TÉCNICO LISBOA

Carnegie Mellon University

Combining multiple parallel streams for improved speech processing

João Tiago Rodrigues de Sousa Miranda

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave, PA 15213
United States of America

Departamento de Engenharia Informática
Instituto Superior Técnico
Universidade de Lisboa
Av. Rovisco Pais, 1, 1049-001 Lisbon
Portugal

Thesis Advisers:

Prof. Alan Black, Carnegie Mellon University
Prof. João Neto, Instituto Superior Técnico, Universidade de Lisboa
Prof. Luísa Coheur, Instituto Superior Técnico, Universidade de Lisboa

Thesis Committee:

Prof. Alan Black, Carnegie Mellon University
Prof. Florian Metze, Carnegie Mellon University
Prof. Isabel Trancoso, Instituto Superior Técnico, Universidade de Lisboa
Prof. Luísa Coheur, Instituto Superior Técnico, Universidade de Lisboa
Prof. Steve Renals, University of Edinburgh
Prof. Tanja Schultz, Universität Bremen

Submitted in partial fulfillment of the requirements
for the Dual degree of Doctor of Philosophy
in Language and Information Technologies.

Abstract

In a number of applications, one often has access to distinct but overlapping views over the same information. For instance, a lecture may be supported by slides, a TV series may be accompanied by subtitles, or a conference in one language may be interpreted into another. Since most useful speech and language processing technologies such as speech recognition are not perfect, it would be desirable to be able to fuse these different perspectives in order to obtain improved performance.

In this thesis, a general method for combining multiple information streams which are, in part or as a whole, translations of each other, is presented. The algorithms developed for this purpose rely both on *word lattices*, representing posterior probability distributions over word sequences, and *phrase tables*, which map word sequences to their respective translations, to generate an alignment of the different streams. From this alignment, we extract *phrase pairs*, and use them to compute a new most likely decoding of each stream, biased towards phrases in the alignment. This method was used in two different applications : transcription of simultaneously interpreted speeches in the European Parliament and of lectures supported by slides. In both of these scenarios, we achieved performance improvements when compared with speech recognition only baselines. We also demonstrate how recovering acronyms and words that cannot be found in the lattices can be used to enhance overall speech recognition performance, and propose a scheme to add new pronunciations to the recognition lexicon. Both of these techniques are also based on cross-stream information.

We also explored how rich transcription techniques, namely sentence segmentation and detection / recovery of disfluencies (filled pauses, hesitations, repetitions, etc.), can benefit from the information contained in parallel streams. Cues extracted from other streams were used to supplement currently existing methods to help solve each of these problems.

Resumo

Em muitas aplicações é possivel ter-se acesso a diferentes perspectivas sobre a mesma informação, as quais se sobrepõem parcialmente entre si. Por exemplo, uma aula pode ser acompanhada por slides, uma série de televisão numa língua estrangeira pode ter legendas associadas, ou uma conferencia pode ser interpretada simultaneamente em múltiplas línguas. Neste sentido, e dado que as tecnologias de processamento de lingua natural não são normalmente perfeitas, seria desejável poder combinar estas diferentes perspectivas, com o objectivo de melhorar a qualidade do resultado produzido.

Nesta tese apresenta-se um metodo genérico para integrar múltiplas fontes sequenciais de informação as quais podem ser vistas, no todo ou em parte, como traduções umas das outras. Os algoritmos que foram desenvolvidos para o efeito utilizam *lattices* (grafos dirigidos acíclicos que representam distribuições de probabilidades posteriores sobre sequências de palavras e que são gerados por um reconhecedor de fala) e *phrase tables*, que são tabelas que relacionam sequências de palavras e as suas respectivas traduções. Utilizando estas estruturas de dados, é gerado um alinhamento entre as diferentes fontes de informação, o qual é utilizado para produzir uma nova descodificação das fontes de informação consideradas. Este método foi especializado para duas aplicações: a transcrição de discursos do Parlamento Europeu com interpretação simultânea, bem como de aulas complementadas por slides. Em ambos os cenários, foram obtidos resultados superiores àqueles que são produzidos utilizando apenas um reconhecedor de fala. Para além disso, demonstra-se de que forma a recuperação de acrónimos e palavras que não se encontram nas *lattices* geradas pelo reconhecedor de fala pode melhorar a qualidade do reconhecimento, e propõe-se um método para introduzir novas pronúncias no léxico. Ambas as técnicas são tambem baseadas em informação extraída de múltiplas fontes paralelas.

Nesta tese, investigou-se igualmente de que forma é possível utilizar a informação contida em múltiplas fontes de informção paralelas de modo a melhorar técnicas existentes de deteção e recuperação de disfluências (fenómenos de fala que incluem pausas preenchidas, repetições e hesitações).

Contents

List of Figures

List of Tables

List of Acronyms

AM Acoustic Model

ASR Automatic Speech Recognition

BTEC Basic Travel Expression Corpus

CAT Computer Assisted Translation

CRF Conditional Random Field

DNN Deep Neural Network

EM Expectation-Maximization

EPPS European Parliament Plenary Speech

FFT Fast Fourier Transform

FST Finite State Transducer

G2P Grapheme-to-Phoneme

GMM Gaussian Mixture Model

HELM Hidden Event Language Model

HMM Hidden Markov Model

ILP Integer Linear Programming

LM Language Model

LVCSR Large Vocabulary Continuous Speech Recognition

MLE Maximum Likelihood Estimate

MFCC Mel-Frequency Cepstral Coefficients

MLP Multilayer Perceptron

MT Machine Translation

MSG Modulation-filtered Spectrogram

NLP Natural Language Processing

OCR Optical Character Recognition

OOL out-of-lattice

OOV out-of-vocabulary

PBSMT Phrase Based Statistical Machine Translation

PDF Portable Document Format

PLP Perceptive Linear Prediction

POS part-of-speech

RASTA Relative Spectral Transform

RBMT Rule-Based Machine Translation

RTF Real-time Factor

ROVER Recognizer Output Voting Error Reduction

SER Slot Error Rate

SGD Stochastic Gradient Descent

S2S Speech-to-Speech

S2T Speech-to-Text

SMT Statistical Machine Translation

SMOTE Synthetic Minority Over-sampling Technique

SNS Speech / Non-Speech

WER Word Error Rate

WFST Weighted Finite State Transducer

1 Introduction

1.1 Introduction

There is a growing number of applications with multiple parallel streams, such that the content of each of these streams corresponds to the content of the others. For example, in the European Parliament, all of the speeches, both of the plenary sessions and of the committees, are interpreted into the different languages of the 26 member states. In the United Nations, speeches must also be interpreted into the six official languages. Worldwide, both official and private actors increasingly recognize the importance of providing simultaneous interpretation as well as translation services in a globalized world. TV shows, movies and series are often dubbed in different languages. Sports events are broadcast in multiple languages simultaneously. The recurring element in all of these situations is that a similar message is transmitted multiple times, perhaps in a translated or otherwise modified form, leading to redundancy which could potentially be used to improve the performance of several speech processing tasks.

The simultaneous interpretation scenario of the European Parliament inspired the development of a parallel combination algorithm, where multiple hypotheses from recognizers in different streams are combined in order to recover from errors. This is accomplished through the use of phrase tables that connect each of the language pairs, and the use of lattices which encode multiple hypotheses compactly. We experimented with recognizers in five different languages, namely Portuguese, Spanish, English, Italian and German. We found that not only is it possible to obtain significant improvements in recognition results in each of the four languages considered, but also that the improvements increase with the number of languages used, as expected. In this way, it is possible to produce automatic transcriptions under conditions that make it hard for speech recognizers to work effectively. This method

was also extended in order to improve the detection of sentence boundaries in Automatic Speech Recognition (ASR) transcripts, using, again, this multistream information: sentence boundaries in interpreted speech are hard to locate, due to frequent speaker pauses, but their correct placement contributes to output readability. Lectures are also an important resource, encompassing a significant body of knowledge which includes, for example, conference talks or academic courses. For this reason, there has been a growing interest in technologies that might assist their efficient dissemination in multiple languages, such as automatic transcription and translation. This, in turn, motivated the extension of our parallel combination algorithm to different types of streams (other than speech streams), thus enabling it to improve the speech recognition performance of lectures which are complemented by slides. In what concerns the rich transcription of speech, we show how it is possible to improve the performance of the punctuation of speech transcripts, by using information gathered from streams in different languages. Similarly, we demonstrate improvements in the detection of disfluencies, phenomena which are know to occur in spontaneous speech. We use the alignments from multiple streams, in different languages, in order to better identify and remove these disfluencies, therefore improving the output quality. Finally, we explore the unsupervised or lightly supervised learning of speech recognition models (i.e., with reduced human intervention in the process) in two ways. We developed an algorithm to iteratively improve transcriptions, by trying to correct common errors that are found in them. We also developed a system that uses fast human corrections to automatically recognized speech, with the goal of minimizing human effort while obtaining high-quality human transcripts, and observe how it is possible to propagate user-input information to update our recognition models and further reduce the necessary effort.

1.2 Previous Work

In this section we present a general overview of related work on the topics of speech recognition, machine translation and system combination, and how they relate to the thesis topic. An analysis of previous work for specific topics will be provided in each chapter, where ap-

propriate.

1.2.1 Automatic Speech Recognition

In ASR, a computer is presented with a speech signal containing one or more sentences uttered by a speaker and is expected to produce the word sequence that was spoken. Although usually straightforward for humans, this is a complex task for computer systems due to the high variability in speech production. Phenomena such as co-articulation, different accents, or disfluencies (filled pauses, filler words, hesitations, etc.) all contribute to this variability, making speech recognition an unsolved problem under several conditions.

Despite significant advances in recent years, humans still outperform machines in speech recognition performance, especially in noisy environments. This is partly due to human superiority in recovering information from multiple sources: not only from the speech signal, but also from conversation context, visual cues such as the movement of the speaker's lips, gestures, or common sense knowledge. In fact, state-of-the-art ASR systems working with clean, well articulated speech often have word accuracies in excess of 95%, but this accuracy tends to degrade in a noisy environment, in the presence of spontaneous speech, or when transcribing speakers with non-native accents.

1.2.1.1 Applications

ASR systems are used in a wide range of applications, including:

- In dictation systems, an user dictates a text to the computer using spoken language instead of typing it in the keyboard. Usually, these systems have large vocabulary models. They can be invaluable for inexperienced computer users or in small embedded devices. The latter are often equipped with input interfaces which, due to size constraints, are not as easy to handle as a regular keyboard.
- The automatic transcription of broadcast news is useful, for instance, to the hearing impaired, or to search for news that have appeared in the past. Since it is a relatively

open-ended ASR application, it requires large vocabulary models to achieve a good coverage. *Audimus* [63] is an example of an ASR system which has been used for Broadcast News Recognition.

- Speech-to-Speech (S2S) translation systems [3, 70] ideally enable two-way communication between two speakers of different languages. They usually consist of an ASR system which converts speech in the source language into text, which is then passed on to a translation module which translates this text into the target language. Finally, a text-to-speech module is used to generate speech in the target language. The development of an S2S system must take steps to limit the propagation of errors generated by the ASR and Statistical Machine Translation (SMT) modules, in order to reduce their impact on the quality of the synthesized output.

- Spoken document retrieval systems [34, 43] index multimedia content (containing spoken data), allowing users to directly search for information that has not been transformed into text. They use an ASR system to transcribe the documents to be indexed as well as the query (if it is a spoken query rather than a text one), combining this with a retrieval engine to locate the most relevant documents.

- Spoken dialog systems [82, 54] may collaborate with a user in spoken language to complete a certain task. Alternatively, they may interact with users in a more open-ended manner, without a fixed, well-defined goal. In some cases, they can replace human operators that would otherwise be necessary or that are currently unavailable. In other situations, namely when the user is unfamiliar with the task at hand, or needs to have their hands free, dialog systems can be more natural and efficient than other interfaces.

The complexity of the task an ASR system has to solve also depends on the type of speech that it encounters. In *read* or *planned* speech, the speaker is able to anticipate the sequence of words they have to say, which usually leads to the production of a coherent utterance, without interruptions or disfluencies. This contrasts with *spontaneous* or *conversational* speech, in which the speaker generates the message as they go, and often has to increase the speaking

rate or to backtrack to correct the utterance. Since the sentences in read speech are better formed and articulated than those in spontaneous speech, they usually match better with the statistical speech recognition models, and are therefore easier to recognize. Additionally, ASR systems dealing with spontaneous speech should find an efficient way to detect and remove disfluencies, since these do not directly contribute to the message being transmitted.

1.2.1.2 Speech Recognition Fundamentals

In this section, an overview of the state-of-the-art of speech recognition is presented. The reader can find a more thorough review in [103]. Essentially, the task of a speech recognition system is to transform a set of acoustic observations into a word sequence that is as close as possible to what has been said. This problem can be recast as finding the word sequence W^* which maximizes the probability that the observations O were generated by it; in other words, we wish to determine the word sequence W^* that maximizes $P(O|W)$. Calculating this probability directly is not easy, but by using Bayes' theorem it is possible to write:

$$W^* = \arg\max_W P(O|W) = \arg\max_W \frac{P(W|O)P(W)}{P(O)} \tag{1.1}$$

This equation immediately separates the key components of a modern Automatic Speech Recognition System. While term $P(O)$ can be dropped, since it does not depend on the word sequence, term $P(W)$ assigns a probability to a given word sequence W and is known as the Language Model (LM). Finally, the term $P(W|O)$ corresponds to the acoustic model. The process through which these models are integrated to find W^* is known as decoding.

Instead of estimating parameters directly from the speech signal, the observations O correspond to a sequence of vectors obtained through feature extraction. Most feature extraction techniques convert the time domain waveforms into the frequency domain using the Fast Fourier Transform (FFT), and then perform processing to compensate for interspeaker variability or environmental factors such as channel properties, reverberation or noise. The output of this feature extraction step is a low-dimensional vector which represents a frame of speech

(in the order of 10 ms in duration); often, delta coefficients are also included in order to provide the subsequent processing steps with context from the surrounding frames. Commonly used feature extraction methods include Mel-Frequency Cepstral Coefficients (MFCC) [21] and Perceptive Linear Prediction (PLP) [37].

Each phone in the speech signal is now usually modeled as a Hidden Markov Model (HMM). An introduction to HMMs and their use in speech recognition can be found in [80]. An HMM corresponds to a first-order Markov process where the states are not directly observable, but can be inferred through the sequence of outputs, which are generated by the states. Therefore, the probability of being at a given state x_i at time i depends exclusively on the state x_{i-1}. Each state has a distribution over the output space, conditioned only on the state itself. In an HMM model, states are hidden in that one only has access to the observations - that is, it is not possible to know which sequence of states generated a given set of observations. In the particular case of speech recognition, the observations are the feature vectors and states represent, for example, the phones that were spoken.

HMMs are widely used in speech recognition due to the existence of efficient algorithms for training (Forward-Backward) and decoding (Viterbi), and because they work well in practice. However, they impose independence assumptions which fail to hold true for speech, such as requiring that the acoustic observations depend only on the current phone, without taking the surrounding context into account. For this reason, many ASR systems use triphones (the current phone concatenated with the previous and following phones) or even quinqphones, as their basic HMM units, in order to be able to model some of this context within the HMM framework.

The emission probabilities are calculated using an acoustic model. The acoustic model can be a Multilayer Perceptron (MLP). In that case, the MLP estimates the posterior probabilities of each of the output phones given the feature vectors, which are then converted to scaled likelihoods. This is known as the hybrid HMM-MLP approach [9]. Recent advances have led to the use of pre-trained deep neural networks, leading to the HMM-DNN concept [20], with significant improvements in performance. An alternative way to implement the acoustic

model is through the use of a Gaussian Mixture Model (GMM) [36], which models the emission distribution of a given state with a linear combination of Gaussian distributions. In the tandem approach [39], a neural network is trained to produce posterior probabilities that are then used as features for GMM modeling.

