversion, alignment, keyword spotting, ... These services will then be integrated into a workflow system, thus creating a convenient access point for users.

As a first extension to interoperability we developed a system that confronts the user with questions about the conditions of the audio and uses his answers to select system parameters.

Future work will focus on more reliable, automatic detection of the different problematic conditions directly from the signal. For each homogeneous block the system will then select the optimal models and parameters. Extra user input e.g. representative text or transcribed speech will be allowed in order to perform model adaptation.

To our knowledge <http://www.spraak.org/webservice/> presents the first free Dutch ASR application on the web, ready to be used by anyone.

6. References

- [1] CLARIN: Common Language Resources and Technology Infrastructure, <http://www.clarin.eu>.
- [2] TTNWW: Language and Speech tools for Dutch as Webservice in a Workflow, <http://www.esat.kuleuven.be/psi/spraak/projects/index.php?proj=TTNWW>.
- [3] K. Demuyne, J. Roelens, D. Van Compernelle and P. Wambacq, "SPRAAK: An Open Source SPeech Recognition and Automatic Annotation Kit", in Proc. ICSLP, 2008, pp. 495–498.
- [4] K. Demuyne, "Extracting, Modelling and Combining Information in Speech Recognition", Ph.D. thesis, K.U.Leuven ESAT, 2001.
- [5] M. van Gompel, "CLAM: Computational Linguistics Application Mediator. Documentation. ILK Technical Report 12-02", <http://ilk.uvt.nl/downloads/pub/papers/ilk.1202.pdf>, 2012.
- [6] K. Demuyne, A. Puurula, D. Van Compernelle and P. Wambacq, "The ESAT 2008 system for N-Best Dutch speech recognition benchmark", in Proc. ASRU, 2009, pp. 339–343.
- [7] K. Demuyne, T. Laureys and S. Gillis, "Automatic generation of phonetic transcriptions for large speech corpora", in Proc. ICSLP, 2002, vol. I, pp. 333–336.
- [8] V. Steinbiss, B.-H. Tran and H. Ney, "Improvements in beam search", in Proc. ICSLP, 1994, pp. 2143–2146.
- [9] R. Fielding, "Architectural Styles and the Design of Network-based Software Architectures", Ph.D. thesis, University of California, Irvine, 2000.
- [10] K. Eneman, J. Duchateau, M. Moonen, D. Van Compernelle and H. Van hamme, "Assessment of dereverberation algorithms for large vocabulary speech recognition systems", in Proc. ECSC, 2003, pp. 2689–2692.
- [11] M. Van Segbroeck and H. Van hamme, "Advances in missing feature techniques for robust large vocabulary continuous speech recognition", IEEE Transactions on Audio, Speech and Language Processing, 2011, vol. 19, pp. 123–137.
- [12] B. Decadt, J. Duchateau, W. Daelemans and P. Wambacq, "Transcription of out-of-vocabulary words in large vocabulary speech recognition based on phoneme-to-grapheme conversion", in Proc. ICASSP, 2002, vol. 1, pp. 861–864.
- [13] J. Pelemans, K. Demuyne and P. Wambacq, "A layered approach for Dutch large vocabulary continuous speech recognition", in Proc. ICASSP, 2012, pp. 4421–4424.
- [14] Y. Shi, P. Wiggers and C. Jonker, "Dynamic Bayesian socio-situational setting classification", in Proc. ICASSP, 2012, pp. 5081–5084.
- [15] K. Woolf and J. H. L. Hansen, "Feature compensation employing online GMM adaptation for speech recognition in unknown severely adverse environments", in Proc. ICASSP, 2012, pp. 4121–4124.