Bachelor thesis - Better accuracy of automatic lecture transcriptions by using context information from slide contents

Jonathan Werner

Contents

1 Introduction

Scannability is crucial for academic research: you have to be able to quickly evaluate the usefulness of a given resource by skimming the content and looking for the parts that are specifically relevant to the task at hand.

The medium in which those resources are available is very centered on textual representation. Spoken content, hereinafter called *speech media* (audio- or audiovisual media that mainly consist of spoken language) doesn't make it possible to scan its contents. You are "stabbing in the dark" when looking for something specific in a medium like this and have to consume it like a linear narrative.

This means that although lectures and conference talks are a central element to science they are much more challenging and tedious to use for research work.

Being able to a) efficiently search and b) look at the temporal distribution of important keywords in a visually dense way would increase the usefulness of speech media in the scientific context immensely.

One approach to accomplish these goals is to utilize Automatic Speech Recognition (ASR) in order to transcribe speech to text and also get timing information for the recognized words. This makes it possible to derive information about the density of given words at a given point of time in the talk, which in turn allows to compute word occurence density maxima. This opens up possibilities for compact visual representation of the interesting keywords, thus allowing the user to scan.

The main challenge when using ASR for this task is the recognition accuracy of technical terms. Most of them are not included in the language models that are available as these are broad and generic so as to optimize accuracy over a wide topic spectrum. But when they are not included in the language model they have a very small chance to be correctly recognized at all.

So the usefulness of applying ASR with a generic language model to the problem is very small, as the intersection of interesting keywords with those technical terms that can not be recognized is very big.

The central goal of this thesis is to explore an approach to overcome this problem. This approach consists of using words from lecture slides or other notes to generate a lecture-specific language model. This is then interpolated with a generic language model. Finally the results are compared with the 'baseline' accuracy of the generic model.

1.1 Research questions

The research questions I want to investigate in this thesis can be formulated as follows:

(1) When we apply ASR to university lectures, what is the advantage of using an approach that consists of creating a lecture-specific language model and interpolating it with a generic language model, given that we are interested in improving the recognition accuracy of *interesting keywords* for the sake of searchability and scannability?

(2) What metric is useful for quantifying this advantage?

A secondary question is: How can we *use* the results from our approach to provide graphical interfaces for improving the user's ability to search and scan the given speech medium? The exploration of this question will not be at the center of this thesis, but it will provide practical motivation for the results of our approach.

1.2 Structure

The structure of this thesis is as follows: I will start by giving an overview over the scientific work done in ASR, explaining fundamental speech recognition concepts and discussing the most prevalent approaches. I will then present the chosen test data, which consists of lectures from the openly available *Open Yale Courses*[1], and explain selection criteria. This will be followed by a description of the LM-Interpolation approach, explaining general concepts and implementation. I will then discuss suitable metrics for evaluation and analyze the results. I will finally explore an interface prototype for dense visual representation of keyword distributions over the timeline of a lecture. I will close by recapitulating the thesis results and summarizing possible extension points.

[1] Website: http://oyc.yale.edu/

2 Scientific background

2.1 The field of Automatic Speech Recognition

Automatic Speech Recognition (ASR) can be defined as an "independent, machine-based process of decoding and transcribing oral speech", where a "typical ASR system receives acoustic input from a speaker through a microphone, analyzes it using some pattern, model, or algorithm, and produces an output, usually in the form of a text" (Lai, Karat, & Yankelovich, 2008).

Rabiner & Juang (1993) date the first research on ASR back to the early 1950s, when Bell Labs built a system for single-speaker digit recognition. Since then the field has seen three major approaches, which Marquard (2012) calls the *acoustic-phonetic approach*, the *statical pattern-recognition approach* and the *artificial intelligence approach*.

The acoustic-phonetic approach aimed to identify phonetic features of speech such as vowels or consonants directly through their acoustic properties and from that build up words based on these constituent elements.

The statistical pattern-recognition approach measures features of the acoustic signal and compares these to existing patterns from a range of reference sources to produce similarity scores by using a search process; patterns are taken from multiple sources such as acoustic and language models.

Artificial intelligence approaches are mainly differentiated by being integrative, combining multiple types of knowledge sources. While the former approach uses fixed input features, AI approaches establish them by *learning*. A key technology here is the use of deep neural networks and other deep learning approaches (Hinton et al., 2012), which has been an active area of research in the last decade.

The most prevalent approach today is the statistical pattern-recognition approach, as it produces results with much higher accuracy compared to the acoustic-phonetic approach. The use of Hidden Markov Models (HMM) has been playing a key role in this approach, as it allows recognizers to use a statistical model of a given pattern rather than a fixed representation. This is the paradigm used as the technical foundation in our approach, as well.

2.2 Dimensions of speech recognition

There are three dimensions which serve to classify different applications of speech recognition (Marquard, 2012):

(1) **Dependent vs. independent**. Dependent recognition systems are developed to be used by one speaker. They are easier to develop and more accurate, but not flexible. Independent systems in contrast are developed

to be used by *any* speaker of a particular language or dialect, i.e speakers of North American English (NAE). Independent systems have lower accuracy but better flexibility. **Adaptive** systems lie between these poles, they are able to adapt to a particular speaker through training.

(2) **Small vs. large vocabulary**. Small vocabularies contain only up to a few hundred words and might be modeled by an explicit grammar, whereas large vocabularies contain tens of thousands of words so as to be able to model general purpose spoken language over a variety of domains.

(3) **Continuous vs. isolated speech**. Isolated speech consists of single words that are spoken with pauses in between them, whereas continuous speech consists of words that are spoken in a connected way. Continuous speech is significantly more difficult to recognize, as it is a) more difficult to find the start and end of words and b) the pronunciation of words changes in relation to their surrounding words.

With these three dimensions we can for example classify the application areas command and control systems, dictation and lecture transcription (Marquard, 2012):

Table 1: Three application areas

Application	Speaker	Vocabulary	Duration
Dictation	Dependent	Large	Connected
Command and control system	Independent	Small	Isolated
Lecture transcription	Independent/Adaptive	Large	Connected

The task of automatic lecture transcription can thus be characterized as speaker-independent (SI) large-vocabulary continuous speech recognition (LVCSR).

2.3 Concepts

Speech recognition in the *statistical pattern-recognition approach* paradigm has these major concepts that are necessary for its understanding:

- phonemes
- phonetic dictionaries
- search
- acoustic models (AM)
- language models (LM)

2.3.1 Phonemes

A *phoneme* is "the smallest contrastive linguistic unit which may bring about a change of meaning" (Cruttenden, 2014, p. 43). Phonemes are the smallest unit of sound in speech which are combined to form words. The word *sun* for example can be represented by the phonemes `/s/`, `/u/` and `/n/`, the world *sum* differs only in the last phoneme, but the words have different meaning – hence `/m/` and `/n/` are different phonemes.

A language with a specific accent can be described by the set of phonemes that it consists of. Figure 1 shows the phonemes that are used in North American English, using symbols from the International Phonetic Alphabet (IPA).

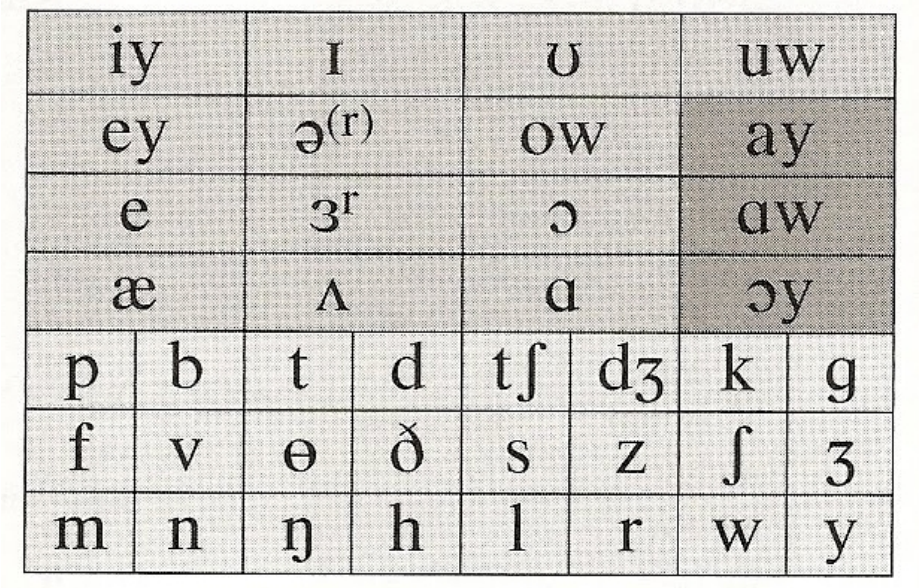

Figure 1: Phonemes in NAE

To be able to use phonemes in software an ASCII representation is more suitable. The standard for General American English is the *Arpabet*. Here each phoneme is mapped to one or two capital letters. The digits `0`, `1` and `2` signify stress markers: no stress, primary and secondary stress respectively.