The acoustic model $P(W|O)$ also contains another component, the lexicon or pronunciation dictionary, which is necessary in order to convert each word w into a sequence of phones p, since the acoustic models described above estimate $P(p|O)$, where $p = p_1...p_j$. Indeed, the lexicon lists the possible pronunciations of each word w in the dictionary and therefore defines the probability distribution $P(p_1 \dots p_j|w)$. It is usually generated from a list of words manually, by a rule-based or data-driven system or a combination of these methods. Rule-based methods take advantage of the fact that, in many languages, there is a systematic relation between the surface form of a word and its pronunciation.

Language models assign a probability $P(w_1 \dots w_j)$ to each word sequence $w_1 \dots w_j$. This biases the recognizer against word sequences that are ungrammatical or unnatural in a given language. When designing a language model, a common goal is to minimize *perplexity*, which represents a measure of how hard it is to predict the next word given the language model. For example, having an LM with a perplexity of 100 in a broadcast news task is equivalent to having to choose among 100 equally probable words.

Many current speech recognizers employ n-gram models, which use a context consisting of the $n-1$ most recent words to condition the probability of the current word. In other words, n-gram models make the simplifying assumption that only the $n-1$ previous words can have an impact on the identity of the current word. While this is not true in general, many syntactic dependencies such as gender, number or tense agreement are usually short-ranged and can be efficiently captured by n-gram models. The probability of a word sequence in this model can be computed using Equation 1.2:

$$P(w_1 \dots w_j) = P(w_1)P(w_2|w_1) \dots P(w_{n-1}|w_1 \dots w_{n-2}) \prod_{i=n}^{j} P(w_i|w_{i-n+1} \dots w_{i-1}) \quad (1.2)$$

The most straightforward way to estimate n-gram probabilities would be through a maximum likelihood estimate obtained from the occurrence counts in the data:

$$P(w_n|w_1 \dots w_{n-1}) = \frac{count(w_1 \dots w_n)}{count(w_1 \dots w_{n-1})} \quad (1.3)$$

However, this would fail to account for the fact that most n-gram sequences have not been observed in the training data. It would lead the model to assign zero probability to certain word sequences that occur in the data, which is certainly undesirable. Instead, smoothing techniques [104], which redistribute some of the probability mass to unseen n-gram occurrences, are used. These include interpolation with lower-order models and back-off techniques, where a lower order model is only used if a word cannot be found in the current n-gram order.

N-gram models are limited in that they are unable to capture long-range dependencies between words, created by the syntactic or semantic properties of the language. They cannot take advantage of properties of the words themselves, such as part-of-speech, prefixes and suffixes, etc. However, they are usually preferred to other more complex alternatives because they can be efficiently trained from large amounts of data to achieve performance competitive with, or superior to, other, more sophisticated approaches to language modeling. N-gram language models also have the advantage of being easily representable as finite state machines.

The integration of all the knowledge sources (acoustic models, lexica, and language models) is performed by the decoder, in order to find the most likely word sequence according to the models. The unifying idea behind most decoders is to concatenate HMMs from sub-word units to form word HMMs, and then these to form sentence HMMs, while applying language model and lexicon weights. Then, the decoder performs a Viterbi or A^* search on the space that was so generated. Several pruning techniques are applied to reduce search complexity

and keep running time under control, most notably beam pruning, which discards the partial hypotheses outside a beam centered around the current best hypothesis.

The Weighted Finite State Transducer (WFST) [68] approach to search space modeling is a principled solution to the problem of combining all the information sources uniformly. In this approach, all the resources such as the lexicon, the language model, and the acoustic model are encoded as WFSTs and combined through well-defined operations. Such an approach has the advantage of decoupling the search algorithm from the search space, while, at the same time, being able to benefit from efficient determinization, minimization and composition algorithms developed in the finite state automata literature. Its main drawback is the need to represent all the knowledge sources as WFSTs, which may be difficult or unnatural in certain cases.

Apart from the best hypothesis, many ASR decoders also provide alternative hypotheses in the form of N-best lists (that contain the N most likely hypotheses), lattices, which are directed acyclic graphs that compactly represent a set of competing paths, or confusion networks [60], lattices with the property that every path goes through the same state sequence. The main advantage of having multiple alternatives rather than just the single-best hypothesis is that it enables downstream processing methods (machine translation, spoken document retrieval, dialog systems, etc.) to recover from ASR errors, if different information sources are available at that time.

Speech recognition systems are usually evaluated by computing the minimum number of operations (insertions, deletions, and substitutions) that are required to transform the transcript generated by the recognizer into the gold standard created by human annotators. This is achieved by first aligning the reference and the hypothesis, as in Figure 1.1. Word Error Rate (WER) is computed by adding the number of insertions, deletions and substitutions and dividing it by the number of words in the reference.

```
REF     and     the     nagoya      ---     protocol    requires
HYP      in     ---     nagoya      the     protocol    requires

        SUB     DEL                 INS
```

Figure 1.1: An alignment between the sentence "And the Nagoya protocol requires" and an ASR hypothesis. This alignment corresponds to three errors (1 substitution, 1 deletion and 1 insertion). Since there are 5 reference words in the transcription, the WER for this alignment would be 60%.

1.2.1.3 Rich Transcription

Rich Transcription is concerned with automatically annotating speech transcripts with information that can be extracted from speech transcripts, but which cannot be found in the word sequence output by the speech recognizer. For instance, most ASR systems do not insert any punctuation in the transcripts they generate, and they output all the words in the same case, regardless of what would the appropriate case for a given word. This makes it harder for the user to understand and is unacceptable for several applications.

A number of approaches have been developed which recover punctuation and perform truecasing (i.e, recover the correct case of each letter in a word, such as turning mcdonalds into McDonalds) as a post-processing technique.

The occurrence of disfluencies is usually not detected by ASR systems, apart from the presence of filled pauses. In spontaneous and conversational speech, filler words and false starts may greatly impair the performance of an ASR system, both by disrupting its language model context and by adding spurious words. In order to mitigate this problem, several methods have been developed to automatically detect, and if possible, remove such disfluencies. This is specially important in spontaneous speech. In particular, interpreted speech often has highly disfluent speech (also depending on the interpreter's skill).

There are different types of disfluencies. One possible categorization is as follows:

- *filled pauses*: interruptions in the normal flow of speech which are marked by a `uh` or `uhm` sound.

- *discourse markers*: words with no information content, used by the speaker while generating the actual message (`you know` or `well` in English, or `bom` in Portuguese). Of course, these are also speaker specific, since different speakers will use different discourse markers, with different frequencies.

- *edit disfluencies*: these are the most complex, and depending on the source, are usually composed of a deletable part, followed by an interruption point and an edit. Often they are further subdivided into repetitions, if the edit does not correct but simply repeats what was in the deletable part, e.g, `I think I think that ...`, repairs, where the deletable part is replaced by a different phrase but with a similar meaning, e.g., `I believe I think that ...` and false starts, where the deletable part is abandoned altogether in the edit. One of the biggest challenges in disfluency detection is to accurately locate the beginning of the disfluency and the interruption point, that is, to determine which part should be removed in order to recover fluent speech.

Disfluencies can either be simple, if only one of the above types occurs, or complex, if a sequence of the above types occurs. For instance, the edit disfluency `<If a If there are ...uh...>* I think that if there are` contains a sub-disfluency, as well as a filled pause, in its edit part.

A number of methods have been proposed that try to identify and remove disfluencies, with a view to reduce word error rate and to increase the quality and readability of the output. Some of these methods consider only prosody-based features [88]. Other methods use lexical-based features [89]. Some of the proposed classifiers are the Conditional Random Field (CRF), Maximum Entropy, and HMM-based [56]. Hidden Event Language Model (HELM) [93] are also used, encoding disfluencies as events that are not directly observable. Yet other methods [40] consider the disfluency-containing text to be a noisy version of the clean text, and learn a system which translates from the former to the latter. Recently, Integer Linear Programming

(ILP) was used with a view to adapt disfluency detection methods to out-of-domain data [31].

A key advantage of removing disfluencies is that their presence is also detrimental to further downstream processing techniques, such as natural language understanding methods. For instance, disfluencies hamper parsing techniques [29], since sentences are no longer well-formed.

In general, more information can be extracted from the speech signal to enrich the transcription. A few examples are detecting overlapped speech [8], background music [41], or identifying which speaker is talking at a given segment [96]. Some of this information can also be used to further improve speech recognition results by discarding non-speech segments and by adapting the acoustic models to the appropriate speaker or group of speakers.

1.2.2 Machine Translation

Machine Translation, the problem of automatically translating a sentence or text from one language to another, is considerably hard, specially between a pair of languages which are not closely related. One of the most significant challenges faced by MT is that natural language is intrinsically ambiguous, at the lexical, syntactic and semantic levels. Another issue are the differences in the mapping between concepts and structures in different languages.

There is a wide range of approaches to machine translation, concerning the complexity and abstraction level of the representations built by the systems. At one end of the spectrum are interlingua-based approaches [24]. Here, a language independent representation of a sentence's meaning is constructed through several analysis steps, and then a natural language generation engine is used to generate the output in the target language. At the opposite end of the spectrum are direct translation methods where words or phrases are translated directly into the target language without regard for morphological, syntactic or semantic information. In between these two extremes are transfer-based systems, which create representations at intermediate levels of abstraction (at the syntactic or semantic levels, but with some language dependencies). The level of abstraction at which each system is built also impacts the ease

with which new language pairs can be added to the system, which is usually easier for higher-level systems.

Most early systems used rule-based approaches, not only because they seemed to have a stronger foundation in linguistic processes, but also because statistical approaches require parallel corpora, which did not exist then. However, the design of rules and grammars for these systems is expensive, since it requires the work of trained linguists, and it is necessary to obtain a good coverage of language constructs. The availability of large multilingual corpora led to the advent of high quality SMT systems. SMT systems learn how to translate from one language to another from data, usually from corpora of aligned sentence pairs that have been created manually. Here, one can use the noisy-channel paradigm to describe the translation problem. This approach assumes that the sentence in the source language, s, is a noise-distorted version of the sentence in the target language, t, which we seek to recover:

$$t^* = \arg\max_e P(t|s) = \arg\max_e \frac{P(s|t)P(t)}{P(s)} = \arg\max_e P(s|t)P(t) \tag{1.4}$$

In the above equation, $P(e)$ is a language model, usually an n-gram language model, as described in Section 1.2.1, whereas $P(f|e)$ represents the translation model. Finally, the maximization corresponds to the search or decoding step.

Word alignments, which allow us to determine an estimate of the translation probability $P(f|e)$, are normally created by applying a generative model, which describes how to generate a target sentence from a source sentence. Probably the best known word alignment models are the IBM 1-5 models [10]. IBM models of increasingly higher number include distortion models, which explain the different word orders in different languages, as well as fertility models (a word is said to be *fertile* if it tends to map into multiple words in the other language). The usual way to estimate parameters for these models (training) is to use the Expectation-Maximization (EM) algorithm [22], although higher number IBM models must use an approximate E-step, since there is no closed form solution for models 3, 4 and 5. Lower number IBM models are often used to initialize parameters for higher number models.

Phrase Based Statistical Machine Translation (PBSMT) systems [48] incorporate context into the translation model, capitalizing on the fact that the translation of a word is influenced by the surrounding words. In a PBSMT system, heuristics are used to extract plausible phrase pairs from a bidirectional word alignment (two unidirectional word alignments combined). Taken together, along with estimated translation probabilities, these phrase pairs constitute what is known as a phrase table.

A PBSMT decoder has a similar function as an ASR decoder, integrating all the information sources and possibly reordering phrases in order to generate the output in the target language. It scores each hypothesis using features such as phrase length, the translation probabilities from the phrase table, distortion probabilities for reording, the language model probabilities, among others. Pruning techniques are also used to keep the complexity of the search space in check, namely beam pruning or restrictions to reodering to maintain tractability.

Recently, syntactic approaches to SMT try to overcome the syntactic gap between languages with different syntactic structures. Some of these approaches do not actually require parsing the source or target languages, whereas others do. These systems can also be categorized based on whether the translation rules are learnt from an aligned string-to-string corpus [18], tree-to-tree [23] or tree-to-string [30]. In general, these systems tend to be superior when compared with phrase based SMT systems for distant language pairs such as Arabic-English or Chinese-English, or when there is a scarcity of resources that leads to data sparsity problems in PBSMT systems.

1.2.3 System combination in ASR and MT

Researchers in both ASR and MT have observed that combining several systems, with different strenghts and weaknesses, often outperforms the best individual system that it is possible to create. It is possible to either combine several systems of the same type (e.g. several ASR systems) as well as several systems of different types (e.g. an MT system with an ASR system).

When combining multiple ASR systems, system integration can occur at the different processing stages: at the feature extraction level [90], where the outputs from multiple feature extraction methods are concatenated to form a larger vector; at the acoustic model level, where the posterior probabilities from different acoustic models are combined, using a weighting scheme, to yield a potentially more accurate posterior distribution over phones [67]; and at the output level [27], where the outputs of several systems (either in the form of N-best lists, confusion networks or lattices) are aligned and rescored.

In MT, the simplest combination method consists of only rescoring hypotheses without merging them, and selecting the optimal one, according to a set of models [71, 13]. However, in this case, the output of the system for a given sentence will not be better than that of the best system for that sentence.

Consequently, other methods for system combination have been developed that try to select the best parts of each system's output and produce a better sentence than any of the systems individually [28, 4, 42]. One of the most successful methods for this type of combination is confusion network decoding [61], which involves aligning the different hypotheses and searching for the best path in the resulting confusion network. MT system combination is often most effective when applied to different system types, such as syntatic and PBSMT systems, because there is less overlap between the types of errors that they produce.

An important special case of MT system combination is multi-source translation [73, 87]. In this application, multiple versions of the same sentences are combined to produce a single target translation. For example, in the cases in which a single document must be translated into several different languages, existing translations of the document in some languages can be used to help improve translation into other languages.

Broadly, we can divide the combination of ASR and MT systems into two categories:

- sequential combination (Figure 1.2) - which is the most common type of combination, where an MT system is pipelined after an ASR system, and it is used in S2S and Speech-to-Text (S2T) systems.

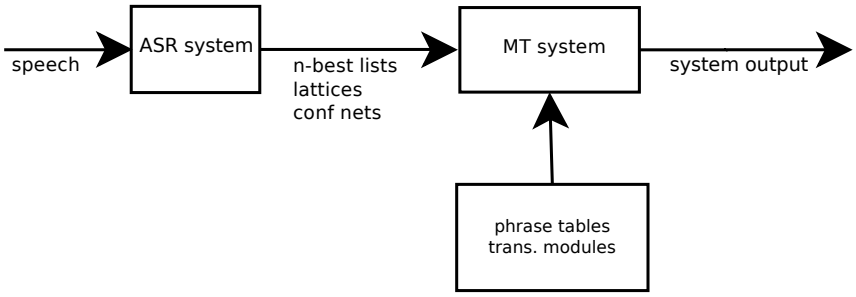

Figure 1.2: Sequential combination of an ASR system with an MT system.

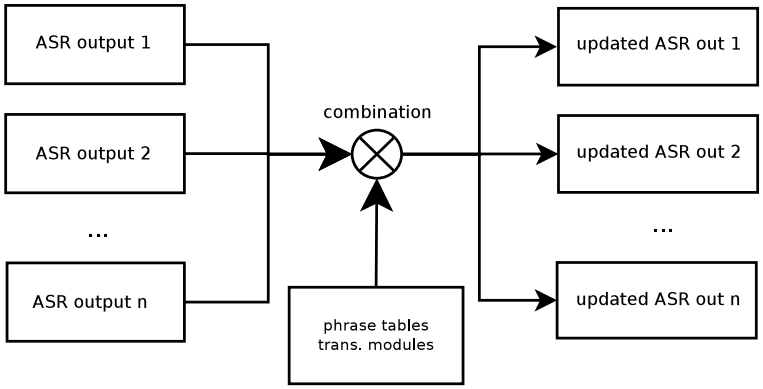

Figure 1.3: Parallel combination of multiple ASR systems. The outputs of different ASR systems (1-best, lattices, etc.) are combined together via phrase tables or translation modules to obtain improved ASR transcripts.

- parallel combination (Figure 1.3) - in this type of combination, a number of ASR systems in different languages, known to contain redundant information, are combined through a translation module or phrase table.

1.2.3.1 Sequential combination

Much work has been developed regarding sequentially combining two systems, mainly for use in a speech translation system. It mostly focuses on integrating ASR and SMT systems as tightly as possible, in order to mitigate their weaknesses.

One of the lines of work is to find a segmentation of the input speech into utterances that are optimized for efficient translation rather than accuracy at the source language level [62]. Another idea is to widen the interface between the ASR and SMT components, which can be done by passing more than just the top recognizer hypothesis - usually lattices [86] or confusion networks [6]. The latter have the advantage of simpler topology leading to simpler and efficient decoding algorithms, but they may also contain paths which are not part of the original lattices. The tightest coupling between ASR and SMT modules can be achieved through WFST composition of the ASR and MT transducers to create a global search space [77]. The idea behind these techniques is that the translation models may be able to recover from recognition errors by implicitly selecting those hypotheses that are the most suitable for translation.

Tighter integration between modules often leads to better translation quality, but it is also more computationally expensive, due to the need to integrate over the different hypotheses that are generated by the ASR system. This sometimes requires approximations for tractability.

Since most MT systems are trained and optimized with written data, techniques have been developed that try to adapt the ASR output to make it more similar to the texts the MT system was trained with. True-casing, punctuation [76], and disfluency removal [81, 102] have all been applied for this purpose. A related approach, which can be used even when

the ASR system is a black box, consists of applying machine translation to the output of the ASR system. This method uses an SMT system trained from a parallel corpus that contains ASR-transcribed sentences, together with the respective manually generated transcriptions.

1.2.3.2 Parallel combination

A number of authors have sought to combine ASR and MT in a parallel fashion. Some of these methods are used to combine speech with redundant information contained in a text stream, usually for an application such as Computer Assisted Translation (CAT) [11, 45, 83]. In CAT, the idea is to have a human translator dictate, rather than type, the translation to a given source document, since the former is usually more efficient. Therefore we transcribe, using a speech recognizer, the target document which is being dictated by the translator, and take advantage of the information contained in the source language text in order to improve our decoding of the translator's speech.

One of the ways proposed to achieve this is to have the ASR system generate an N-best list, and rescore it with a translation model, such as one of the IBM models [11]. An alternative is to use arbitrary word lattices instead of N-best lists for rescoring; however, in this case, the rescoring model must be implemented as a finite-state machine for efficient rescoring. A few works have considered combining multiple ASR streams in different languages [75, 74]. Using speech recognizers in two languages, a pair of speech streams is initially transcribed. Then, the method proposed is to iteratively translate, with an SMT system, the ASR generated transcriptions from one language to another. At each iteration, a new pair of transcriptions is produced, and new phrases are added to the translation model. This is done by applying what the authors call n-gram hints - discounting the cost, in the ASR system, of n-grams that appear in the top translations, and doing the same in relation to the MT system. Large improvements are demonstrated in the Basic Travel Expression Corpus (BTEC) corpus [75], and smaller improvements in the European Parliament data [74].

In Chapter 3, we therefore propose a method that combines the information in multiple languages to yield improved recognition results. In order to connect the language pairs, we

use phrase tables trained for an SMT [58] system, which use, as one of their knowledge sources, a phrase table consisting of pairs of the form (source phrase, target phrase). Using these phrase pairs, it is possible to find potential correspondences between speech in different languages, and to bias the speech recognition models towards these correspondences. We then generalize this technique to sets of arbitrary streams, which can be considered to be approximate pairwise translations of each other.

1.3 Contributions

The main contribution of this thesis is a generic algorithm for combining multiple parallel information streams, which are approximate translations of each other. These information streams can be of different types (e.g., speech, text or slides), and can encode different uncertainty types (e.g. the uncertainty generated by a speech recognizer by virtue of being less than perfect, or that which is present in the different ways of expanding abbreviations in slides). This enables its application to a variety of scenarios in which parallel information from multiple streams can be obtained. The methods developed in this thesis are applied not only to improve speech recognition accuracy, but also to enhance several other speech processing aspects, including the rich transcription of speech. In this thesis, we preferentially used speech as the stream type from which to recover information, but the algorithms presented here can be applied in a straightforward manner to deal with uncertainty in other stream types. This work improves upon some of the previous work on related topics in several ways:

- it does not directly depend on a translation engine decoder, but rather finds phrase pairs that appear both in a phrase table and in lattice pairs. Using lattices instead of N-best lists or 1-best recognizer outputs allows for greater flexibility in the selection of different hypotheses. This allows us to cope effectively with simultaneously interpreted speech, or with lower resource languages, which often lead to reductions in recognition performance.

- no prior alignment between speech segments generated by the recognizers in the different

streams is required or assumed, since one is implicitly constructed during the execution of our algorithm.

- through the tuning of a maximum delay parameter it is, in certain cases, possible to integrate the improvements into a low latency system, by breaking up the input streams into several chunks that can be processed immediately.

- time stamps are used to ensure that the language model scores are not modified for words other than those belonging to phrase pairs in the generated alignment, avoiding errors which could otherwise arise.

- it enables the combination of a mixture of text and speech streams, or other types of information that may be available and representable as lattices over word sequences, therefore generalizing the idea of language combination to arbitrary streams.

- it scales to the combination of an arbitrary number of streams, whenever more than two streams are available.

This work has led to the following publications:

- J. Miranda, J. P. Neto and A. W. Black. "Parallel combination of speech streams for improved ASR", in Proceedings of Interspeech, Portland, USA, September 2012.

- J. Miranda, J. P. Neto and A. W. Black. "Recovery of acronyms, out-of-lattice words and pronunciations from parallel multilingual speech", in Proceedings of SLT, Miami Beach, USA, December 2012.

- J. Miranda, J. P. Neto and A. W. Black. "Improving ASR by integrating lecture audio and slides", in Proceedings of ICASSP, Vancouver, Canada, May 2013.

- J. Miranda, J. P. Neto and A. W. Black. "Improved punctuation recovery through combination of multiple speech streams", in Proceedings of ASRU, Olomouc, Czech Republic, December 2013.

1.4 Document Structure

In this chapter we have presented a review of previous work on the topics of speech recognition, machine translation, and system combination. The remainder of the document is organized as follows.

- Chapter 2 explains the initial work on unsupervised learning of acoustic models using the TED Talks as an application, by using an iterative procedure to recover filled pauses and by correcting other problems with the transcriptions. It also discusses a system that updates speech recognition models using fast human corrections to automatic transcripts, in order to minimize the human effort required to produce high-quality results.

- Chapter 3 focuses on improving speech recognition by combining streams, possibly in different languages, and presents the lattice-phrase table intersection algorithms, as well as the alignment construction techniques that were developed in order to accomplish this. The chapter also describes recovery of acronyms and of words that are not in any of the generated lattices, in order to improve on the results of the baseline algorithm. Two application scenarios are explored: simultaneous interpretation in the European Parliament Committees, and combination of slides with lecture speech to improve automatic speech recognition of lectures.

- Chapter 4 describes improvements that can be obtained in the topic of rich transcription of speech, when one considers multiple parallel information streams. We discuss the improvements that have been obtained in automatic punctuation recovery, as well as disfluency detection in the European Parliament Committee domain, when compared to methods that do not use information in multiple streams.

2 Lightly supervised acoustic model training

2.1 Introduction

The performance of automatic speech recognition systems is closely related to the quality of their training data, since the presence of errors in transcripts will contribute to increase the probability of similar errors in the testing data. Usually, the process of creating acoustic models for automatic speech recognition involves intensive manual labor. Transcribing large amounts of audio data with the appropriate level of detail can, therefore, quickly become prohibitive. Therefore, it would be desirable to be able to leverage other types of training data, including errorful or incomplete transcriptions.

The present chapter is organized as follows. In Section 2.2, we discuss some previous work in the topic of lightly supervised acoustic model training. In Section 2.3, we describe an unsupervised method for iteratively improving imperfect transcriptions, which contain a number of errors, such as insertions, omissions or substitutions of words. Examples of such transcriptions include subtitled television shows or films, which do not contain annotations indicating the locations of disfluencies or other background events. These totally unsupervised methods do not correct all of the errors in the transcriptions, and are not suitable for applications in which no initial transcription is available. Therefore, in Section 2.4, we discuss a semi-supervised method which uses crowd-sourced user corrections of speech transcriptions. This technique minimizes user effort by combining an optimized interface for efficient option selection with user feedback to update the speech recognition models. This is used to complement the method described in Section 2.3. In Section 2.5, we draw conclusions and lay out future work.

2.2 Related work

The need to avoid the detailed manual transcription of large amounts of speech data has led to a considerable amount of research into techniques that take advantage of imperfect transcriptions, which are less accurate but also produced at a lower cost. In [52], a lightly supervised training technique was developed where the output produced by a speech recognition system is compared to closed captions (which are simplified text versions of speech in a television presentation), which contain errors, in order to identify regions that agree. These are then selected for training an initial model, which is then used to iterate this process, hopefully enlarging the training set at each step (i.e each successive model is potentially more accurate than the previous one, so it is expected that a larger proportion of the ASR output agrees with the manual transcriptions). However, simply discarding segments where the audio does not agree with the transcription corresponds to removing the sentences that are the hardest to recognize with the existing models, leading to a limited potential improvement in recognition performance. In [15], this problem is addressed by using confusion networks to compactly represent a set of plausible alternatives. These, and not just the best hypothesis, are compared against the closed captions, and a sentence is accepted if any of the generated alternatives match. In [78], closed captions are interpreted as distortions of the correct transcriptions in the context of the noisy-channel model, and a probabilistic translation model is designed in order to attempt to recover them. In [35], a method was proposed to automatically correct human generated transcripts by using "pseudo-forced alignments", which is defined by the author as being a relaxed version of forced aligment, where a number of editions, encoded as FSTs, are allowed. A related approach was suggested in [98], where missing disfluencies and words are corrected using lattices constructed for that purpose. In [100], speech recognizers in multiple languages combined with a multilingual confidence measure are used to select data for training an acoustic model for a different language, in a completely unsupervised manner.

Another class of approaches to obtain speech transcriptions considers the use of crowdsourcing techniques [26], which can be defined as the collaboration of multiple non-experts to solve a task. This can be perfomed through general purpose platforms such as Mechanical Turk

[12], or Crowd Flower[1]. There are a number of challenges that need to be addressed when using crowdsourcing for speech transcription, in particular in what concerns the quality of the result.

Some of the proposed techniques consist of finding a consensus among the task solutions generated by several different users. In [1], both supervised and unsupervised methods to determine the reliability of the crowdsourced transcription are investigated. In [2], a similar approach is utilized to deal with noisy speech transcriptions. Both papers employ Recognizer Output Voting Error Reduction (ROVER) [27] to combine multiple transcriptions which were obtained using the Mechanical Turk platform. This allows the system to be robust to errors introduced by the human transcribers, as long as the majority of the users agree on the correct solution, but it has the drawback of requiring that multiple versions of the same speech transcript be produced.

An alternative are post-editing approaches to crowdsourcing, which consist of automatically generating an initial solution to a problem, which is then corrected by human annotators. These have been applied to different tasks, ranging from MT [7, 53] to ASR [99, 101] as well as Optical Character Recognition (OCR) [69], among other tasks. The work of Section 2.4 fits into this general framework: our focus is on reducing user intervention in the process, which in our system is limited to post-editing automatically generated transcriptions. For that reason, we employ a custom crowdsourcing platform with a small set of high quality editors and we only produce a single transcription for each utterance, which precludes the use of ROVER. Instead, user effort is minimized by the use of both acoustic and language model adaptation, rather than the fusion of multiple hypotheses.

[1] http://www.crowdflower.com/

2.3 Acoustic model generation from imperfect transcripts

2.3.1 Proposed Method

We propose an iterative technique for automatically correcting speech transcripts. The general control flow of this algorithm is depicted in Figure 2.1.

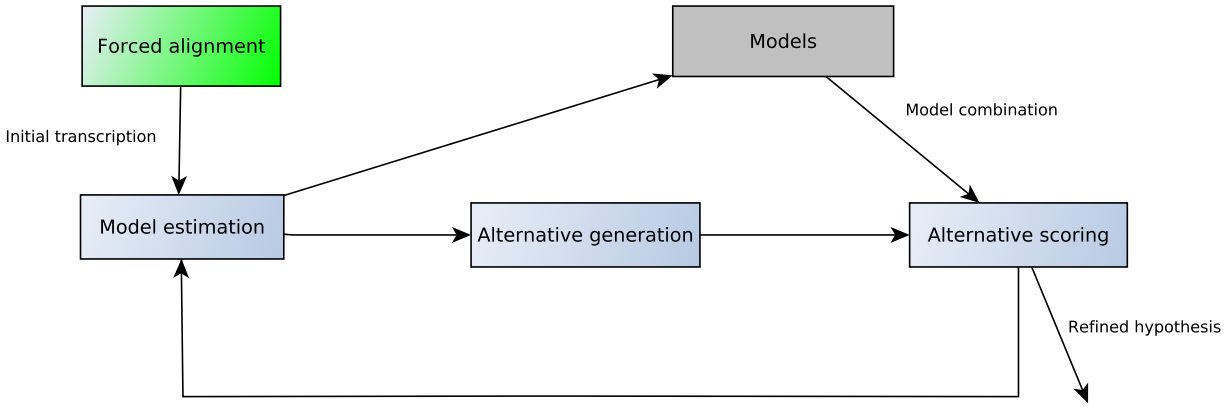

Figure 2.1: Flow diagram illustrating the transcription improvement algorithm.

The algorithm is initialized by force-aligning the subtitles and the audio, using a baseline acoustic model. At its core, it consists of a number of iterations (we used a fixed number of iterations as the stopping criterion, but others, such as the number of sentences that change between iterations, are equally valid). At the beginning of each iteration, we estimate a set of models using the transcriptions obtained in the previous step. We then generate a set of alternatives for each sentence in the transcription. These alternatives are then rescored using the models generated at the beginning of the iteration as features. Our method may be seen as similar to the well-known EM algorithm [22], in that it alternates between estimating model parameters and transcriptions. However, our algorithm does not integrate over the space of all possible transcriptions; rather, it only considers the best transcription when re-estimating model parameters. There is also no guarantee of convergence to a local maximum.

For computational convenience, we further subdivide each iteration into a sequence of steps. In each step, only modifications in one of the classes described in Figure 2.2 are made to each of the sentences in the transcription. These classes were designed to cover the most frequent errors found in the transcriptions, which we would like to fix. So, in each step, we generate a set of alternative transcriptions for each sentence, which are force aligned to the corresponding speech, and this alignment is used to compute each of the individual model scores. We compute an overall score for a sentence in the transcript by linearly combining the individual model scores, with the weights for each model assigned through manual tuning. Finally, for each sentence, we select the transcription with the best overall score.