2.3.2 Phonetic dictionaries

Phonetic dictionaries map words to one or more versions of phoneme sequences.

A phonetic representation of a word is specified manually based on the knowledge of how written words *actually sound* when spoken.

An excerpt from the CMU EN-US Pronouncing Dictionary (*cmudict-en-us.dict*, 2015) looks like this (phonemes are given in Arpabet representation):

```
...
abdollah AE B D AA L AH
abdomen AE B D OW M AH N
abdomen(2) AE B D AH M AH N
abdominal AE B D AA M AH N AH L
abdominal(2) AH B D AA M AH N AH L
...
```

The dictionary has 133.425 entries. Generally only words that are in the phonetic dictionary being used can be recognized during speech recognition. *Grapheme*[2]*-to-Phoneme converters* (G2P) however make it possible to get phoneme sequence hypotheses for arbitrary words (i.e arbitrary sequences of graphemes). While these results are on average less accurate than manually created variants, they play a vital role in texts with many technical terms as these are often not included in phonetic dictionaries.

2.3.3 Search

Search is the basic abstraction that lies at the center of the speech recognition process, which is called the *decoding phase*. The recognition process is implemented as a search algorithm on a directed search graph. This graph is constructed by taking different information dimensions into account: typically an acoustic model, a language model and phonetic dictionary. An example graph is shown in figure 2[3]. Here words (from a language model) are shown in rectangles, phonemes (from a phonetic dictionary) as dark circles and audio input features as white circles (from an acoustic model). The general search space topology and phonetic context size is dependent on the specific algorithm used.

2.3.4 Acoustic models

An acoustic model (AM) describes the relation between an audio signal and the probability that this signal represents a given phoneme.

Acoustic models are created by *training* them on a *corpus* of audio recordings and matching transcripts. When being used in the context of speaker-independent recognition, these models are trained with a variety of speakers that represent a broad spectrum of the language/accent that the acoustic model should represent.

[2] A grapheme is "the smallest unit used in describing the writing system of a language" (Florian, 1996). Some languages have strong correspondences between phonemes and graphemes, this is not a necessary relationship however – the English word "debt" for example has the grapheme `<b>`, which is not represented by a corresponding sound.

[3] The graph is taken from Walker et al. (2004).

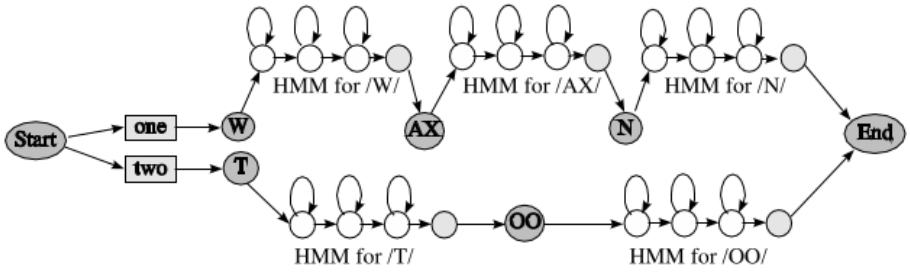

Figure 2: Search graph with example words "one" and "two"

During the decoding phase the acoustic model and a phonetic dictionary are used to match sequences of small audio "slices" to possible phonemes and those phonemes to possible word sequence hypotheses.

However, acoustic models alone are not sufficient for speech recognition as they do not have the "higher-level" linguistic information necessary to distinguish e.g. between homonyms and similar-sounding phrases such as "wreck a nice beach" and "recognize speech" (Marquard, 2012, p. 11). This information is provided by *language models*.

2.3.5 Language Models

Language models (LM) guide and constrain the search process that a speech recognition system performs by assigning probabilities to sequences of words. The basic premise, called Markov assumption, is that the overall probability of a sentence with the words $w_1, ..., w_n$ can be approximated as follows:

$$P(w_1, ..., w_n) = \prod_{i=1}^{m} P(w_i \mid w_1, ...w_{i-1})$$

This assumption says that an approximate probability of a given word can be calculated only by looking at the last n - 1 prior words. This property significantly decreases statistical complexity and thus makes it computationally feasible.

An example approximation with a bigram model for the sentence "I saw the red house" represented as P(I, saw, the, red, house) would look like

$$P(\text{I} \mid \langle s \rangle) \cdot P(\text{saw} \mid \text{I}) \cdot P(\text{the} \mid \text{saw}) \cdot P(\text{red} \mid \text{the}) \cdot P(\text{house} \mid \text{red}) \cdot P(\langle s \rangle \mid \text{house})$$

The most commonly used form of language models are *n-gram language models*. In the context of a language model an *n-gram* is a sequence of n words. 1-grams

are called *unigrams*, 2-grams are called *bigrams* and 3-grams are called *trigrams*. An *n-gram language model* maps a set of *n-grams* to probabilities that they occur in a given piece of text.

N-gram language models do not need to be constrained to one type of n-gram; the *Generic US English Language Model* (*cmusphinx-5.0-en-us.lm*, 2015) from CMUSphinx we will use as the baseline for our approach consists of 1-, 2, and 3-grams, for example.

Language models are trained by applying statistical methods on a text corpus. Analogous to acoustic models, generic language models use huge text corpora with a broad variety of topics. It is however possible to train language models on small and specialized text corpora, which is the central technical foundation for the approach discussed in this thesis.

2.4 Work done on ASR for lecture transcription

I will now give an overview over the scientific work done on lecture transcription, using Marquard (2012) as a guiding reference.

The research for speech recognition on lectures can be partitioned into three general approaches: generalization approaches, specialization approaches and approaches involving the user for manual correction and improvements.

2.4.1 Generalization approaches

Generalization approaches try to create models that capture common characteristics of lectures. Those characteristics include highly spontaneous presentation style and "strong coarticulation effects, non-grammatical constructions, hesitations, repetitions, and filled pauses" (Yamazaki, Iwano, Shinoda, Furui, & Yokota, 2007). Glass, Hazen, Hetherington, & Wang (2004) note the "colloquial nature" of lectures as well as the "poor planning at the sentence level [and] higher structural levels".

The generalization approach has been applied on the acoustic model level: Cettolo, Brugnara, & Federico (2004) have examined adapting a generic acoustic model to account for spontaneous speech phenomena ("filler sounds").

While a subfield of ASR called "speaker diarization" tries to account for the interactivity between lecturers and students by identifying multiple speakers, most research treats lectures as single speaker events with the audience as background noise.

Generalization approaches at the language model level try to model common linguistic traits of the lecture genre (this can be called the *macro level*). Kato, Nanjo, & Kawahara (2000) investigate topic-independent language modeling by

creating a large corpus of text from lecture transcripts and panel discussions and then removing topic-specific keywords.[4]

2.4.2 Specialization approaches

Specialization approaches try to use context specific to a single lecture (*meso level*) or parts of a single lecture (*micro level*[5]).

Methods used for creating LMs from context information can be categorized into two approaches: direct usage of lecture slides and notes for the creation of LMs versus usage of "derived" data from these materials. Deriving data by using keywords found in slides, using them as web search query terms and using the found documents as the basis for LM creation is explored in Munteanu, Penn, & Baecker (2007), Kawahara, Nemoto, & Akita (2008) and Marquard (2012).

Using the whole text from lecture slides has been explored by Yamazaki et al. (2007). They compare the meso level with the micro level by dynamically adapting the LM to the speech corresponding to a particular slide. Kawahara et al. (2008) also examine dynamic local slide-by-slide adaption and compare it to global topic adaption using Probabilistic Latent Semantic Analysis (PLSA)[6] and web text collection, concluding that the latter performs worse than the former because of a "worse orientation to topic words".

[4] In a second step they combine this generalization technique with a specialization technique by adapting the resulting LM with a lecture-specific language model by using preprint papers of a given lecture.

[5] The three levels are taken from Marquard (2012).

[6] Latent Semantic Analysis is an approach to document comparison and retrieval which relies on a numeric analysis of word frequency and proximity.

3 Test data

The test data I will use for evaluating our approach will be from *Open Yale Courses*[7], which is a selection of openly available lectures from Yale university. It consists of 42 courses from 25 departments. Each course has about 20-25 sessions that have an average length of 50 minutes. Each lecture is provided with good quality audio and video recordings, precise manual transcripts and lecture material when available. Only about 20% of the lectures have lecture notes or slides at all and most materials from the natural and formal science departments (physics, astronomics, mathematics) consist of hand-written notes, making them unsuitable for our approach. All talks are in English.