In this work, we used the following models for rescoring:

- N-gram language models: we used 5-gram models to predict the probability of a given sentence, including both filler and non-filler words. However, these models were not trained directly on the transcriptions. Instead, we tied all the non-filler words together except for those high-frequency words, to try to prevent the model from learning peculiarities of the transcription, instead of important properties of the filler distribution. The idea is to learn what words are usually followed or preceded by fillers.

- Rule-based models: these apply a number of hand crafted rules that reflect prior human knowledge about the filler distribution in sentences. For instance, long alternating sequences of fillers and non-fillers are considered unlikely and thus penalized by this model.

- Phone duration models: we modeled the phone durations as Gaussian, conditioned on the identity of the current and last two phones [50]; for phone combinations for which not enough data is available, however, we condition only on the current and last phones.

- Acoustic model: this corresponds to the score output by the force aligner for the baseline acoustic model used.

At the beginning of each iteration of the algorithm, these models are reestimated from the

Class I - Filler swaps

- In this class, we allow swapping the position of a filled pause and the word that immediately precedes or follows it. Often, the force aligner incorrectly places the filler before the word or vice-versa, usually in the case of small words that are acoustically confusable with filled pauses.

Class II - Filler insertion

- sometimes, silences, breathing noises, and filled pauses that are relatively short are ignored by the force aligner. A number of potential locations for their insertion are identified by using confidence scores (words with low alignment scores or with extra long initial or final phones are good candidates for having absorbed one of these noises into their pronunciation).

Class III - Garbage word insertion

- Repetitions and a number of filler words or expressions ("so", "you know") are often overlooked in fast transcriptions of speech. In this class, we create an alternative hypothesis which consists of replacing such phrases with a garbage filler, which avoids them being merged with adjacent words in the transcription during acoustic model training.

Class IV - Filler deletion

- One of the problems of the force aligner is that it tends to insert a large number of spurious noise and garbage fillers, usually with short durations. This is because the acoustic models for these usually have very large variance and so will often match regular words as well. Here we generate alternatives to remove short duration fillers.

Class V - Word deletion

- In a small number of cases, the human transcriber adds words that were not in the original speech, in order to make the transcription easier to understand. This creates a mismatch between the acoustic signal and the transcription. In order to address this issue, we allow very low confidence words to be deleted.

Figure 2.2: Classes of rules for sentence alternative generation

currently-best transcriptions using Maximum Likelihood estimates, except the rule-based model and the acoustic model, which remain unchanged throughout the process. The SRILM toolkit was used for language model reestimation.

2.3.2 Experimental Setup

2.3.2.1 Dataset

We selected the TED talks as a dataset for experimenting with our algorithm. The TED talks concentrate on topics as diverse as entertainment, science, design or politics, and are in the English language (although some satellite events have talks in other languages). The speakers are mostly fluent, although some of them are not native, and a wide variety of accents is represented. The talks are subtitled in English, as well as translated to over 40 different languages (both these tasks are performed by volunteers, so availability may vary). The main purpose of the subtitles is to convey the meaning of what is being said by the speaker; in this sense, disfluencies such as filled pauses or other non-speech noises are not annotated (although some events like laughter and applause occasionally are). Also, it is not uncommon for the subtitles to lack words or invert their order. This makes the dataset ideal for the application of our algorithm. In total, 525 talks, with a duration of 180 hours, were collected from http://www.ted.com. They were pre-processed by first manually removing talks containing very little speech (such as musical performances). Talks containing background music or other noises were not discarded. We randomly selected 12 of the talks for our held-out validation set, and 31 of the remaining talks as our test set. One speaker, Al Gore, has talks in both sets; all the other speakers are represented only either in the training or the testing set. Since the subtitles in our test set were also not manually annotated, they contain the same types of errors as those found in our training set. In this way, the word error rates that we measured are artificially high. However, we expect the effects of this distortion not to impact our model comparisons significantly, since the word error rate of all the models will be similarly affected.

	Complete	Half 1	Half 2
Iteration 3	35.2%	39.4%	39.5%
Iteration 4	34.9%	39.0%	39.3%
Iteration 5	36.0%	39.9%	40.1%
ROVER	33.8%	37.6%	37.8%

Table 2.1: N-best rover results (as well as results for 3 different iterations) for the whole dataset and each of the two halves

2.3.2.2 Corpus preprocessing

In order to have appropriate segments for training and testing, it is desirable to segment the audio of each talk into chunks of reasonable duration (about 30 seconds), since the timings provided with the subtitles are only approximations which are unusable for segmentation purposes. We first break the audio in the locations of silent pauses with durations greater than 0.50s. We then decode these chunks using an acoustic model trained on the Hub4 data [32], and a biased language model, estimated using the transcriptions, and use the words for which both the decoder and the subtitles agree, as the locations where we split the audio into segments. To prepare our data for training, it was also necessary to expand numbers into sequences of words. There is often more than one way to do this, depending on speaker and context. Our approach consisted in building a FST for the transcriptions containing the various alternative ways of spelling a number, and another FST encoding the decoder output (obtained while segmenting the data), and then finding the best path through the transducer resulting from the composition of the two.

2.3.3 Results

All the acoustic models were trained using SphinxTrain, using 4,000 senones and 16 Gaussians per density. All the recognition experiments were carried out using the Sphinx3 decoder [79]. The language model used for the recognition experiments reported below was built by interpolating the subtitles with a Gigaword trigram language model, using the held-out validation set to optimize interpolation weights. The baseline acoustic model used for the sentence rescoring described in Section 2.3.1 was trained using the existing transcriptions

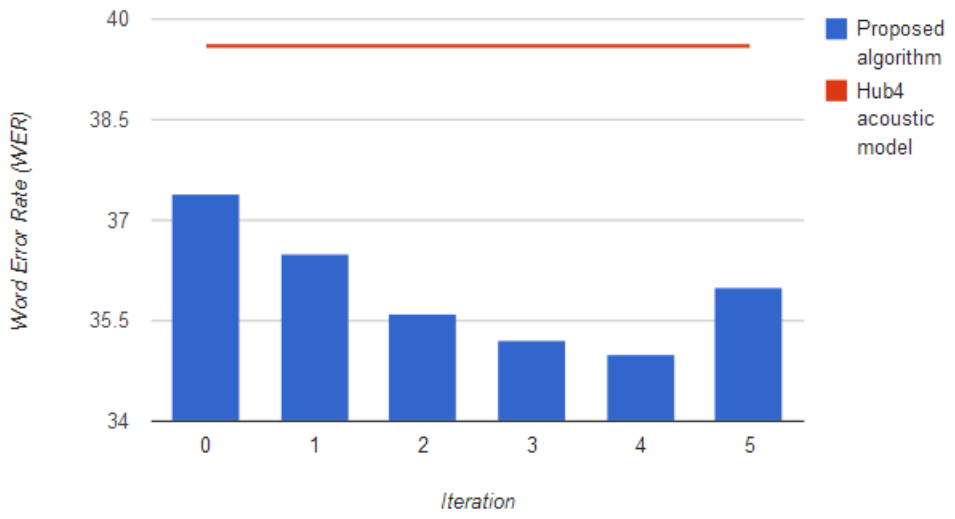

Figure 2.3: Error rate on the test set for acoustic models trained on each iteration of the transcriptions, compared with the WER achieved by the Hub4 model.

(i.e, at iteration 0).

Figure 2.3 shows the WER of each of the acoustic models, measured after the end of each iteration, compared with the WER of the Hub4 acoustic model measured on the same testing set. At iteration 0, we use the acoustic model trained directly on the force-aligned transcriptions; in subsequent iterations, we use the transcription output by the algorithm at that iteration to train the acoustic model for recognition. We found that the WER increases again after reaching a minimum at the end of the fourth iteration. We hypothesize this may be because our algorithm is introducing a number of errors in the transcriptions as well as correcting them, and after the fourth iteration the errors introduced outweigh the corrections, thus increasing the WER.

We observed that the types of recognition errors made by each of three acoustic models - corresponding to the third, fourth and fifth iterations respectively - were considerably different. This motivated combining the outputs of the different decoding runs by using the N-best ROVER [94] implementation in the SRILM toolkit, and observed a 1.1% absolute improvement in WER. To test the reproducibility of this result, we randomly split the training

data into two halves of approximately equal size, and repeated the process on each of the halves, having obtained improvements of 1.4% and 1.5% when compared to the best result in each of the two halves. The results are summarized in Table 2.1. The transcriptions for each iteration have different errors, since they represent different stages of the transcription correction process. So acoustic models built from transcriptions corresponding to different iterations of the process will tend to make different errors, owing to the different errors in the transcriptions they were trained from, and therefore benefit from system combination.

2.4 Semi-supervised human acquistion of transcriptions

In this Section, we explore a different path for obtaining speech transcriptions, which can be applied without the need for an existing transcription. We developed a post-editing system, where an initial transcription is generated by an automatic speech recognizer, and then corrected by users, with minimal human intervention in the process.

Our system is divided into essentially two parts: a front-end or interface where users can correct the output generated by an automatic speech recognizer, together with a back-end which propagates user changes to sections not yet corrected by users. This improves the recognizer output for the remaining sentences, therefore reducing the human workload that is required to correct the transcripts.

The interface allows users to choose among a small number of alternatives to that which the ASR system has considered to be the most likely hypothesis. The user also has the possibility of manually entering a different option from the ones that are displayed, if he or she finds that none of those is correct. Preferably, the alternatives that are generated should minimize the need for manual insertion of words, since this is more time-consuming than simply clicking on an alternative, and our goal is to minimize the overall time spent by human editors.

2.4.1 Confusion network generation

Recognition lattices cannot be easily displayed to users, since these are non-linear and usually have far too many redundant paths to be presented. Therefore, we begin by converting the ASR lattices to confusion networks, which are graphs in which the only arcs are between two consecutive nodes.

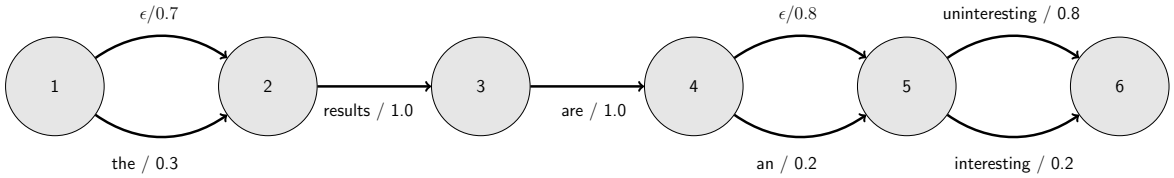

Figure 2.4: Example of a confusion network with two ϵ-dominated slots

The traditional method to generate these confusion networks, however, creates a significant number of "ϵ-dominated slots" : those for which either the most likely or the correct hypothesis is the absence of any word, as can be seen in Figure 2.4. The main reason for this is that the confusion network has to encode many competing hypotheses of different lengths. Such slots, however, are cumbersome because they tend to distract the user from the sentence being corrected, and may require a larger number of steps to correct a given utterance.

We propose an algorithm which aims to minimize the number of slots that are generated for a given utterance, by merging together words in contiguous slots to create phrases.

Our method for generating compact option lists from confusion networks consists of the following steps:

- We identify which slots are to be merged into one. This is done greedily, by selecting sequences of at most four slots, in which at least one of those slots is not epsilon-dominated, i.e. the highest probability arc does not have an epsilon label.
- We generate a candidate list of phrases by considering all possible combinations of words in the slots that are to be merged. For each of the candidate phrases, we compute its posterior probability according to the recognition word lattice.

- The option list consists of the selected phrases, sorted in decreasing order of their posterior probabilities.
- If the resulting list has a larger number of options than a predefined threshold, the phrases beyond that threshold are discarded, in order to keep the list size manageable.

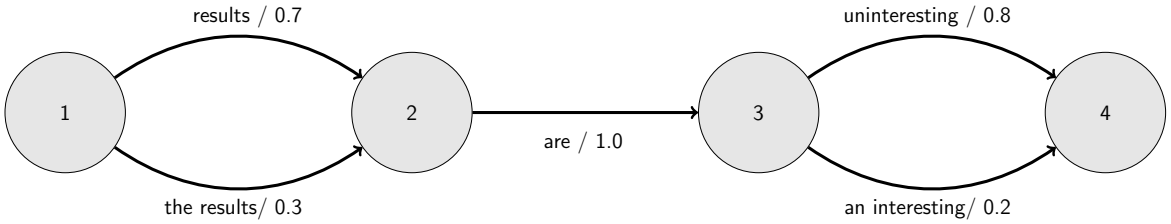

Figure 2.5: Example of a compact confusion network with no ϵ-dominated slots

The result of applying this method to the confusion network of Figure 2.4 can be seen in Figure 2.5.

2.4.2 Model adaptation

In order to reduce unnecessary human intervention in the transcription process, we update the parameters of each of the three main ASR system components: the acoustic model, the language model and the lexicon. We use the corrected sentences returned by users to adapt each of these components.

2.4.2.1 Acoustic model adaptation

It is possible to use the corrected sentences that were returned from users to adapt the recognizer's acoustic models and improve accuracy for the speaker's remaining utterances. Of course, since the humans are not guaranteed to produce optimal transcriptions, there may still be errors in the corrected sentences. To try to mitigate the errors that this would potentially introduce into the acoustic model adaptation process, the regions with a confidence measure value (as given by the recognizer) below a given threshold are skipped during acoustic model adaptation.

We used Stochastic Gradient Descent (SGD) [84] to adapt our neural network based acoustic models.

The targets for adapting the models were generated by force aligning the corrected sentences with the audio. To avoid overfitting on the adaptation data, we kept 10% of it to use as a validation set, and selected the best performing model on this set, over 50 training iterations.

2.4.2.2 Language model adaptation

We also adapt the language model using user-corrected segments, in order to improve recognition in the remaining uncorrected utterances. In this context, we desire to be able to work with small amounts of data (as small as a few utterances) and, as a result, methods such as simple linear interpolation are not very effective in improving recognition accuracy.

Instead, we explore the use of a cache-based language model which assigns a larger weight to n-grams that occur in human-corrected utterances.

The idea behind the use of such a language model is to capture speaker and topic specific expressions, and reinforce their probability of occurrence without affecting the probabilities of other n-grams significantly. The conditional probability of a word under this model is given by Equation 2.1:

$$P(w_i|w_{i-n+1}...w_{i-1}) = \alpha P_{ng}(w_i|w_{i-n+1}...w_{i-1}) + (1-\alpha)P_c(w_i|w_{i-n+1}...w_{i-1}) \quad (2.1)$$

$$P_c(w_i|w_{i-n+1}...w_{i-1}) = \frac{\sum_{j \in occur(w_{i-n+1}...w_i)} d(j,i)}{\sum_{k \in occur(w_{i-n+1}...w_{i-1})} d(k,i)} \quad (2.2)$$

In Equation 2.1, P_{ng} represents the background n-gram probability, whereas in Equation 2.2, $d(i,j) = a^{-|i-j|}$ represents the distance function between two word positions i and j, and $occur(n)$ is the set of positions where n-gram n occurred. The term d gives emphasis to occurrences of an n-gram closer to the current word position.

The language model defined by Equations 2.1 and 2.2 depends on the position of the current word, and therefore cannot be used directly in our WFST decoder. We approximate it with a set of n-gram models, one for each sentence, which can be used efficiently in our recognizer. In order to do this, each n-gram probability is kept fixed within each sentence. This is achieved by setting the position i, of each n-gram, in Equation 2.2 to the beginning of that sentence.

This language model is then used when rescoring uncorrected human sentences, in place of the background n-gram language model.

2.4.2.3 Pronunciation learning

Since the words that can be input by users are unrestricted, they can, in particular, be out-of-vocabulary (OOV) words, for which we would like to learn the pronunciations. Pronunciation learning enables the ASR system to recognize further instances of a previously OOV word, thus eliminating the need for the user to correct the same word repeatedly. It is also applied to other words which we suspect have missing or incorrect pronunciations, in particular those words which have been added manually by users, without being part of any of the system generated options.

We use a modified version of our recognizer to automatically learn pronunciations for these words. During alignment, the decoding graph is constructed with an expanded lexicon, which consists of the lexicon for the original words, merged (with a union operation) with transducers representing the pronunciations of the words that we wish to learn.

The idea of the language specific transducers is to encode the phonotactic constraints of the language and therefore improve the quality of the learned phone sequences by ruling out implausible pronunciations. The transducers are constructed by viewing Grapheme-to-Phoneme (G2P) as a machine translation task, in which we map from the set of words to the set of pronunciations, and enumerating possible translations for each letter or letter sequence that compose a given word. These phone sequences are those that a given letter sequence is usually mapped to in the pre-existing lexicon.

When the force aligner is run, the pronunciation which is learned for a word corresponds to the path chosen by the decoder in the augmented lexicon transducer, whenever that word occurs in the alignment reference text. If the word occurs more than once in the alignment, the most common pronunciation found for that word is kept.

2.4.3 Results

2.4.3.1 Data set

We experimented with a data set of 20 TED talks, randomly drawn from the TED data set, which was described in Section 2.3.2.1.

Of these, two were set aside as a held-out validation set, and eighteen were used for testing. The talks were uploaded to the Unbabel system [97], which was modified for the purpose of correcting speech transcripts, and they were corrected by users.

The recognizer used for these experiments was Audimus [63], a hybrid WFST-based recognizer. We generate multiple feature streams - Relative Spectral Transform (RASTA) [38], PLP [37] and Modulation-filtered Spectrogram (MSG) [46], from the input audio, which are combined at the MLP posterior level. We used our existing English acoustic models and lexica [65]. The language model was trained using the SRILM toolkit [92] using the TED talk subtitles, excluding those of the selected data set, interpolated with the Gigaword language model.

2.4.3.2 Experiments

We designed and executed three different sets of experiments. First, we sought to analyze to what extent the human correction of an increasing fraction of the talks led to an improvement in recognition performance in the uncorrected portion, through the application of the previously described adaptation techniques. The second set of experiments was designed to assess how close the human corrections are from the best possible corrections, in terms of the im-

provements that can be obtained through model adaptation. In the third set of experiments, we looked into the impact of pronunciation learning in recognition accuracy in the remaining uncorrected data.

For the proposed experiments, we considered three different types of adaptation: language model adaptation (Section 2.4.2.2), acoustic model adaptation (Section 2.4.2.1) and both applied simultaneously. Besides these adaptation types, we tested how learning new pronunciations would influence our results (Section 2.4.2.3).

We also considered three different correction types: human corrected transcriptions, which are obtained directly from the users, reference oracle transcriptions, which correspond to the gold-standard transcriptions, and lattice oracle transcriptions, which are intended to measure the extent to which users can improve the automatically generated transcriptions if they could not manually insert new words.

Lattice oracle transcriptions are created by selecting the slots in the generated confusion networks in a way that minimizes the word error rate between them and the reference transcriptions.

For the first set of experiments, we simulated the scenario where users had only corrected a random sample of the utterances in each talk. We varied the sample size, as a percentage of the talk size, from 10 up to 60% of the total number of utterances, and selected the utterance sets such that the smaller samples are proper subsets of the larger samples. Additionally, we tested each of the different adaptation types (LM, Acoustic Model (AM) and both). The results can be seen in Figure 2.6.

We can observe from the graph that acoustic model adaptation seems to outperform language model adaptation, across all tested sample sizes. Similarly, combining both acoustic and language model adaptation is superior to acoustic model adaptation, but by a small margin. A second observation is that the absolute WER improvement varies between 2.5% and 3.6%. The overall improvement in accuracy is largest in samples corresponding to 20% and 30% of the talk's size. The reason for this appears to be that for larger sample sizes, the number of

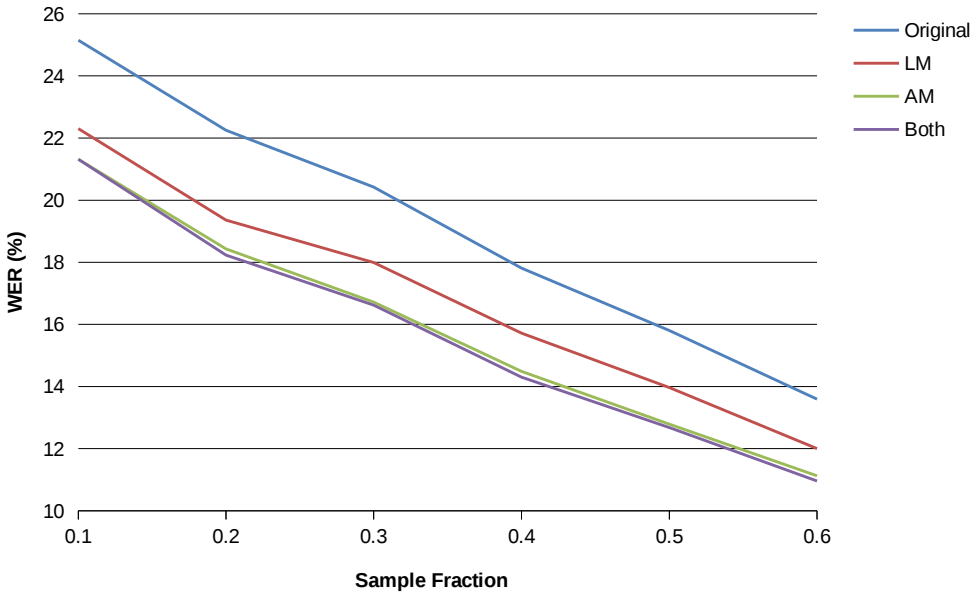

Figure 2.6: Results (% WER) for the adaptation of the acoustic and language models using transcription samples of increasing sizes, as a fraction of the total talk size. The *original* line represents the WER when replacing the sentences in the sample with the reference, without performing any adaptation.

remaining errors that can be corrected by these techniques is also smaller, while for smaller sample sizes there is usually not as much data available, so the proposed adaptation methods are not as effective.

From another perspective, it can be seen in Figure 2.6 that correcting 30% of the data without using any type of adaptation, is roughly equivalent to correcting 15% of the data using both adaptation types. In other words, using adaptation eliminates the need to correct 15% of the data, while still achieving the same recognition accuracy.

For the second set of experiments, we compared the adaptation impact of human corrected transcriptions with that of lattice oracle transcriptions and reference oracle transcriptions, using a fixed sample size corresponding to 40% of the talk size.

The results from the second experiment can be seen in Table 2.2:

It is possible to see from Table 2.2 that human corrected transcriptions were not very far from

Type	Reference	CN oracle	Human
ASR (baseline)	28.55	28.55	28.55
40% corrected	17.23	22.12	17.81
+AM adapted	13.36	18.99	14.49
+LM adapted	14.68	19.93	15.72
+Both adapted	13.20	18.91	14.30

Table 2.2: Results (% WER) for system adaptation when using the reference transcription, the user selections and the confusion network oracle. The first row indicates the baseline ASR WER. The second row indicates the WER after replacing the corrected sentences with the appropriate type of feedback. On the last three rows, we display the results of additionally performing acoustic model adaptation, language model adaptation or both, and then recognizing with the adapted models.

the reference transcriptions. Indeed, less than 1% of the errors are left uncorrected in the human transcriptions, when compared to the reference, which is our gold standard. Also, a similar number of errors in the remaining utterances is corrected using both the reference and the human corrected transcriptions as input data for our adaptation methods. This seems to indicate that the additional errors present in the human transcriptions do not adversely affect the effectiveness of our adaptation techniques. By analyzing the results for the confusion network oracle presented in Table 2.2, we note that roughly half of the errors in this sample could be corrected by selecting options from the generated confusion networks. This implies that correcting the remaining errors required the manual insertion of additional words, and suggests that there may still be room for improvement in terms of how the options which are presented to the user are generated.

Finally, we used a fixed sample size corresponding to 45% of the talk size to test the effectiveness of our pronunciation learning techniques. The results are displayed in Table 2.3.

Type	AM adapt	LM adapt	Both adapt
ASR (baseline)	28.55	28.55	28.55
PL	14.08	15.07	13.74
No PL	14.19	15.13	13.95

Table 2.3: Results (% WER) for pronunciation learning (PL), for a human-corrected sample with 45% of the talk's utterances. The first row indicates the baseline ASR WER. In the second row pronunciation learning is used, whereas in the third row it is not. The columns represent the different types of adaptation used (acoustic model, language model and both simultaneously).

We observe a small improvement in recognition accuracy, in the condition where new pronunciations are learned. This improvement is seen across the different adaptation types used, but is slightly larger when acoustic model adaptation is used. We hypothesize that this may be because the new pronunciations learned lead to more precise alignments, which are then used as a basis for acoustic model adaptation. Overall, the improvements in recognition accuracy are limited, because the words for which pronunciations are learned tend to not re-occur many times in the remaining uncorrected utterances.

2.5 Conclusions and Future Work

We presented a technique with the aim of improving fast human transcriptions of audio for acoustic models. Our technique consisted of force-aligning the transcriptions using existing acoustic models, and then exploring alternatives to repair errors in these alignments. The most promising resulting transcriptions are then selected, and the process is repeated. Our algorithm demonstrated an absolute improvement in WER of 2.5%, when compared to using the initial force-aligned subtitles for training. By combining the decoder output for the three best acoustic models we had generated, using n-best ROVER, we were able to improve WER by an additional 1.1%. Possible future work on this topic includes exploring different information sources for rescoring alternatives, as well as different model combination functions, and alternative ways of decoder output combination.

We also presented a complementary technique which tries to minimize user correction effort by first dividing sentences into slots and providing a set of alternate options for each slot, and then updating the recognition models in order to improve recognition accuracy for the remaining, uncorrected sentences. We experimented with both different adaptation types (acoustic model adaptation, lexicon model adaptation or both) and sample sizes. We showed that it is possible to improve recognition accuracy (between 2.5% and 3.6% absolute) across different sample sizes ranging from 10% to 60% of the size of the talk being corrected, and that both acoustic model and language model adaptation are beneficial.

In the experiments presented here, we selected a random subset of each talk's utterances for adaptation. However, there is no guarantee that this is the optimal strategy. In future work, we would like to experiment with different sample selection techniques. In fact, it is possible that, by selecting sentences which are in some way representative of the talk's content, while at the same time not too difficult to correct, we could improve performance over the random sampling strategy described in this chapter.

3 Multistream Combination

3.1 Introduction

There are a number of applications where we have access to multiple, parallel streams, carrying the same or a closely related information content. In the European Parliament, as well as in other multilingual or supranational bodies such as the United Nations, interpreters translate a speech into multiple languages. In Computer Assisted Translation (CAT), a translator dictates the translation for a given text, instead of typing it. Since speech is more natural than text input, this reduces the overall translation effort and, consequently, also translation cost. CAT techniques therefore take advantage of the two parallel streams that are available (the speech and the original text) in order to produce a more accurate transcription of the translator's speech. In respeaking applications [51], whether offline or online, spontaneous speech is repeated by a different person under controlled conditions, thus creating two parallel streams. This is done to improve the accuracy of transcripts produced by speech recognizers, but the information present in the original speech is usually discarded.

Naturally, it is possible to generate automatic transcripts for those speeches using a standalone speech recognizer, but due to the challenge presented by interpreter speech or low-resourced languages, we would like to find a way to use the additional information that is available. In this chapter, we propose a method that takes advantage of the redundant information present in parallel, partially or totally redundant streams that may exist, in order to improve the quality of the automatic transcripts of these speeches. Our method is generic, allowing for encoding the variability in different stream types as probability distributions over word sequences, and it is extensible, for it allows an arbitrary number of streams to be combined together.

We explore how our method can be applied to two tasks: simultaneous interpretation in the European Parliament, and lectures supported by slides. The developed system can trivially be extended to other applications, such as combining the lattice outputs of multiple recognizers of the same speech. It can also easily be applied to situations where multiple copies of the same speech input are available [55, 91], spoken by either the same or different speakers, of which respeaking is a particular case.

The remainder of the current chapter covers the work done in the scope of multistream combination. We begin by describing the overall system architecture, which is used in all of the multistream combination applications described. We then identify a number of problems with our original algorithm, namely the lack of certain words in the recognition lattices, that led to the development of a number of techniques to mitigate this problem: acronym, out-of-lattice (OOL) word and pronunciation recovery. Finally, we demonstrate that text streams can also be used as an information stream by combining speech and slides to obtain improved recognition performance.

3.2 System architecture

The overall system architecture can be seen in Figure 3.1. Our baseline system for combining multiple streams consists of the following steps :

- Depending on the stream type, which may be speech, text, or other information source, collect or generate phrase tables for each combination of streams (not all combinations of phrase tables need to be generated for this to work). This step is usually performed offline, before the actual combination takes place, and only needs to be done once for each system setup.
- Generate lattices for each stream, which represent a set of alternative hypotheses for that stream. Again, the manner in which lattices are generated depends on the type of stream. For a speech stream, for instance, lattices can be generated by running that stream through an appropriate ASR system.

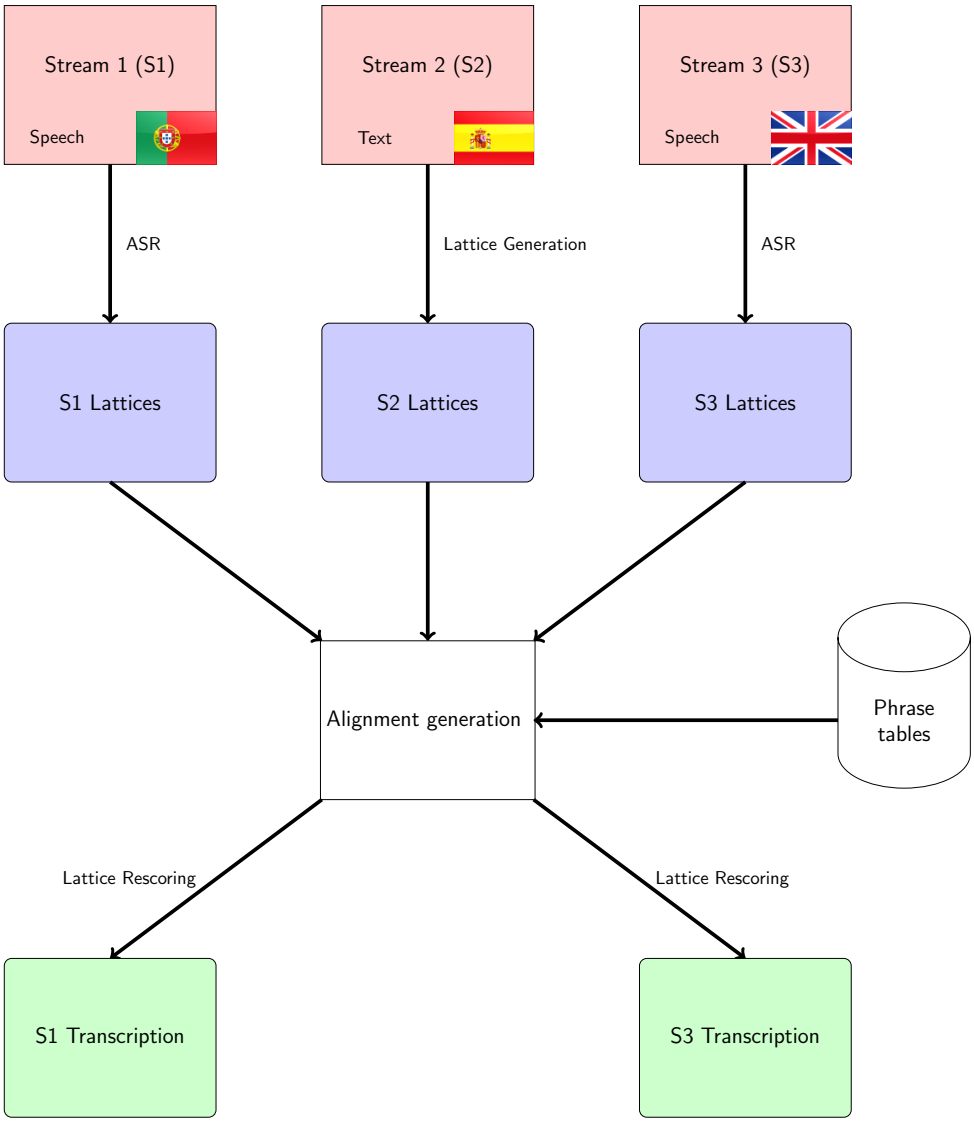

Figure 3.1: The proposed system architecture for combining three parallel streams (two speech streams and one text stream).