I have chosen the following lectures: (Department, Course, Lecture Number - Title, abbreviation)

- *Biomedical Engineering*: Frontiers of Biomedical Engineering, 1 - What is Biomedical Engineering? (`biomed-eng-1`)
- *Environmental Studies*: Environmental Politics and Law, 8 - Chemically Dependent Agriculture (`environmental-8`)
- *Geology & Geophysics*: The atmosphere, the ocean, and environmental change, 8 - Horizontal transport (`geology-8`)
- *Philosopy*: Philosophy and the science of human nature, 8 - Flourishing and Detachment (`human-nature-8`)
- *Psychology*: Introduction to Psychology, 14 - What Motivates Us: Sex (`psy-14`)
- *Psychology*: Introduction to Psychology, 5 - What Is It Like to Be a Baby: The Development of Thought (`psy-5`)

The main selection criterion here was topical diversity, the challenge being that the majority of talks with computer-parsable notes was from the humanities.

3.1 Materials overview

The available material is very heterogeneous. I will now give an overview with excerpts which will serve as a basis for examining at a later point if the quality and quantity of the supplied material is correlated with the amount of improvement of our approach.

`geology-8` supplies a 2-page excercise sheet.

[7] http://oyc.yale.edu/

> "Mars has a radius of 3.39 x 10^6 m and a surface gravity of 3.73 ms^{-2}. Calculate the escape velocity for Mars and the typical speed of a CO2 molecule (assume T = 250 K). How can Mars retain its CO2 atmosphere? (Hint: the molecular weight of carbon dioxide is 44. Use the formulae given in class.) [...]"

`biomed-eng-1` provides a 7-page glossary of technical terms.

> "[...] active transport - the transport of molecules in an energetically unfavorable direction across a membrane coupled to the hydrolysis of ATP or other source of energy
>
> ATP (adenosine 5'-triphosphate) - a nucleotide that is the most important molecule for capturing and transferring free energy in cells. Hydrolysis of each of the two high-energy phosphoanhydride bonds in ATP is accompanied by a large free-energy change ("G) of 7 kcal/mole
>
> aquaporin – a water channel protein which allows water molecules to cross the cell membrane much more rapidly than through the phospholipid bilayer [...]"

`human-nature-8` provides reading assignments for four books with short summaries each.

> "[A] Epictetus, The Handbook
>
> Background information about the Stoic philosopher Epictetus (c. 50-130 CE) and his famous work Encheiridion (The Handbook) appears in Nicholas White's introduction to our translation. White has also added footnotes that explain points of potential confusion.
>
> As the title indicates, The Handbook is intended as a tidy introduction to a more complex philosophical outlook. It is written in an accessible and engaging style.
>
> The Stoic movement originated around 300 BCE and flourished for over five hundred years. The Stoics believed that the external world is deterministic: its state at any time is completely determined by its prior states. So, they maintained, it is pointless to wish for things to be different because to do so is to wish for something impossible. A wise person would, therefore, accept whatever befalls them without desiring that things go otherwise – hence the English word 'stoic.'
>
> Passages to focus on/passages to skim
>
> I encourage you to read the text in full, at a steady reading pace. [...]"

`psy-14/5` and `enviromental-8` provide ~10-page slides with a typical amount of text.

3.1.1 Conclusion

Only about 20% of the courses have lecture material at all; only about 20% of these courses actually have typical "slides" – the rest provides heterogenous other kinds of material. While it cannot be inferred from this dataset that this is a general condition, it nevertheless shows a clear "real-world" disadvantage of an approach only relying on those materials. We will look at the impact of the varying quality and quantity in the analysis later on.

4 The LM-Interpolation approach

I will now describe the LM-Interpolation approach. The high level overview is as follows: we will use the open source speech recognition framework Sphinx 4[8] as the software for performing speech recognition. Sphinx 4 has a modular architecture which allows specifying components of the whole process per configuration. It provides multiple implementations of LMs[9], the default one being an n-gram model.

It also provides an `InterpolatedLanguageModel`[10] (ILM) which allows to specify multiple LMs and weights and interpolate the probabilities for a given n-gram from all models' probabilities ($p = w_1 * p_1 + w_2 * p_2 + \ldots$ where w_n are the weights ($\sum_{i=1}^{n}(w_i) = 1$) and p_i is the probability for a given n-gram in LM_i).

The purpose of the ILM in our approach is to factor in the importance of keywords. These keywords have to be supplied in the form of an n-gram language model. For this we extract text content from the lecture material, preprocess it and create an n-gram LM from the resulting corpus. Sphinx 4 is then a) run with a generic English n-gram LM only and b) with the ILM configured to use the generic English LM and the keyword language model in a 50/50 weighting. Finally the two resulting transcriptions are compared with a selection of metrics.

As an example, the 1-gram *sex* has a probability of 2.82% in the keyword model of `psy-14`, but a probability of 0.012% in the generic English LM.[11] When applying 50/50 interpolation, the result is $2.82\% * 0.5 + 0.012\% * 0.5 = 1.416\%$, which is an increase by the factor ~117 over the generic probability.

I will now give an overview over the Sphinx 4 architecture, followed by a description of the implementation of the approach.

4.1 Sphinx 4 architecture

Sphinx 4's overall architecture as shown in figure 3 is comprised of three primary modules: the *FrontEnd*, the *Decoder* and the *Linguist*.[12] The *FrontEnd* takes one or more input signals and parameterizes them into a sequence of *Features*. The *Linguist* takes any type of standard language model, pronunciation information from a phonetic *Dictionary* and information from one or more sets of *AcousticModels*, generating a *Search Graph*. The *SearchManager* in the *Decoder* uses *Features* from the *FrontEnd* to perform decoding. Each of the components (written italicized) are interfaces for which the actual implementation can be configured at runtime.

[8] Homepage: http://cmusphinx.sourceforge.net/wiki/sphinx4:webhome

[9] Overview: http://cmusphinx.sourceforge.net/doc/sphinx4/edu/cmu/sphinx/linguist/language/ngram/LanguageModel.html

[10] Javadoc: http://cmusphinx.sourceforge.net/doc/sphinx4/edu/cmu/sphinx/linguist/language/ngram/InterpolatedLanguageModel.html

[11] (*cmusphinx-5.0-en-us.lm*, 2015)

[12] The following description is based on Walker et al. (2004).

4.1.1 FrontEnd

The frontend parameterizes an *Input* signal into a sequence of output *Features*. This process is realized as a one or multiple chains that can run in parallel, permitting simultaneous computation of different types of parameters from the same or different input signals.

4.1.2 Linguist

The *Linguist* encapsulates the details of generating a *SearchGraph* used by the *SearchManager* in the *Decoder*. It consists of the *LanguageModel*, the *Dictionary* and the *AcousticModel*.

LanguageModel implementations typically can be categorized into graph-driven grammars and n-gram models. The latter are primarily differentiated by their usage parameters (e.g. small vs. very big LMs; input formats).

Dictionary provides the pronunciations for words found in the *LanguageModel*. A G2P Model can be specified in the configuration which will statistically model missing entries in the dictionary based.

The *AcousticModel* provides a mapping between a phoneme and an HMM that can be scored against incoming *Features* provided by the *FrontEnd*, also taking word-contextual information into account.

4.1.3 SearchGraph

The primary data structure in the decoding phase is the *SearchGraph*. It is a directed graph which nodes are *SearchStates* that are either *emitting* or *non-emitting*. The *emitting* state can be scored against an incoming acoustic feature, while *non-emitting* state represent higher-level linguistic constructs such as words and phonemes that are not directly scored against those features.

4.1.4 Decoder

The *Decoder* uses *Features* from the *FrontEnd* by using a *SearchManager*, that controls the *Linguist's SearchGraph* to generate *Result* hypotheses. The *Decoder* tells the *SearchManager* to recognize a set of *Feature* frames. At each step, the *SearchManager* creates a *Result* object that contains all paths that "reached a final non-emitting state".

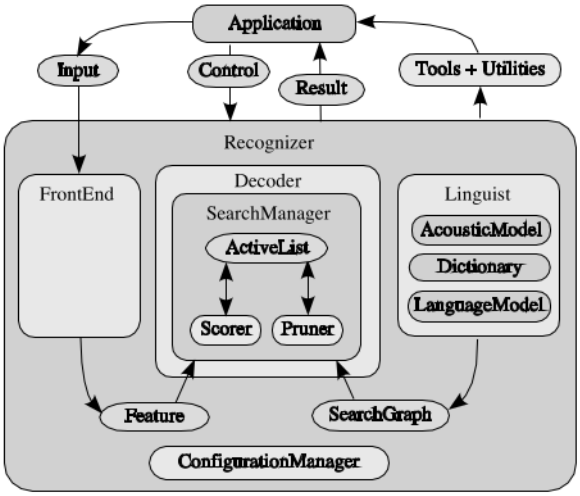

Figure 3: Sphinx 4 architecture

4.2 Implementation

The pipeline is implemented with a collection of standalone command line tools and a set of shell and Python scripts[13].

The tasks are the following, in chronological order:

1. **Prepare the input**
   - The audio file is converted into Sphinx 4 compatible format (16khz, 16bit mono little-endian).
   - A testcase folder with a given shortname (e.g. `psy-15`) is created in the `results`-directory[14] of the source code repository.
   - The reference transcript, the material (PDF format is required) and the converted audio file are moved into a `resources` subfolder of the testcase folder.

2. **Create a keyword LM from lecture material**
   - `pdftohtml -i -xml` is applied on the given material PDF. The XML output representation is input to `pdfreflow`[15]. Compared to the tool `pdftotext` the combination of these 2 tools preserves paragraphs

[13] The source code is available here: https://github.com/jonathanewerner/bachelor/tree/master/bin

[14] https://github.com/jonathanewerner/bachelor/tree/master/results

[15] pdftohtml (http://pdftohtml.sourceforge.net/) and pdfreflow (http://sourceforge.net/projects/pdfreflow/) are open source linux command line utilities

correctly, whereas `pdftotext` represents each line break in the input PDF as a new paragraph in the output text file. This is a significant disadvantage for the LM creation step, as a newline in the input file there has the semantic "end of sentence" – so that a sentence split into 4 lines by `pdftotext` would count as 4 sentences in the LM.

- The HTML output from `pdfreflow` is filtered by taking only relevant HTML-tags such as `<p>`'s (paragraphs) and `<blockquote>`'s, further improving the content-to-noise ratio.
- The resulting text is then preprocessed for optimal compatibility with the LM creation tool by removing punctuation and superfluous whitespace[16].
- The resulting corpus is input to `estimate-ngram`, a LM creation tool from the MIT Language Modeling Toolkit[17] (MITLMT).

For clarification intermediate results from this step follow as an example. They are from the test case `psy-5`[18]. Figure 4 shows an example slide.

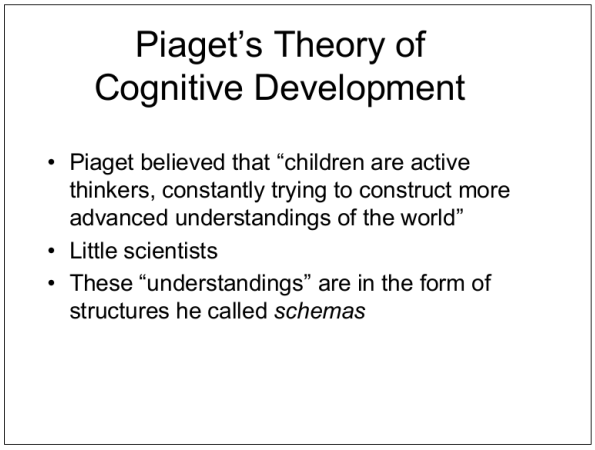

Figure 4: Slide from lecture `psy-5`

When using `pdftotext` the result looks like this for the given slide:

```
Piaget's Theory of
Cognitive Development
• Piaget believed that "children are active
thinkers, constantly trying to construct more
advanced understandings of the world"
```

[16] I use a combination of command line text processing (sed) and a perl script from Stephen Marquard here.

[17] https://code.google.com/p/mitlm/wiki/EstimateNgram

[18] "Introduction to Psychology, 5 - What Is It Like to Be a Baby: The Development of Thought"

```
• Little scientists
• These "understandings" are in the form of
structures he called schemas
```

Notice how each newline in the slide maps to a newline in the output. When using the combination of `pdftohtml` and `pdfreflow` the result looks like this:

```
<p class="p9">Piaget's Theory of
Cognitive Development </p>
<p class="p10">•  Piaget believed that "children are active
thinkers, constantly trying to construct more
advanced understandings of the world" </p>
<blockquote class="b9">•  Little scientists </blockquote>
<p class="p10">•  These "understandings" are in the form of
structures he called <i>schemas</i> </p>
```

Notice how a paragraph is captured in a `<p>`-tag. This allows extracting a sentence as one line in the corpus. After applying the preprocessing described above the final corpus for the slide looks like this (where ".." marks a line continuation):

```
piagets theory of cognitive development
piaget believed that children are active thinkers constantly
.. trying to construct more advanced understandings of the world
little scientists
these understandings are in the form of structures he called schemas
```

3. **Convert transcript to reference corpus**

   The transcript from Open Yale is supplied as HTML. We apply processing steps to transform it to a corpus ready to be consumed by the WER analysis tool (no punctuation, all lowercase). As these are specific to just the format chosen by Open Yale Courses, the details are omitted, as they have no general use.

4. **Run Sphinx 4 in baseline and interpolated mode**

   `bin/sphinx-interpolated.py`[19] supplies a wrapper for interfacing with Sphinx 4. The Java API of Sphinx 4 is exposed for command line usage by a a JAR package which bundles the Sphinx 4 libraries and a small Main class. This class uses command line arguments supplied from `bin/sphinx-interpolated.py` to correctly configure Sphinx 4 and start the actual recognition.

   Each testcase folder has a configuration file which specifies which models the test run should use:

[19] https://github.com/jonathanewerner/bachelor/blob/master/bin/sphinx-interpolated.py

```
{
  "acousticModelPath": "en-new/cmusphinx-en-us-5.2",
  "dictionaryPath": "en-new/cmudict-en-us.dict",

  "languageModelPath": "en-new/cmusphinx-5.0-en-us.lm",
  "keywordModelPath": "model.lm",
  "g2pModelPath": "en-new/en_us_nostress/model.fst.ser",

  "resultsFolder": "biomed-eng-1"
}
```

`bin/sphinx-interpolated.py` interprets the "global" models relative to the repository root-folder `models`, the `resultsFolder` relative to the root folder `results` and the `keywordModelPath` relative to the `resultsFolder`. It then supplies the absolute paths to the JAR. It also supplies absolute output file paths for the transcription result and transcription word timings results.

This setup ensures reproducible results, as the environment of a given testcase is exactly specified (as long as the same binaries and script versions are assumed).

`bin/sphinx-interpolated.py` can now be used to run the baseline or/and interpolated version.

5. **Analyze and compare the results**

   Finally the results from the two recognition runs are analyzed and compared by running `bin/hotword-analyze <testcase folder name>`. This performs two things: a) WER comparison and metrics generation and b) keyword visualization.

   5.1 *WER comparison and metrics generation*

   This first calls `bin/wer.py`[20] on each run, which will calculate the WER and show a summary of substituted (SUB), inserted (INS) and deleted (DEL) words when comparing the reference (REF) to the hypothesis (HYP):

```
OP  | REF     | HYP
INS | ****    | this
INS | ****    | is
INS | ****    | that
OK  | this    | this
OK  | is      | is
OK  | a       | a
OK  | course  | course
```

[20] `wer.py` has been adapted from http://progfruits.blogspot.de/2014/02/word-error-rate-wer-and-word.html

```
SUB | a       | that
SUB | version | aversion
OK  | of      | of
OK  | which   | which
SUB | i've    | i
OK  | taught  | taught
INS | ****    | him
...
...
{'Sub': 1230, 'Ins': 674, 'WER': 0.316, 'Del': 324, 'Cor': 5492}
```

In a second step it compares the two WER result files with `bin/compare-wer.py`.

The result is an HTML file with a) shows a WER comparison table and b) various statistical measures which will be explored later. The table (shown in Figure 5) colors correctly recognized words as green and incorrect words as red. It also marks words that have been improved in the interpolated version with a green border and words that have been worsened with a red border.

5.2 *Keyword visualization*

Data from the reference and the recognition results is compiled into a format suitable for consumption by a visualisation module, which will be discussed in chapter 6.

All intermediate steps from the pipeline are represented as files in the testcase folder. Table 2 gives an overview of the files created by each pipeline step.

Table 2: File results of a testcase run

File	Description
Step 1: Prepare the input	
`resources/audio.mp3`	original audio
`resources/audio.wav`	converted audio
`resources/slides.pdf`	lecture material
`resources/transcript.html`	lecture transcript
`config.json`	run configuration
Step 2: Create a keyword LM	
`slides.corpus.txt`	lecture material corpus
`model.lm`	keyword LM
Step 3: Convert reference to corpus	
`reference.corpus.txt`	reference transcription corpus

File	Description
`reference_wordcounts.json`	reference transcription word counts[21]
Step 4: Run Sphinx 4	
`sphinx_log_baseline.txt`	Sphinx 4 logging output
`sphinx_log_interpolated.txt`	
`sphinx_result_baseline.txt`	Sphinx 4 transcription
`sphinx_result_interpolated.txt`	
`sphinx_word_times_baseline.txt`	Sphinx 4 word times
`sphinx_word_times_interpolated.txt`	
Step 5.1: WER comparison / metrics generation	
`results.json`	run metrics in json format[22]
`wer_baseline.txt`	WER table / metrics
`wer_interpolated.txt`	
`wer_comparison.html`	rich WER comparison + metrics
Step 5.2: Keyword visualization	
`cloud_baseline.json`	data representation for visualization
`cloud_interpolated.json`	

[21] They are needed for the visualization later.
[22] This eases parsability for aggregating multiple testcase results later.