- From the generated lattices, obtain posterior n-gram distributions, in order to estimate the probability of each word sequence compared to the other word sequences in the lattice.

- For each pairwise combination of two streams (*stream pair*), intersect the lattices for each of these streams with the phrase table that maps one of the streams to the other. This generates a set of phrase pairs that are common to both the lattices and the phrase table.

- Score the obtained phrase pairs, and select a subset of these that is consistent (i.e. does not have conflicts among phrase pairs), therefore creating an alignment across streams.

- The alignment thus created is then used to rescore the lattices. This consists of finding new optimal paths through these, which may have changed compared to when they were generated due to the new information that is available in the alignment. This is done for all streams containing speech, leading to a new, hopefully more accurate transcription for these streams. Note that in Figure 3.1, this step is not performed for stream 2, since it is a text stream.

Figure 3.2 depicts a simplified version of this method, for a concrete example using two streams.

3.2.1 Lattice and phrase table generation

Lattices are directed acyclic graphs which can encode an exponential number of different paths using polynomial space. This compactness makes them suitable for our algorithm, in which they are used to represent a posterior probability distribution over word sequences induced by an underlying stream. In such a graph, exemplified in Figure 3.3, the words are edge labels and the weights correspond to the scores - usually separated into acoustic scores and language model scores - assigned to each word, and each node is associated with a time stamp. There is also an initial and a final state. Lattices can be generated in different ways, depending on

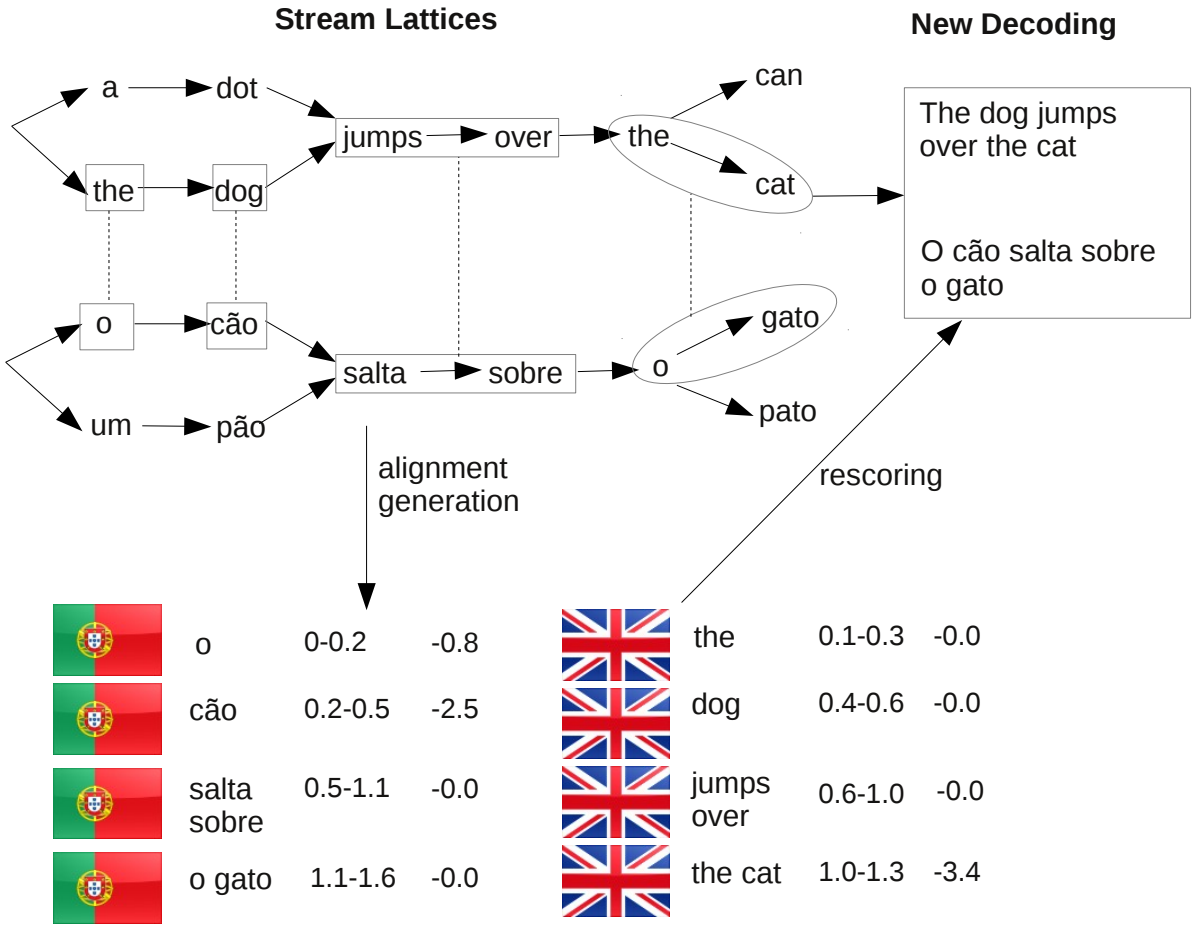

Figure 3.2: An example of the system for two streams, one in English and the other in Portuguese. An ASR system generated lattices for each of the two streams; words or phrases that are translations extracted from the phrase table are connected with dotted lines. These are used to create an alignment of phrase pairs, shown below, which is used to rescore the lattices and obtain a new decoding for each of the streams. In the alignment, the middle column represents the time stamps, and the right column represents the posterior probability for the extracted phrase pairs.

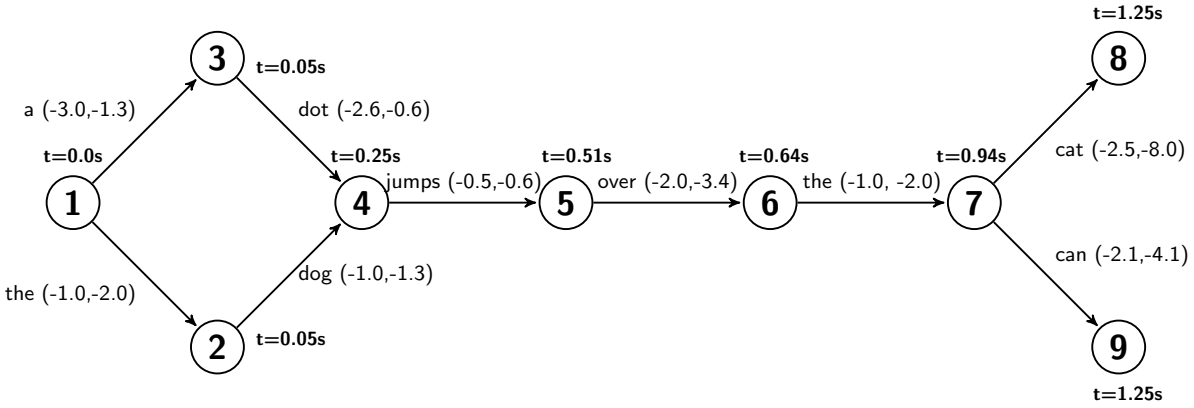

Figure 3.3: A possible lattice or word graph generated by a speech recognizer for the sentence "The dog jumps over the cat". The numbers inside parentheses represent, respectively, the language model and acoustic scores, in the log domain, for the word in that edge.

the type of the stream that they originate from. If it is a speech stream, then these lattices are usually output by a speech recognizer - while decoding the speech, the recognizer will keep track of the multiple possible ways of reaching a given partial hypothesis, rather than only the best one. If, instead, it is a text stream, a possible way of generating the lattices would be to create a "linear path" representing them, or in the event that there are multiple ways of tokenizing or normalizing a given sentence, one could build a lattice encoding those alternatives. Of course, a number of text streams do not have any associated time stamps (subtitles being an exception), meaning that these will need to be determined at a later stage of the combination process.

One of the most significant pieces of information that can be extracted from the lattices are n-gram posteriors. They provide information about the relative importance of different word sequences, and are used as features in subsequent steps of our method. Consider all the lattice paths from the initial to the final state. If we take all those that contain a particular n-gram, sum their scores and divide by the sum of the scores of all lattice paths, we obtain the posterior. Of course, since there can be an exponential number of paths in a lattice, it is not practical to sum over all possible paths directly. Instead, we apply a dynamic programming algorithm, the forward-backward algorithm, to compute them in polynomial time.

```
...
a actividade ||| the work of ||| 0.003 0.002 0.004 0.001 2.718
||| 1-1 2-2 2-3 ||| 546 509
a actividade ||| the work ||| 0.013 0.002 0.081 0.021 2.718
||| 1-1 2-2 ||| 3243 509
a actividade agricola ||| agricultural activities ||| 0.091 0.003 0.048 0.057 2.718
||| 1-2 2-2 3-1 ||| 11 21
a actividade agricola ||| agricultural activity ||| 0.108 0.044 0.190 0.311 2.718
||| 1-2 2-2 3-1 ||| 37 21
a actividade criminosa ||| criminal activity ||| 0.018 0.004 0.333 0.290 2.718
||| 1-2 2-2 3-1 ||| 114 6
a actividade das ||| operation of ||| 0.0038 0.001 0.027 0.002
||| 1-0 2-1 ||| 290 36
a actividade das ||| the activities of ||| 0.0189 0.0007 0.1111 0.0096
||| 0-0 1-1 2-2 ||| 211 36
...
```

Figure 3.4: A sample of phrase pairs from a Portuguese - English phrase table, indicating the source phrases, the target phrases, the translation probabilities, the alignments of the words in the phrases and the phrase counts, respectively.

Phrase tables map word sequences in a given stream to word sequences in a different stream. They were first used in PBSMT, where they yielded more robust results than single-word translation models. In order to build phrase tables connecting two languages, we need a parallel training corpus that contains a sentence in the first language aligned to the corresponding sentence in the second language. Then we run GIZA++, which iterates through increasingly complex models (IBM1, HMM, IBM2, IBM3 and IBM4), which progressively relax assumptions imposed by the lower-order models, in order to generate bidirectional word alignments. From these alignments, using an heuristic, Moses [49] is executed to generate the target phrase table. The generated phrase tables are often very large, so we apply a further pruning step, removing phrase pairs that are not very frequent in the corpus or have low probability. An example of such a phrase table can be seen in Figure 3.4. It consists of a sequence of phrase pairs, where each phrase pair contains, delimited by "|||", each of the source and target phrases, followed by the direct and inverse translation probabilities, word-by-word alignments within the phrase, and finally, the number of occurrences of each phrase in the training corpus. Note that, while this is the most straightforward method to build phrase tables to be used in our method, it is not the only possible one.

For certain stream pairs, parallel training data may not be available, but information about

the domain in the form of a set of rules may be possible to obtain, as in Rule-Based Machine Translation (RBMT). A way of integrating this information within the PBSMT framework is to generate a set of aligned sentence pairs, where the target sentences are generated by applying the rules to the source sentences, and the resulting parallel corpus is used to train the phrase table [17, 25].

3.2.2 Baseline systems

We selected Audimus [63], a hybrid WFST-based recognizer, that combines the temporal modeling power of HMMs with the pattern discrimination ability of multilayer perceptrons, as our baseline ASR system. Our feature extraction setup consists of generating multiple feature streams - RASTA [38], PLP [37] and MSG [46] - which are then combined at the MLP posterior level. We used our existing English, Italian, Portuguese, Spanish and German acoustic models and lexica [65]. The acoustic models that were used are diphone models that consider the preceding phone as context. Note, however, that all context modeling occurs at the intraword level (the system does not consider interword diphones). The language models needed for each of the languages were trained using SRILM [92] from the monolingual versions of the Europarl Parallel Corpus texts [47].

A survey of existing speech recognition systems in the European Parliament domain led to the use of the RWTH European Parliament Plenary Speech (EPPS) one-pass recognition system [57, 85], in order to create a baseline against which to compare our results. The system's front end consists of extracting MFCC features. It uses a triphone acoustic model with 900,000 Gaussian densities, trained on the TC-Star data. The language model is a 4-gram with a vocabulary of 60,000 words, which was estimated using the transcriptions of the TC-Star data and the Final Text Editions of the European Parliament. All the recognition experiments with this system were run using the default parameters of the demo system [57].

3.2.3 Data sets

We next describe the two data sets used for testing our approach.

3.2.3.1 Multiple interpreted streams in the European Parliament

The official texts produced by the European Parliament are translated into each of the languages of the EU Member States. Similarly, speeches both in the Plenary Sessions and in the Committees are intepreted into the official languages of the EU, depending on the availability of interpreters. Although politicians are allowed to speak in their native language, they often prefer to speak in English, which means that a significant proportion of the English speech in the European Parliament is non-native. Also, many of the interpreters are non-native speakers of the language or variety that they are interpreting into; one example are the Brazilian interpreters for the European Portuguese language. In addition to that, and depending on interpreter skill and the complexity of the original speech, interpreter speech can be highly disfluent.

For this data set, we collected and manually transcribed two sets of English speeches (both from native and non-native speakers) from the Environment, Public Health and Food Safety (ENVI), Development (DEVE), Internal Market and Consumer Protection (IMCO) and Legal Affairs (LEGAL) committees. The first of these sets was used for tuning the parameters for phrase pair selection mentioned in Section 3.3.2, whereas the other, consisting of 4 speeches, was used for testing. We then collected the interpreted versions of these speeches in the German, Italian, Spanish and Portuguese languages. These versions were also manually transcribed. In addition to the transcription, we also annotated the occurrence of punctuation marks and the location of disfluencies in all of the collected speeches.

For this scenario, we used the Europarl Parallel Data [47], to train phrase tables for the various language pairs with the Moses toolkit [49].

3.2.3.2 Lecture speech and slides

The other data set was extracted from the Stanford online Natural Language Processing (NLP) course. Our test set consisted of 8 lectures from the course, together with the slides for each of the lectures. Four other lectures constituted a held-out development set used for parameter tuning. All of the lectures, both in the development and testing sets, were given by the same speaker. As reference transcripts for computing WER, we used the subtitles that were manually created for the course, although these contained a number of errors, since they were created by volunteers (so the computed WER may be artificially high).

3.3 Baseline algorithm

In this section, we describe each of the system components depicted in Figure 3.1, and evaluate the resulting system in the European Parliament simultaneous interpretation task, comparing it with a baseline that uses speech recognition only.