Reference	baseline	interpolated
this	this	this
is	is	is
a	a	a
course	course	course
a	that	ab
version	aversion	version
of	of	of
which	which	which
i've	i	i
taught	taught	taught
almost	almost	almost
every	every	every
year	year	year
for	for	for
the	the	the
last	last	last
twenty	the	the
years	years	years
and	and	added
it	it	a

Figure 5: WER comparison

5 Analysis

I will now discuss how to evaluate the usefulness of the LM-Interpolation approach in light of the goal to improve recognition accuracy of interesting keywords.

5.1 Approaching a good metric

We want to find a metric that describes if and how much the interpolated version improves upon the baseline version. The "canonic" metric used to evaluate speech recognition performance is called *Word Error Rate* (WER). The WER is derived from taking the Levenstein distance at the word level between a reference transcript and the recognition result. WER is calculated as:

$$WER = \frac{S + D + I}{N}$$

where

- S is the number of substitutions
- D is the number of deletions
- I is the number of insertions
- N is the number of words in the reference ($N = S + D + C$, where C is the number of correct words)

This approach circumvents the problem that the reference and the recognition result get "out of sync" when words are inserted or removed.

The problem of using WER for our purposes is twofold: 1. it does not help answering the question of how much our approach improves the accuracy of keywords; it only pertains to *all* words. 2. it penalizes *insertions*. But insertions are not relevant to the question how many of the keywords from the reference transcript have been detected. Insertions only matter when we are interested in the *sentence integrity* of the result transcript. In our case, we do not care if there are superfluous words *between* correctly recognized keywords[23].

These considerations lead to the conclusion that a first improvement over regular *WER* would be to just look at the *detection rate* of words, where detection rate describes the rate of words from a transcript that have been correctly recognized. This can be expressed by using WER in a first step in order to get the benefit of word synchronization and then filtering out insertions in a second step. It is not necessary to filter out deletions as those can be seen as non-detected words.

[23] Although there might be cases of technical terms that are compound words – in that case a insertion would be problematic. Detection of this edge case is beyond the scope of our approach, however.

A next improvement consists in just considering the words which are part of the material corpus. By just taking the words that we used to create the material corpus for our interpolated LM we can precicely evaluate the improvement effect of this change. This is still not perfect however. The lecture material corpus includes a substantial amount of words that would not be classified as "keywords": filler words and very common words. One approach to sort them out is subtracting a set of top x most common words (top_X *words*) from this list. The resulting metric can then be parameterized on the given x. This is an idea that Marquard uses when he proposes the metric "Ranked Word Correct Rate" (RWCR-n):

> "RWCR-n is defined as the Word Correct Rate for all words in the document which are not found in the first n words in a given general English word dictionary with words ranked from most to least frequent." (Marquard, 2012, p. 71)

5.1.1 Lemmas

When searching for a specific term the user is interested in the *lemma*[24] for a given word: when he wants to find occurences of *child* in the given lecture, occurences of "children" would also be relevant. This implies two things: 1) when looking at the "atomic" level of improvements and degradations it is more relevant to have lemmas as atoms and not words and 2) the exact matching (a hypothesis word is only "correct" if it exactly matches the reference word) should be "loosened" to also mark hypothesis words as correct if their lemmatized version matches the reference.

The same principle holds for the top_X words: we only want to capture words for which the *lemma* is not in the top_X words.

5.1.2 Definition of KWDR-x and WDR

We can distill these concerns into a definition of a metric called *KWDR-x*, which expresses the "Keyword Detection Rate", where a keyword is defined as the lemma of a word occuring in a given lemmatized lecture material corpus given that this lemma is not present in the lemmatized top_X list of most common words of the given language. A keyword is *detected* when it is *lemma-equal*[25]to the corresponding word in the reference; if it is different or deleted, it is not detected.

The *Word Detection Rate* (WDR) is analogously defined as the rate of detected words, where a word is detected when it is lemma-equal to its reference.

[24]A lemma, also called "headword", is the canonical form of a word that is chosen by convention to represent a group of lexemes that refer to the same meaning. A typical use would be the key of an dictionary entry.

[25]Two words are called *lemma-equal* when their lemmatized versions are equal.

The value of x has to be determined empirically: how many of the top words should be filtered out? There has to be a balance between not accidentally excluding keywords (i.e "sex" is in the in the top_{500} words) and filtering out enough filler words.

After experimenting with some values I went with $x = 500$ for my measurements. It is hard to find a less ad hoc approach to determining the "best" x, as you have no "meta"-metric that assesses how well a given x captures the goal of accurately describing the detection accuracy of keywords; it necessarily is a "best guess". $x = 500$ was chosen by looking at the relationship from x to $\Delta KWDR$ (the improvement in KWDR-500) as shown in figure 6. For values of x below ~200 the growth of improvement is explained by the gradual removal of filler words from the set of keywords. Their recognition accuracy is not improved by our approach which is why they prevent the accuracy improvement of actual keywords to be visible. Above ~200 this factor is ruled out and the chosen x value has only miniscule influence on the resulting KWDR.

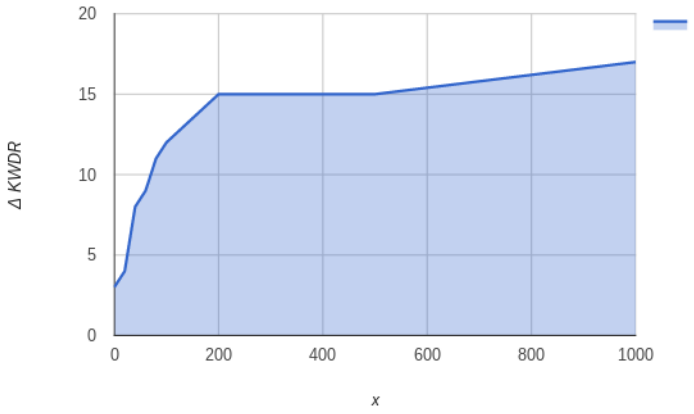

Figure 6: Relation of x to $\Delta KWDR$, compiled by running the analysis on lecture `geology-8` with the x-values 0, 20, 40, 60, 80, 100, 200, 300, 400, 500 and 1000.

This strategy, while depending on choosing an "ad hoc" value, was sufficient to validate our approach because it was possible to manually evaluate the metric's "precision" by observing the resulting word sets. Another approach would have been to take the *tf-idf* (Term Frequency - Inverse Document Frequency) as a criterion for "keyword-ness" of words. *tf-idf* computes the "relevance" of a word in the context of a document by taking into account the occurences of the word in the document offset by the word's frequency in a broader corpus. This way common words are rated lower although they occur frequently in the given document. This way there is no need for the arbitrary aspect of choosing an value of x and the negative side effect of accidentally excluding a keyword. On the other hand there would be the need to choose a treshold *tf-idf* score which would have to be met for inclusion into the keyword set.

5.1.3 Secondary metrics

The following secondary or derived metrics are also evaluated:

- W: Number of words
- KW: Number of keywords
- $WER_{A|B}$: WER of baseline (A) / interpolated version (B)
- $WDR_{A|B}$: WDR of baseline (A) / interpolated version (B)
- $KWDR_{A|B}500$: KWDR-500 of baseline (A) / interpolated version (B)
- $W_{worse|improved}$: Proportion of worsened/improved words
- $KW_{worse|improved}$: Proportion of worsened/improved keywords
- $W_{worse|improved}(K)$: Proportion of worsened/improved words that are keywords
- E: $W_{improved}(K) - W_{worse}(K)$: A percentage score for "effectiveness" of version B

Their use can be exemplarily demonstrated on the lecture `human-nature-8`: The lecture has 5342 words overall, of which 333 are keywords. When looking at the general WDR, run A and B both have a score of 57%. This can be "explained" by looking at $W_{worse|improved}$, which is 4% each, meaning that 4% (223/5342) of the words have been improved from run A to B, but 4% (221/5342) them have been worsened, which sums up to 0% difference in WDR.

Secondly, the KWDR-500 of A is 47% (156/333 keywords) versus 62% for B (121/376 keywords). This improvement of 15% can analogously be explained by looking at $KW_{worse|improved}$: when looking at the 376 keywords, 2% (7/333) of them have been worsened while 17% (58/333) have been improved. $17 - 2 = 15\%$ explains the improvement from 47% to 62%.

The last metric of $W_{worse|improved}(K)$ looks at the overall worsened/improved words and informs about the proportion of words that were keywords. As mentioned, W_{worse} is 4% (221 of the overall 5342 words have been worsened). What is the proportion of keywords in this number? Analogously, what is the proportion of keywords when looking at the overall improved words? This metric is key in identifying the *effectiveness* (E) of our approach: the $W_{improved}(K)$ value answers the question how well our approach is targeted towards improving the words we are interested in, the $W_{worse}(K)$ value answers the question how big the "side effect" of worsening keywords is. In the example, $W_{worse}(K)$ is 3% (7/221) and $W_{improved}(K)$ is 26% (58/223). This is great: of the 221 overall worsened words only **7** were relevant given our goals. In essence, we can interpret $W_{improved}(K) - W_{worse}(K)$ as an **effectiveness score**, the same way we interpret the difference between $W/KW_{improved}$ and W/KW_{worse} as "singular" metrics (WDR and KWDR respectively). We can say that our example had an effectiveness of $26 - 3 = 23\%$. An effectiveness of 100% would mean that *all* words that were improved had been keywords and *none* of the worsened words would have been keywords.

5.2 Analysis implementation

The metrics are calculated automatically by `bin/wer.py` and `bin/compare-wer.py`. I will now outline some relevant implementation details.

`bin/wer.py` performs a typical WER calculation algorithm. This is run on version A and B. The results are input to `bin/compare-wer.py`. The top_{5000} words are taken from the *Corpus of Contemporary American English*[26]. The lemmatization of words is done by the `lemmatize` function of `nltk.stem.wordnet.WordNetLemmatize`.

The KWDR of A and B is then calculated by iterating over both WER results, where inserted words have been filtered out as we are only interested in the proportion of correctly recognized keywords. Each iteration operates on a tuple of (reference word, hypothesis A, hypothesis B). For both hypotheses it checks if the lemma of the reference word is equal to the lemma of the hypothesis word and possibly puts the reference word in a corresponding "wrong words" bin (one bin for A and B each). If both hypotheses were correct the next iteration is performed; else the given hypothesis is put into an "improved" or "worsened" bin (if A was correct and B was false, A will be put into a "worsened" bin to remember what the correct version of the word was that B detected wrong; if it is the other way round, B will be put into the "improved" bin.) Finally, if an hypothesis was wrong *and* in its lemma is in the lemmatized, top_x-words excluded, material corpus[27], the reference will be put into an "wrong and keyword" bin (also one for each run)[28].

After the iterations are finished, the mentioned metrics are calculated by simple proportional calculation, taking into account the size of the resulting bins and overall measurements (like reference word count).

As side effects, the bin contents as well as the calculated metrics are exported to HTML and JSON for further consumption for manual inspection or result aggregation.

5.2.1 Example

The steps of this process can be made clearer by running through a small example. The following toy data set serves as an input:

- Material corpus: [axons the people psychology is rewarding][29]

[26] http://www.wordfrequency.info/free.asp

[27] This refers to the set of lemmas from the material corpus, minus the lemmatized versions of the top_x words; compiled at script startup.

[28] The nomenclature in the code differs slightly in that keywords are called "interesting" words there.

[29] Imagine this being taken from sparse slides with bullet points.

- Reference transcript: “Axons are firing to stimulate people’s minds.” (This gets preprocessed to “axons are firing to stimulate peoples minds” in a former step.)
- Top_x words: [i are it is the people people’s]
- The WER results for A:

```
OP  | REF       | HYP
sub | axons     | accent
ok  | are       | are
ins | ****      | very
sub | firing    | tiring
ok  | to        | to
ok  | stimulate | stimulate
sub | peoples   | people
ok  | minds     | minds
{'Ins': 1, 'Cor': 4, 'WER': 0.571, 'Del': 0, 'Sub': 3}
```

- The WER results for B:

```
OP  | REF       | HYP
sub | axons     | axon
ok  | are       | are
ins | ****      | very
sub | firing    | tiring
ok  | to        | to
ok  | stimulate | stimulate
sub | peoples   | people
sub | minds     | may
{'Sub': 4, 'Ins': 1, 'Del': 0, 'Cor': 3, 'WER': 0.714}
```

The inputs are transformed to the following forms (removed words displayed as “**-removed**”; changed words **bold**):

- Lemmatized material corpus: [**axon** the people psychology is rewarding]
- Lemmatized top_x words: [i are it is the people **-people’s**]
- Lemmatized material corpus minus lemmatized top_x words (“keywords”): [axon **-the -people** psychology **-is** rewarding]

For the demonstration the following pseudo code conventions and abbreviations are used:

- (a, b, c) = (1, 2, 3) is a destructuring assignment, binding a to 1, b to 2, c to 3

- `ref` refers to the reference word, hypA|B to the hypothesis from version A/B
- l(word) refers to the lemmatized version of a given word
- `->` is equivalent to "thus"
- bin names:
  - wrong words from run X: `wrongX`
  - wrong keywords from run A: `wrongKW_X`

The KWDR algorithm performs with the following intermediate steps:

- **Step 0**: (`ref`, `hypA`, `hypB`) = (axons, accent, axon)
  - l(`ref`) != l(`hypA`) -> add `ref` to `wrongA`
  - l(`ref`) == l(`hypB`) -> do nothing (explanation: axon == axon)
  - `hypA` was wrong, `hypB` was correct -> add `hypB` to `improved`
  - `hypA` was wrong and a keyword -> add `ref` to `wrongKW_A`
  - `wrongA`: `[axons]`
  - `wrongB`: `[ ]`
  - `wrongKW_A`: `[axons]`
  - `wrongKW_B`: `[ ]`
  - `improved`: `[axons]`
  - `worsened`: `[ ]`
- **Step 1**: (`ref`, `hypA`, `hypB`) = (are, are, are)
  - l(`ref`) == l(`hypA`) -> do nothing
  - l(`ref`) == l(`hypB`) -> do nothing
  - both are correct -> next iteration
- **Step 2**: (`ref`, `hypA`, `hypB`) = (firing, tiring, tiring)[30]
  - l(`ref`) != l(`hypA`) -> add `ref` to `wrongA`
  - l(`ref`) != l(`hypA`) -> add `ref` to `wrongB`
  - `wrongA`: `[axons, firing]`
  - `wrongB`: `[firing]`
  - `wrongKW_A`: `[axons]`
  - `wrongKW_B`: `[ ]`
  - `improved`: `[axons]`
  - `worsened`: `[ ]`
- **Step 3/4**: (`ref`, `hypA`, `hypB`) = (to, to, to) / (stimulate, stimulate, stimulate)
  - l(`ref`) == l(`hypA`) -> do nothing
  - l(`ref`) == l(`hypB`) -> do nothing
  - both are correct -> next iteration

[30] Notice how the inserted row (marked with "INS" in the WER output) has been skipped).

- **Step 5**: (`ref`, `hypA`, `hypB`) = (peoples, people, people)
  - l(`ref`) == l(`hypA`) -> do nothing (explanation: people == people)
  - l(`ref`) == l(`hypB`) -> do nothing (same explanation)
  - both are correct -> next iteration
- **Step 6**: (`ref`, `hypA`, `hypB`) = (minds, minds, may)
  - l(`ref`) == l(`hypA`) -> do nothing
  - l(`ref`) == l(`hypB`) -> add `ref` to `wrongB`
  - `wrongA`: `[axons, firing]`
  - `wrongB`: `[firing, minds]`
  - `wrongKW_A`: `[axons]`
  - `wrongKW_B`: `[ ]`
  - `improved`: `[axons]`
  - `worsened`: `[minds]`

Results:

- `wrongA`: `[axons, firing]`
- `wrongB`: `[firing, minds]`
- `wrongKW_A`: `[axons]`
- `wrongKW_B`: `[ ]`
- `improved`: `[axons]`
- `worsened`: `[minds]`

General/derived metrics:

- W (count of words in reference): 7 ("axons are firing to stimulate peoples minds")
- KW (count of keywords in reference): 1 ([axon])
- `worsenedKW`: all from `worsened` where the word is in KW -> `[ ]`
- `improvedKW`: all from `improved` where the word is in KW -> `[axons]`

With these results we can calculate all metrics:

- $WDR_A = 1 - \frac{|wrongA|}{W} = 1 - \frac{2}{7} = 85\%$
- $WDR_B = 1 - \frac{|wrongB|}{W} = 1 - \frac{2}{7} = 85\%$
- $W_{improved} = \frac{|improved|}{W} = \frac{1}{7} = 14\%$
- $W_{worsened} = \frac{|worsened|}{W} = \frac{1}{7} = 14\%$
- $KWDR_A = 1 - \frac{|wrongKW_A|}{KW} = 1 - \frac{1}{1} = 100\%$
- $KWDR_B = 1 - \frac{|wrongKW_B|}{KW} = 1 - \frac{0}{1} = 0\%$
- $KW_{improved} = \frac{|improvedKW|}{KW} = \frac{1}{1} = 100\%$
- $KW_{worsened} = \frac{|worsenedKW|}{KW} = \frac{0}{1} = 0\%$

- $W_{improved}(K) = \frac{|improvedKW|}{|improved|} = \frac{1}{1} = 100\%$
- $W_{worsened}(K) = \frac{|worsenedKW|}{|worsened|} = \frac{0}{1} = 0\%$
- $E = KW_{improved}(K) - KW_{worsened}(K) = 100\%$

These metrics are more accessible by looking at the HTML output from `compare-wer.py` as shown in figure 7.