3.3.1 Lattice - phrase table intersection

The intersection between the lattices representing a pair of streams through the phrase table connecting them lies at the core of the multistream integration algorithms described. It consists of finding those phrase pairs $source \mid\mid\mid target$ which simultaneously belong to the phrase table, for which $source$ can be found in the source lattice and $target$ in the target lattice. Additionally, the source and target phrases must be found 'close' together in the respective lattices. The notion of close depends on the type of stream that is being used. For instance, if one is using a pair of speech streams in which one of the streams is a simultaneously interpreted version of the other, one might set a fixed parameter, for instance $\delta = 10s$, controlling the maximum expected delay from the interpreter. If, instead, we know how different speech streams are segmented into sentences, we can set δ in such a way that only phrases from equivalent sentences are grouped together. Alternatively, if one of the stream consists of subtitles along with time stamps for another speech stream, one could set δ appropriately to

account for the maximum period of time during which subtitles are displayed on the screen. Of course, in the case of interpreted streams, the interpreted version of a phrase is required to appear, in time, after the original version, so in those cases the maximum distance δ is one-sided.

The intersection step is potentially computationally expensive since, depending on the type of streams used, the generated lattices can be extremely large. For instance, lattices produced for a few minutes of speech can have millions of nodes and edges. Simply enumerating and comparing phrases in the lattices and phrase tables would not be feasible. So, we developed a number of techniques to alleviate this problem. Our approach consists of a three step algorithm. It is composed of preprocessing, intersection and post-pruning steps which we describe next.

In the preprocessing step we first convert each lattice into a forest (a set of trees, one for each lattice node) and each phrase table into a tree. To convert a phrase table into a tree, we sequentially insert each of its phrase pairs into it. In order to do so, we first insert each word of the source phrase, starting at the root and following existing tree branches, and then each word of the target phrase. Then, we process each of the lattice nodes into a tree of depth k, where k is the length of the longest phrase to be considered, by adding to the tree all the sequences of k words or less that can be reached from that node. We also annotate each node n of the tree with the posterior probability of the phrase corresponding to the path from the root to n, together with the end time of that phrase. This information is used by later stages of the multistream combination pipeline, namely to extract features for phrase pair scoring.

Figure 3.5 describes the intersection process in pseudo-code after both the lattices and the phrase table have been converted into trees, as described above. At this point, the intersection process can be recast as simply walking down the phrase table tree, and the lattice trees, simultaneously. On the source side of the phrase table tree, we keep only those paths that are also in the source lattice tree, and on the target side those which are also in the target lattice tree.

```
function intersect_source(n_i, pt_i)
rt ← new result_tree()
for (w, t) ∈ common_children(n_i, pt_i) do
  if is_source_node(child(pt_i, w)) then
    rt_i ← intersect_target(child(n_i), child(pt_i))
    add_child(rt, rt_i)
  else
    rt_i ← intersect_source(child(n_i), child(pt_i))
    add_child(rt, rt_i)
  end if
end for
return rt
end function

function intersect_target(n_i, pt_i)
rt ← new result_tree()
for (w, t) ∈ common_children(n_i, pt_i) do
  rt_i ← intersect_target(child(n_i), child(pt_i))
  add_child(rt, rt_i)
end for
return rt
end function
```

Figure 3.5: Pseudo-code for generating the intersection between a phrase table and two trees representing the source and the target lattices, respectively.

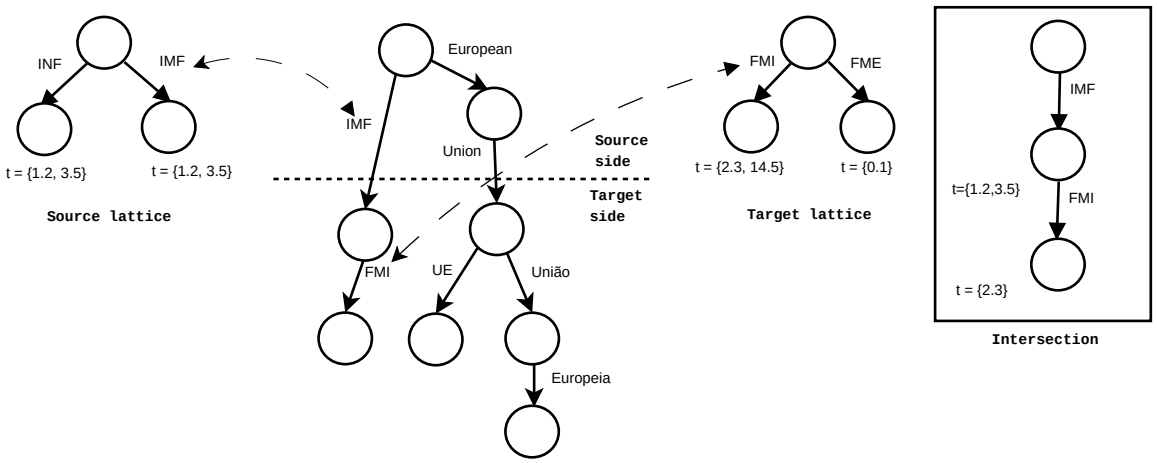

Figure 3.6: Intersection between an EN-PT phrase table, a EN source lattice, with phrases "IMF" and "INF", and a PT target lattice, with phrases "FMI" and "FME". The phrase table contains the pairs 'IMF ||| FMI', 'European Union ||| UE', and 'European Union ||| União Europeia'. The dashed arrows represent tree branches matched during the intersection process.

Finally, we perform the post-pruning step. Due to the way in which the intersection is constructed, some of the resulting nodes and branches may not be productive. In other words, there is no guarantee that, starting from any node in the result tree, we will find a path to a completed phrase pair. These nodes and branches are therefore pruned in a post-processing step, which consists of a post-order depth-first traversal of the intersection tree. The process removes nodes which do not correspond to completed phrase pairs and for which no child has a path leading to a completed phrase pair node.

Figure 3.6 illustrates the preprocessing and intersection for a simple phrase table, containing the phrase pairs 'IMF ||| FMI', 'European Union ||| UE', and 'European Union ||| União Europeia', a source lattice with 'IMF' and 'INF', and a target lattice with 'FMI' and 'FME'. We begin by considering the branches at the root of the source lattice that match with those at the root of the phrase table; only 'IMF' matches, so that is added to the intersection, including the times of occurrence. A leaf of the tree that represents the source lattice has been reached, so the algorithm switches to the root of the tree that represents the target lattice. The only branch that matches with the outgoing branches of the phrase table's current node is the one labeled 'FMI', so that is added to the intersection - note that the target entry at time 14.5

is dropped because the source entries are at times 1.2 and 3.5 and therefore too far apart for a $\delta = 10$ - and the algorithm terminates. Again, we reached a lattice leaf, and there are no remaining pending decisions to backtrack to. Also, the post-pruning process does not change the resulting tree, since all the created nodes are productive.

3.3.2 Phrase pair selection and rescoring

Clearly, we should not add all the phrase pairs to the calculated intersection, since some of these might have occurred out of chance. For instance, a phrase pair such as 'and ||| y' (in an English-Spanish speech stream pair) occurs very often in the lattice-phrase table intersection, but not in the streams themselves, i.e., in what has actually been said. These words are both frequent in their respective languages, and acoustically confusable with other small words, and so they will often be contained in the lattices generated for each of the streams. If they appear within a sufficiently short time interval, the corresponding phrase pair will be extracted by the proposed method.

In light of the problem described above, we train a simple linear model to predict whether a generated phrase pair pp is actually present in the speech streams. We compute its score according to the expression $SC(pp) = a_0 + \sum_{i=1}^{N} a_i f_i(pp)$ where N is the number of features, the f_i are feature functions, the a_i are feature weights and we select those pairs which have $SC > 0$.

The features used were the following:

- posterior probability – the posterior probabilities of each of the phrases in the phrase pair, obtained from the lattices generated for each stream, as described in Section 3.2.1.
- phrase table features – the phrase pairs are extracted from Moses format phrase tables, which include both direct and inverse translation and lexical probabilities. We expect phrase pairs with high translation probabilities to be more relevant than those with lower probabilities.

- language model scores – the n-gram language model probabilities of each of the phrases, as well as individual words in the phrases are also included. Phrases with high language model probabilities are relatively common and therefore should have a lower overall score.

- word counts – the number of words in each phrase of the phrase pair. For example, the phrase pair 'european parliament ||| parlamento europeu' has a word count of four. The higher the total number of words in a phrase pair is, the lower the likelihood that it has not appeared in the respective streams.

- frequency features – the number of times each of the phrases in the phrase pair appears in the initial transcription, the idea being that phrases that have occurred frequently in the transcriptions are likely to occur again, and therefore should be assigned a higher score compared to phrases that never occurred.

The feature weights (a_i) used for computing the SC function are selected by using a line search method to optimize the WER, averaged across each of the streams being rescored, on the development set. The direct evaluation of this objective function would, however, be very expensive, since it would require performing the whole process including lattice rescoring for each parameter setting. Instead, we optimize a closely related function: the total number of errors corrected by the constructed alignment (alignment words that are in the reference but that were not correctly recognized in the initial decoding step), from which we deduct the total number of errors introduced by the alignment (alignment words that are incorrect, i.e, that are not in the reference transcription). This function is less expensive to compute, since no lattice rescoring step is necessary. It is sufficient to run the alignment construction algorithm (Section 3.3.3) once for each evaluation.

3.3.3 Alignment construction

In Section 3.3.2, we select a subset of the extracted phrase pairs that we expect to be correct. However, we cannot use this set as our generated alignment directly, since some of these

	parlamento	12.2–12.9	-0.5		parliament	11.3–12.3	-6.5
	lamento	12.4–12.9	-5.6		lo siento	10.2–10.7	-0.2
	exige	12.9–13.3	-0.0		demands	12.3–12.6	0.0
	demanda	13.8–13.8	-0.6		demands	12.3–12.6	0.0

Figure 3.7: Example of a subset of an inconsistent alignment. The alignment is inconsistent because both phrase pairs 'parlamento/parliament' and 'lamento/lo siento' have a phrase drawn from the Portuguese stream, those phrases 'parlamento' and 'lamento' are incompatible and they overlap in time.

phrase pairs may overlap with each other. In particular, consider the case where we extracted two distinct phrase pairs, based on the same stream pair but overlapping time spans. Figure 3.7 illustrates such a scenario. In such a case, a conflict will arise since it is not possible to include both phrase pairs in the output, which means it will be necessary to discard one of them.

Therefore, instead of using the extracted phrase pairs directly, we set our target alignment to be the subset S of those phrase pairs that maximizes function f, defined in Equation 3.1, while being internally consistent.

$$f(S) = \sum_{i=1}^{n} \alpha SC(pp_i) + \beta \sum_{i=1}^{n} \sum_{j=1}^{n} (-dist(pp_i, pp_j) + adj(pp_i, pp_j)) \tag{3.1}$$

A phrase pair pp_i is considered to be adjacent to a phrase pair pp_j ($adj(pp_i, pp_j)$ in Equation 3.1) if they have at least two phrases in the same stream and the start time of one of the phrases is equal to the end time of the other. By assigning a bonus to adjacent phrase pairs, we acknowledge the fact that sequences of matched phrases are unlikely to occur by chance.

Furthermore, the distance ($dist(pp_i, pp_j)$) between two phrase pairs pp_i and pp_j is computed as follows:

- If the two phrase pairs do not correspond to exactly the same streams r and s, then the distance is zero, since we do not wish to compare phrase pairs in different streams.

- If the time difference between the two phrase pairs, in either of the streams, is greater than a previously set threshold, the *influence radius* for phrase pairs, then $dist(pp_i, pp_j)$ is set to zero.

- Otherwise the distance is set to be $dist(pp_i, pp_j) = |(st_{i_r} - st_{i_s}) - (st_{j_r} - st_{j_s}))|$, where st_{i_r} indicates the starting time of the phrase of phrase pair pp_i that corresponds to stream r.

The idea behind computing this distance is that we expect stream pairs to be locally synchronized. In other words, given two phrase pairs pp_i and pp_j that are in the alignment, with phrases on the same streams r and s, if we extract the phrases from each phrase pair that correspond to stream r, we would expect them to have approximately the same time distance than those that correspond to stream s.

The alignment construction process may be best understood as a subset selection problem. In fact, among all the *consistent* subsets S of the generated phrase pairs, we search for an S that maximizes $f(S)$, defined in Equation 3.1, and satisfies the consistency constraint. A subset $S = \{pp_1, \ldots, pp_n\}$ is said to be consistent iff $\forall_{pp_i \in S, pp_j \in S} \neg overlap(pp_i, pp_j)$ or, in other words, if it is not possible to find two overlapping phrase pairs in the subset S. We define $overlap(pp_i, pp_j)$ to be true if and only if the following conditions are cumulatively satisfied:

- Phrase pairs pp_i and pp_j share two phrases pp_{i_s} and pp_{j_s} belonging to a stream s.

- pp_{i_s} and pp_{j_s} occur at overlapping time spans.

- pp_{i_s} and pp_{j_s} are incompatible, that is, neither pp_{i_s} is a substring of pp_{j_s}, nor pp_{j_s} is a subset of pp_{i_s}. In Figure 3.8, the alignment is consistent because, although phrases 'há' and 'há várias' overlap in time, the former is contained in the latter.

There is an exponential number of subsets of the extracted phrase pairs, which makes it intractable to perform an exhaustive search of all the subsets to find S that maximizes f. Furthermore, due to Equation 3.1, adding or removing a given phrase pair to the alignment

	há	10.4–10.5	-0.0		there are	11.3–12.3	-6.5
	há várias	10.4–10.8	-0.1		hay muchas	10.2–10.7	-0.2

Figure 3.8: Example of a subset of a consistent alignment. The alignment is consistent, despite the overlap in the Portuguese stream, because the phrase 'há' is contained in the phrase 'há várias', rendering them compatible.

	há	2.4–2.7	-0.5		there are	5.5–5.9	-18.5
	muitas	2.7–3.1	-5.6		many	5.9–6.1	-1.2
	hipóteses	3.1–3.7	-0.0		possibilities	6.1–7.0	0.0

Figure 3.9: Example of an adjacent phrase pair sequence, consisting of three phrase pairs. The ending time for each phrase corresponds to the starting time of the next phrase in the sequence, thus forming a connected phrase chain.

causes more than simply a local impact in the computation of function f: it influences all the distance terms involving that phrase pair.

A possible way to build the aligment would be to add phrase pairs pp in the decreasing order of their scores, $SC(pp)$, skipping those phrase pairs that conflict with the ones that are already in the alignment. That would, however, not lead to the optimal solution. In certain cases, adding two phrase pairs to S might increase the value of f more than adding one, but if the former conflict with the latter, the two phrase pairs would not be added to the alignment. This led us to consider a procedure in which we maximize $f(S)$ using steepest-ascent hill climbing, but with a modified successor generating function. Therefore, we take an alignment S and generate a list of successors S' of S, by applying one of the following heuristics to S:

- Adding any phrase pair that is not already in S.
- Adding a sequence of adjacent (as defined for 3.1) phrase pairs (Figure 3.9).
- Adding a set of consistent phrase pairs spanning multiple languages (Figure 3.10).

If any phrase pairs in the original alignment are in conflict with the phrase pairs being added, then these are removed from the successor alignment to ensure that it is still consistent and

desenvolvimento	5.6–6.4	-0.0	desarrollo	7.3–7.8	-0.0
desenvolvimento	5.6–6.4	-0.0	development	4.3–5.0	-0.0
desenvolvimento	5.6–6.4	-0.0	sviluppo	6.8–7.6	-0.0

Figure 3.10: Example of a phrase spanning multiple languages, in this case four. In this alignment subset, the phrase "desenvolvimento" in Portuguese matches with its respective translation in Spanish, English and Italian.

therefore a valid alignment. By adding multiple phrase pairs at a time to S, it is possible to obtain an alignment with a higher score, even if adding each of these phrase pairs individually decreased the score, because it required the removal from the alignment of a conflicting phrase pair.