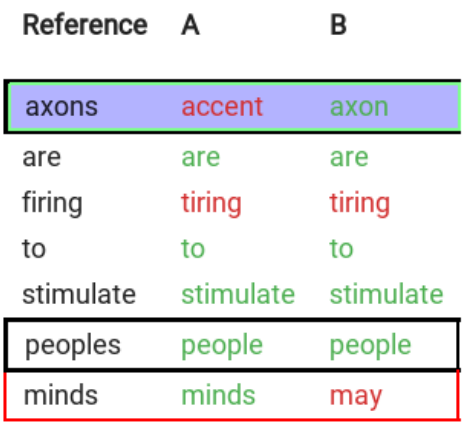

Figure 7: Visualization. Blue background: keyword. Green border: improved word. Red border: worsened word. Black border: lemmatization has changed a word.

5.3 Results

The results for the test lectures described above (chapter 3) are as follows[31] (Δ refers to the improvement from version A to B):

Table 3: Results

Lecture	1	2	3	4	5	6
W	5342	7233	7618	7142	7046	6024
KW	376	715	974	607	518	314
WER_A	52%	46%	39%	42%	32%	46%
WER_B	52%	44%	38%	42%	32%	47%
ΔWER	**0%**	**2%**	**1%**	**0%**	**0%**	**-1%**

[31] Column 2-7 represent the lectures, the numbers refer to the following lectures: 1: `human-nature-8`, 2: `environmental-8`, 3: `psy-14`, 4: `psy-5`, 5: `biomed-eng-1`, 6: `geology-8`.

Lecture	1	2	3	4	5	6
WDR_A	57%	70%	66%	63%	78%	60%
WDR_B	57%	70%	66%	63%	78%	60%
$W_{improved}$	4%	4%	5%	5%	3%	4%
W_{worse}	4%	4%	5%	5%	3%	5%
ΔWDR	**0%**	**0%**	**0%**	**0%**	**0%**	**0%**
$KWDR_A$[32]	52%	66%	67%	60%	68%	60%
$KWDR_B$	68%	82%	83%	81%	83%	78%
$KW_{improved}$	18%	16%	17%	22%	16%	20%
KW_{worse}	2%	1%	1%	0%	1%	1%
$\Delta KWDR$	**16%**	**15%**	**16%**	**22%**	**15%**	**18%**
$W_{improved}(K)$	30%	39%	41%	40%	44%	26%
$W_{worse}(K)$	3%	2%	2%	0%	2%	1%
E	**27%**	**37%**	**39%**	**40%**	**42%**	**25%**

The means are:

Table 4: Result means

Metric	Mean in %
WER_A	42.3%
WER_B	42.5%
ΔWER	**0.3**%
WDR_A	65.6%
WDR_B	65.6%
ΔWDR	**0.0**%
$KWDR_A$	62.2%
$KWDR_B$	79.2%
$\Delta KWDR$	**17.0%**
E	35.0%

5.4 Interpretation

Several things are notable. The WDR as well as $W_{improved}$ and W_{worse} nearly don't change at all, the differences are only zero-digit absolute amounts. It is interesting that the results are so unambiguous in this respect; it is also unexpected that $W_{improved}$ and W_{worse} always cancel each other out completely.

[32] KWDR means KWDR-500 for brevity if not noted otherwise.

Assessing the $\Delta KWDR$ presents the challenge that no comparison is available that uses the exact same metric. However it is possible to "fuzzily" compare the performance by looking at metrics with the same basic idea.

The metric "RWCR-n" used by Marquard (2012) mentioned above is comparable, as it also uses the concept of filtering out the top_n most frequent words; it differs by not taking the lemmatized word version as their atomic unit. With that said, the average improvement in RWCR-10k over 13 lectures also taken from Open Yale Courses is 9.0%, while their average WER decreases by 0.8%.

Kawahara et al. (2008) use a metric called "Keyword Detection Rate", where keywords are defined as content words (nouns and verbs excluding numbers and pronouns) that appear in the slide text. They then compute the f-measure (the "mean of the recall rate of keywords included in utterances and the precision of keywords detected in ASR results."). They report improvements of 7.5% and 3.0% (for two test sets) in detection rate over the baseline accuracy, while the increase in WER is 2.2% and 1.3% over the baseline respectively[33].

Miranda, Neto, & Black (2013) do not use a custom metric and report a WER improvement of 3.6%, when interpolating the LM with slide text contents; they achieve an improvement of 5.9% WER when using their proposed method of integrating the speech input with synchronized slide content.

While comparing WER performance has the discussed disadvantage of low relevance to the given evaluation goals and the non-standardized spectrum of custom metrics disallows an objective comparison of the different approaches, it yet gives an impression how our approach's performance relates to other work: the $\Delta KWDR$ of 17.0% seems like a good indicator that our approach is a viable solution for the goal of improving speech recognition for searchability and scannability. Additionally, the *effectiveness score* demonstrates that the approach nearly does not worsen keywords at all and 35% of the improved words are actually keywords.

In general, the uniform distribution of results over the various topic domains with their very different types of provided materials is also suprising. The results seem to suggest that the form and supposed "quality" of material (e.g. excercise sheet versus lecture slides) does not correlate with the improvement in KWDR. The initial assumption that lectures from the natural and formal sciences would be harder to recognize, based on the "naive" presumption that words like "adenosine 5'-triphosphate" would be impossible to recognize, seems to be invalid as well – apparently the combination of preprocessing, G2P and adapted weighting in the LM makes it possible to detect complicated technical terms like this as well.

5.4.1 Qualitative interpretation

While representing the performance of our approach with a set of metrics allows (at least internal) comparability of results, it can not convey a holistic

[33] The mentioned results refer to the combined method of global and local adaptation.

impression of what would actually change for a user of a hypothetic speech media search/scan interface when using data generated with our approach versus the baseline approach.

This impression can be given by looking at the following detailed results of the run on the `biomed-eng-1` lecture.

Normal words improved

(of, 8) (that, 7) (the, 6) (or, 6) (and, 5) (a, 5) (in, 4) (to, 4) (is, 4) (it, 3) (course, 3) (into, 2) (an, 2) (your, 2) (from, 2) (than, 2) (one, 2) (those, 2) (this, 2) (talk, 2) (bridge, 1) (set, 1) (don't, 1) (some, 1) (are, 1) (annoying, 1) (really, 1) (again, 1) (there's, 1) (would, 1) (it's, 1) (there, 1) (how, 1) (version, 1) (we're, 1) (which, 1) (you, 1) (more, 1) (week, 1) (be, 1) (students, 1) (free, 1) (i've, 1) (with, 1) (by, 1) (distance, 1) (about, 1) (like, 1) (well, 1) (infectious, 1) (yale, 1) (very, 1) (where, 1) (engineers, 1)

Normal words worse:

(and, 15) (a, 10) (so, 9) (you, 8) (the, 8) (it, 7) (have, 6) (to, 5) (they're, 4) (of, 4) (that, 4) (are, 3) (can, 3) (be, 3) (we, 3) (on, 3) (at, 3) (in, 3) (how, 3) (online, 3) (that's, 3) (day, 2) (we'll, 2) (see, 2) (our, 2) (for, 2) (genes, 2) (could, 2) (it's, 2) (one, 2) (there, 2) (we're, 2) (but, 2) (is, 2) (as, 2) (if, 2) (two, 2) (principle, 2) (concept, 1) (office, 1) (years, 1) (london, 1) (go, 1) (just, 1) (had, 1) (easy, 1) (bridge, 1) (somebody, 1) (increased, 1) (very, 1) (familiar, 1) (safe, 1) (i've, 1) (every, 1) (they, 1) (now, 1) (organ, 1) (did, 1) (doctor's, 1) (because, 1) (old, 1) (some, 1) (really, 1) (what, 1) (said, 1) (lots, 1) (vessels, 1) (health, 1) (approach, 1) (patient, 1) (here, 1) (come, 1) (about, 1) (bow, 1) (or, 1) (cancer, 1) (point, 1) (period, 1) (long, 1) (apply, 1) (city, 1) (would, 1) (leading, 1) (three, 1) (been, 1) (their, 1) (way, 1) (was, 1) (tell, 1) (life, 1) (buy, 1) (posted, 1) (physician, 1) (these, 1) (say, 1) (us, 1) (patient's, 1) (thin, 1) (were, 1) (heart, 1) (an, 1) (heard, 1) (get, 1) (other, 1) (details, 1) (week, 1) (kinds, 1) (i, 1) (mechanical, 1)

KW improved:

(biomedical, 35) (dna, 7) (cells, 7) (engineering, 6) (biochemistry, 3) (cell, 3) (polymer, 2) (graph, 2) (gibbs, 2) (certain, 1) (energy, 1) (site, 1) (occur, 1) (plot, 1) (due, 1) (specifically, 1) (membrane, 1) (answer, 1) (has, 1) (higher, 1) (drugs, 1) (molecule, 1) (known, 1) (post, 1) (polymers, 1) (disease, 1) (order, 1)

KW worse:

(cells, 1) (maintain, 1) (beyond, 1) (genetic, 1) (due, 1)

You notice two things: a) the "exchange" of filler words from version A to B and vice versa, which is of no interest for searching and scanning, and b) interesting keywords that have substantial amounts of occurrences that were not found before, while the amount of worsened KW is tiny. This is the important

"qualitative", high-level conclusion: the approach allows users to find technical terms in speech media which they weren't able to find before and it works consistently over a broad spectrum of topics.

6 Visualization for scannability

We have shown that the LM-Interpolation approach is a viable tool for improving recognition accuracy of keywords on university lectures. The output data of our system are words with meta information: their associated timing and if they are keywords. How can this information be further used for helping a user with the task of scanning and searching through a given lecture? While it is technically possible to use the whole transcript and present the user an interface were the transcript is time-aligned with the lecture, that presentation is problematic as the *WER* of the transcript has not been improved and reading comprehension for texts with WERs above 30% is too low.

A better approach would be focusing the interface exclusively on the keywords in such a way that the provided timing meta information is transformed into a dense visual representation, thus making scanning possible. The user should be able to see the distribution of topics during the timeline of the lecture with *once glance*.

To this end I have developed a prototype implementation of such an interface[34]. It features two views: the first one is a list of word timelines (Figure 8). A word timeline shows the distribution of occurences of a given word over the time of the lecture. An occurence is displayed as a dot; clicking the dot posititions the corresponding lecture audio at the time the word occurence is spoken. The timelines are vertically sorted by count of word occurences. For analytical purposes the interface also shows the count of recognized occurences in relation to the actual count of occurences in the reference transcript, seen next to the word. It also overlays a graph which shows the *word density* at a given time point. The density function is calculated by performing a Gaussian Kernel Density Estimation (KDE) algorithm on the array of time positions for a given word. The red dots are local maxima of the function[35], so that a word can have multiple maxima. The information about maxima is being used primarily in the second view.

The second view (Figure 9) is a *word cloud* with "semantic axes", compared to regular word cloud visualizations where the axes don't have meaning. The x-axis still is the time-axis of the lecture and the y-axis still is the keyword frequency. The central feature of this cloud is that it can show *multiple instances* of one keyword – one instance for each local maximum. The word instance is on the same point on the x-axis as the corresponding local maximum. The timeline for a word can be shown by clicking on it. The example shows "brain" in the activated state; the timeline shows up below the map. One can see the two instances of

[34] The source code is available at https://github.com/jonathanewerner/bachelor/tree/master/viz. The prototype is implemented with web technology (Javascript, interactive SVGs, React.js, CSS) with the goal of easing possible integration into existing web video portals.

[35] The local maxima are computed with the `scipy.signal.argrelextrema` function from the python `scipy` package and had some mildly surprising results, which were of no relevance for the interface prototyping task however.

Figure 8: Word timelines

"brain" being horizontally aligned with the two local maxima below[36]. Clicking on the word also transports the audio to the position of the word next to the given local maximum. The font size of the word is computed by counting the word occurrences for which this maximum is the nearest. Additionally multiple instances of one keyword have the same color to further aid scanning by allowing the brain to pre-attentively process the representation.

This view allows an user to immediately scan the distribution of topics during the whole lecture. If particularly interested in the parts about the brain, he/she might click on "brain", be immediately transported to the relevant audio position and additionally have a more in-depth view in the bottom timeline below the cloud, allowing him/here to intuitively grasp how long the relevant part might be, maybe skipping around by clicking on other instances of the word in the timeline.

You could imagine integrating this interface as a semi-transparent overlay view on a video player, for example on platforms like lecture2go[37], the lecture video streaming platform used by the University of Hamburg. When using a system that integrates many lectures in one database like this, it would also be possible to not only link to keyword instances in the same lecture but also on a broader scope, e.g the whole course or even other relevant courses & lectures. Another interesting extension point would be to integrate human intelligence by allowing to review/score the quality of keyword instances. This would allow filtering out

[36] It is obvious here that the first local maximum for the word should rather be at about 30-40% of the word's timeline, but that could be optimized.

[37] lecture2go.uni-hamburg.de

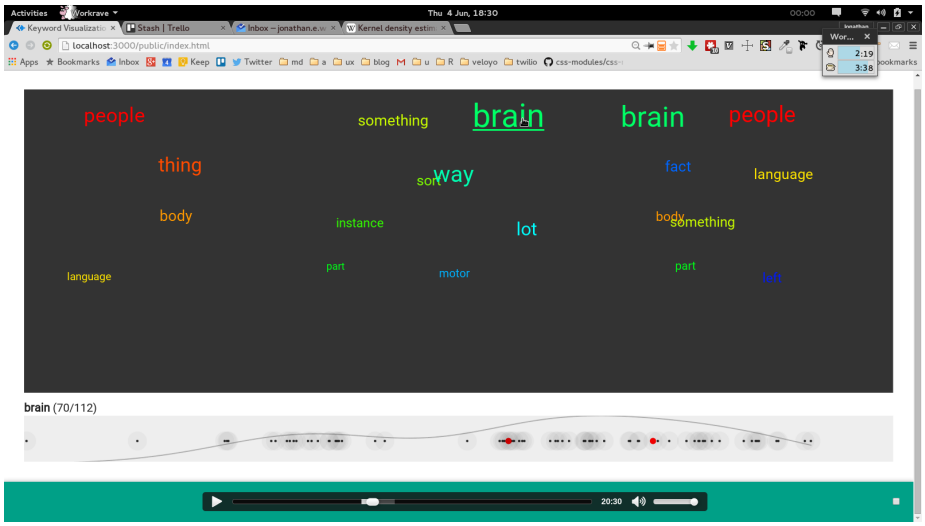

Figure 9: Word cloud

false-positives and emphasize the keyword instances that students find helpful.

7 Conclusion

This thesis asked the question: given that we are interested in improving speech recognition accuracy of keywords in university lectures, what is the advantage of creating a lecture-specific LM and interpolating it with a generic model and how can we measure this advantage?

References

Cettolo, M., Brugnara, F., & Federico, M. (2004). Advances in the automatic transcription of lectures. In *Acoustics, speech, and signal processing, 2004. proceedings.(ICASSP'04). iEEE international conference on* (Vol. 1, pp. I–769). IEEE.

cmudict-en-us.dict. (2015). https://github.com/cmusphinx/sphinx4/blob/master/sphinx4-data/src/main/resources/edu/cmu/sphinx/models/en-us/cmudict-en-us.dict.

cmusphinx-5.0-en-us.lm. (2015). http://sourceforge.net/projects/cmusphinx/files/Acoustic%20and%20Language%20Models/US%20English%20Generic%20Language%20Model/.

Cruttenden, A. (2014). *Gimson's pronunciation of English.* Routledge.

Florian, C. (1996). The Blackwell encyclopedia of writing systems. *Oxford: Blackwell.*

Glass, J., Hazen, T. J., Hetherington, L., & Wang, C. (2004). Analysis and processing of lecture audio data: Preliminary investigations. In *Proceedings of the workshop on interdisciplinary approaches to speech indexing and retrieval at hLT-nAACL 2004* (pp. 9–12). Association for Computational Linguistics.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, *29*(6), 82–97. IEEE.

Kato, K., Nanjo, H., & Kawahara, T. (2000). Automatic transcription of lecture speech using topic-independent language modeling. In *Sixth international conference on spoken language processing.*

Kawahara, T., Nemoto, Y., & Akita, Y. (2008). Automatic lecture transcription by exploiting presentation slide information for language model adaptation. In *Acoustics, speech and signal processing, 2008. iCASSP 2008. iEEE international conference on* (pp. 4929–4932). IEEE.

Lai, J., Karat, C.-M., & Yankelovich, N. (2008). Conversational speech interfaces and technologies. *The human-computer interaction handbook: Fundamentals, evolving technologies and emerging applications*, 481–491.

Marquard, S. (2012). Improving searchability of automatically transcribed lectures through dynamic language modelling. University of Cape Town.

Miranda, J., Neto, J. P., & Black, A. W. (2013). Improving aSR by integrating lecture audio and slides. In *Acoustics, speech and signal processing (iCASSP), 2013 iEEE international conference on* (pp. 8131–8135). IEEE.

Munteanu, C., Penn, G., & Baecker, R. (2007). Web-based language modelling for automatic lecture transcription. In *INTERSPEECH* (pp. 2353–2356).

Rabiner, L., & Juang, B.-H. (1993). Fundamentals of speech recognition. Prentice hall.

Walker, W., Lamere, P., Kwok, P., Raj, B., Singh, R., Gouvea, E., Wolf, P., et al. (2004). Sphinx-4: A flexible open source framework for speech recognition. Sun Microsystems, Inc.

Yamazaki, H., Iwano, K., Shinoda, K., Furui, S., & Yokota, H. (2007). Dynamic language model adaptation using presentation slides for lecture speech recognition. *Proc. INTERSPEECH 2007*, 2349–2352.