3.3.4 Lattice rescoring / decoding

The final step consists of a rescoring of the lattices associated with each of the rescorable streams, which in this work are limited to speech streams. Other types of rescorable streams could be considered, if a suitable rescoring algorithm were to be defined. Such an algorithm would be specific to that particular stream type.

At this point, the streams are again decoupled and lattice rescoring proceeds independently for each of the rescorable streams. First, the obtained alignment is projected into each of the streams being rescored, an operation that preserves the unique phrases in that stream together with their time stamps. For example, the alignment in Figure 3.8 would yield "there are / 11.3–12.3" if projected into the English stream.

For speech streams, the rescoring is carried out using the A* algorithm. Each of the phrases in the projection is assigned an additive bonus (in the log-score domain), which is applied to the language model scores of these phrases during rescoring, whereas the original acoustic scores are left unmodified. This biases the decoder towards these phrases, making them more likely to be recognized. This bonus is only a function of the number of words of the phrase, and is determined empirically on the development set. The time stamps derived during the

Speech	EN	+PT	+ES	+IT	+PT+ES	+ES+IT	+PT+ES+IT	All
DEVE	18.70	16.87	16.63	17.15	16.81	15.78	15.43	15.61
ENVI	19.70	18.19	18.78	16.08	17.49	15.41	15.34	14.40
IMCO	30.73	27.90	25.79	27.77	26.33	26.42	25.18	24.31
LEGAL	29.18	27.81	28.95	26.54	24.78	24.45	23.60	23.22
Average	25.51	23.60	23.40	22.70	22.09	21.32	20.63	20.05

Table 3.1: WER (%) for the 4 speeches. The 1^{st} column is the error of the baseline system, the 2^{nd} and represents the WER of the original English speech after combining with Spanish, and so on for different language combinations. The last column represents the WER obtained when combining with all the languages (PT, ES, IT and DE).

intersection step are used to ensure that we only assign a bonus to occurrences of phrases at the appropriate times. In this way, the language model scores are modified only in the vicinity of the location where a given phrase pair occurs, rather than for a whole utterance or speech. This is more precise, since it limits the scope of the LM changes created by a phrase pair, and reduces the number of errors that can be introduced by the modified LM. This is especially important for long sentences and phrases that occur frequently. For instance, considering the alignment of Figure 3.7, if the phrase "there are" occurs at times 9.5 and 11.3, only at time 11.3 will it be assigned an LM score bonus. Of course, the resulting language model is no longer an n-gram model. However, we can still apply hypothesis recombination to speed up the search, and no modifications are required if the extracted phrases are not longer than the n-gram context length.

3.3.5 Evaluation

We executed the proposed method, using the same data set described in Section 3.2.3.1. Table 3.1 summarizes the main results, when combining English (the original language of the speeches) with several combinations of the remaining four languages. We observe improvements in the English WER, relative to the baseline, when using just one, two, three or four languages to combine with the original (English versions). We can also see that the improvements, on average, increase with the addition of further languages. There are different improvements, depending on the sets of languages used to perform the combination. Some of these variations are naturally caused by differences in the interpretation quality -

an interpretation that is closer to the target language will have a larger number of phrase pairs available for matching with other streams, and therefore will improve the results to a larger extent than an interpretation that has a large number of errors and omissions. We can also see that the improvement in WER of the best combination increases with the number of languages, although the improvement decreases, in terms of absolute WER delta. This suggests that using a larger number of language helps, but that the improvements will tend to saturate after adding a few languages.

3.3.6 Oracles

In order to obtain an assessment of the maximum improvement achievable and therefore situate the improvements achieved by the proposed method, we developed two oracles. Both of these oracles rely on forced alignments of the test set to the reference transcripts, which provide information not only about the words that were actually said but also about the times at which they occur.

The first oracle (oracle A) evaluates the maximum improvement that could be expected from the proposed system, if the generated alignments were perfect. In other words, given the phrase pairs extracted from the intersection between the lattices through the phrase tables, we build the optimal alignment. Here, optimal is defined with respect to the reference transcripts - it refers to an alignment for which all phrase pairs are grounded in the references, meaning that both phrases of each phrase pair actually occur in the respective references.

In order to build oracle A, we only add to its alignment those phrase pairs that agree with the reference transcripts. For each phrase of every generated phrase pair, we check if that phrase appears in the reference transcript, with the same time stamp. If that is the case, we add the phrase to the optimal alignment and proceed with the remainder of the algorithm, which consists of rescoring the lattices with this alignment.

The second oracle (oracle B) evaluates the maximum improvement that could theoretically be obtained from the lattices, regardless of any restrictions imposed by the phrase pair extraction

Speech	Baseline	Best	RWTH	Oracle A	Oracle B
DEVE	18.70	15.61	18.23	12.58	9.15
ENVI	19.70	14.40	22.05	9.34	6.11
IMCO	30.73	24.31	26.26	16.57	10.89
LEGAL	29.18	23.22	25.60	13.38	9.62
Average	25.51	20.05	23.70	13.17	9.02

Table 3.2: WER (%) for the English versions of the 4 speeches. The 1^{st} column is the error of our baseline ASR system. The 2^{nd} column is the error of the system that combines English with the other languages. The third and fourth columns represent the error of oracles A and B, respectively.

process. In other words, oracle B considers all the phrases in the stream lattices, even those that do not have a correspondence in other streams. If we consider all the lattices for a given stream concatenated together sequentially, to form one large lattice that represents the entire stream, then the second oracle consists of finding the path through the lattice (word sequence) which minimizes the word error rate or, alternatively, the edit distance to the reference transcript. From this definition it becomes apparent that oracle B performs at least as well as oracle A, since the former selects optimal lattice paths and the alignments are built using words from the lattices.

We implemented the two oracles (oracle A and B) mentioned above. We then contrasted them with the developed system, on the test set described in Section 3.2.3.1. Table 3.2 summarizes these results.

We see that both oracles clearly improve on the performance achieved by our baseline system, in some cases by over 50% relative, which indicates the existence of a significant margin for the proposed system to decrease word error rates. It is also the case that, as expected, oracle B is always at least as good as oracle A.

We inspected the alignments produced by oracle A, so as to try to determine the reason for the differences in performance between the proposed system and oracle A. We observe that the majority of the differences between the alignments generated by the two systems were phrase pairs consisting of function words or phrases and their respective translations. An example of such phrase pairs, with phrases in the Portuguese and English streams, would

be *de* ||| *of* or *que* ||| *that*. These are often the hardest to add correctly to the generated alignments, because there are often multiple possible matches for a function word in each of the streams, and not always enough information to decide which (if any) of these matches is correct. Naturally, oracle A has a much better performance in these cases because, by construction, it knows exactly which of these pairs to include in the final alignment.

Finally, it can be seen, from the error rate of oracle B, which is not zero, that there is a considerable number of words that cannot be found in the generated lattices (which we call *OOL words*), which motivates the recovery of these words.

3.4 Out-of-lattice word recovery

Lattices generated for speech streams often contain a large number of alternative hypotheses for a given utterance, but in most cases, due to a high degree of mismatch between the acoustic models and the actual pronunciations, they do not contain all the words that were actually said. Equivalently, when the lattice oracle B WER is greater than zero, which is the case in the results of Table 3.2, there are words and phrases that cannot be recovered by the method presented in Section 3.3.

In light of this, in the current section we extend the algorithms presented in Section 3.3 to account for words and phrases that, for a number of reasons such as a pronunciation mismatch with the lexicon, were not present in the original ASR generated lattices, but which if recovered could lead to improved performance. We explore this topic by both recovering OOL words and acronyms, and trying to use the information we gathered to recover pronunciations in order to improve our pronunciation dictionaries. Note that, although we only recover OOL words and acronyms in speech streams, because this operation is not well-defined for other stream types, streams of all types can be used, whenever appropriate, as supporting information to help in the recovery of these words.

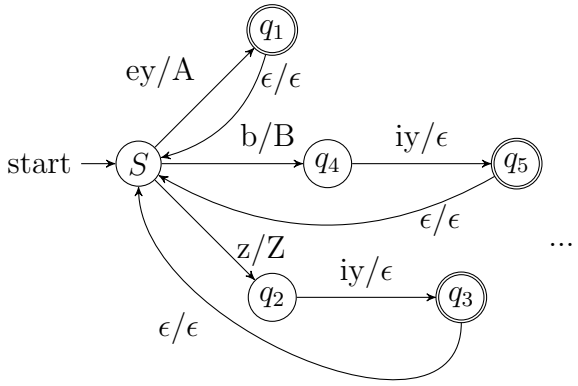

Figure 3.11: An FST to detect and recover acronyms (matches only one letter of the acronym, it would have to be repeated to recover the complete acronym)

3.4.1 Acronym recovery

Abbreviations or acronyms are very common in parliamentary speech or in technical talks. However, their correct recognition presents a number of challenges, since many of them are not present in the vocabulary. Also, acronyms can usually be pronounced in one of two different ways: either as a regular word, or by spelling out each of its constituent letters in sequence. Our acronym recovery method focuses on the latter. Additionally, our acronym recovery techniques assume that the acronym has the same form in the original and interpreted streams - often, due to time constraints, interpreters are unable to reorder the letters in an uncommon acronym and they leave it as in the original. An example of this is the acronym IPOA (Istanbul Programme of Action); Portuguese interpreters will often say IPOA when the correct acronym would be PAI (Programa de Acção de Istambul).

To recover abbreviations, we first search for candidates 3 to 5 letters in length in the streams, such as IPOA (Istanbul Programme of Action) or EBA (European Banking Authority). We also group together plural versions of the candidate abbreviations, for instance, LDC (Least Developed Country) and LDCS (Least Developed Countries).

To search for the abbreviations, we build a finite state automaton which encodes all possible abbreviation sequences (as depicted in Figure 3.11), and compose it with a previously generated phone lattice. We therefore select the most plausible abbreviations at each time step,

together with their acoustic score, thus generating a candidate list. Most of the generated abbreviations are not useful, so only a subset of these is selected for further processing. To rank the candidates according to the likelihood that they are actually present in the speech, we extract a number of features, namely:

- The number of different streams in which the acronym appears and its total number of occurrences in all of the streams.
- The number of letters in the acronym.
- Whether the acronym had been spelled out before its occurrences. It is often the case, when introducing an acronym with which the audience may be unfamiliar, that the speaker will explain its meaning by spelling it out. For example, to introduce the acronym UNSC, the speaker might say "The United Nations Security Council (UNSC) has decided ...". We guess whether that happened by comparing potential acronyms to word sequences in the transcriptions, and use this information as a feature.
- The average score of the acronym occurrence and the acronym's language model score.

The language model used for acronym recovery is a trigram language model trained from a list containing 500 abbreviations using SRILM [92].

The feature values are linearly combined to form a score, and the top k (where we fixed $k = 10$) acronyms according to this score are selected and added to the phrase table / lattice intersection result before the alignment is rebuilt, using the method described in Section 3.3.2.

Once we obtain the new alignment, we proceed with the rescoring of the streams described in Section 3.3.4. Before we do this, we have to modify the original lattices for each of the streams for which we have recovered acronyms. For each such acronym, we find all the pairs of states in the respective lattice that have the same start and end times as the acronym, and add an arc to the lattice that connects these two states and has the acronym as label. The acoustic score of this arc is that which was determined when generating the candidate acronym list.

3.4.2 Out-of-lattice word recovery

Although our method considers multiple alternatives in the form of lattices, these are finite and therefore cannot contain all of the possible word sequences. In several cases it may be desirable to recover these words that do not appear in the output lattice, since this can improve transcription accuracy. We generate a list of locations (time intervals) where these OOL words could potentially appear, according to the following criteria:

- If a certain word appears in two or more streams, its translation is predicted to appear in the third and subsequent streams as well, around the corresponding time in these other streams. For example, suppose that the word "Aarhus" appears close in time in the most likely decodings of the Spanish and Portuguese speech streams, but not in the English stream lattice. We therefore hypothesize that "Aarhus" is an OOL word in the English stream, which did not appear in the generated English lattice due to an acoustic model or pronunciation mismatch.

- If two words are aligned in a given stream pair, then the surrounding words are also predicted to be translations of each other. For instance, if "European", in English, is aligned with "Europeia" in Portuguese, and the word "Commission" follows the word "European" in the English transcription, then it is natural to assume that the word "Comissão" will precede or follow the word "Europeia" in Portuguese, even if it cannot be found in the corresponding lattice. We therefore hypothesize the occurrence of an OOL word ending near the beginning of "Europeia".

When selecting the potential translations of a given phrase to a target language, we use the translations in the appropriate phrase table that score above a manually pre-selected threshold. We then intersect an FST representing the possible pronunciations of the phrase that we are trying to recover with the phone lattice which represents a phone decoding of the target stream. The obtained acoustic score is subsequently compared to a fixed threshold; if it is below, this potential OOL word occurrence is discarded. Otherwise, it is carried on to

the next step where it is added to the phrase pairs that originate from the phrase-table-lattice intersection process. The phrase pair scoring method described in Section 3.3.2 is augmented to use an extra feature, which indicates whether the current pair is an OOL pair. In this way, we can have OOL phrase pairs compete in a balanced way with regularly extracted pairs.

At this point, we generate a new alignment using the method described in Section 3.3.2. Finally, and in a way comparable with what is done for the recovery of abbreviations, recovered words are added to the lattices as new edges at the appropriate locations, before the execution of the lattice rescoring step.

3.4.3 Pronunciation recovery

Pronunciation lexica are key components of ASR systems - if the correct pronunciation is not in the lexicon, recognition performance degrades substantially - and can be notoriously hard to build. Manually creating these dictionaries is a very laborious and expensive task, and most languages have an open vocabulary, in the sense that new words are continuously added or imported from other languages. This means that a combination of manual and automatic methods is often used to generate these pronunciations. However, in many languages, such as English, the pronunciation of a word is very hard to predict from its ortography alone. Words imported from foreign languages, as well as names of people and locations, also pose important challenges.

In this section, we capitalize on the multi-stream alignment that we generated, described in Section 3.3.3. The idea is that the words that we were able to recover by way of increasing their language model scores are more likely to have been pronounced in a manner which differs from their dictionary pronunciation than words that were recognized correctly in the first pass. From the alignment, we select high-confidence words - those that are part of one or more phrase pairs with scores that sum to a value greater than a manually selected threshold. For each of the selected words, and given their occurrence times, we compute the most likely phonetic transcription by performing a Viterbi search on the phone lattice. This process yields a set of potential pronunciations for each word.

At this point we compute, using a string edit-distance algorithm, the number of insertions, deletions and substitutions required to transform each of the potential pronunciations to the closest among the reference (dictionary) pronunciations. The alignment of the two strings defines a sequence of operations (deletions, insertions and substitutions) that one would have to apply in order to transform the reference pronunciation into the potential pronunciation. We select the subset of the potential pronunciations that can be obtained by performing at most two such operations, creating a candidate pronunciation set $p_1 ... p_k$. Each of $p_1 .. p_k$ is assigned a score by Equation 3.2, where p_0 is the original pronunciation:

$$SC(p_i) = \alpha L(p_{i_1} .. p_{i_n}) + \beta \frac{\sum_{j=1}^{v} A_i(o_j)}{v} + \gamma \delta(p_0, p_i) \tag{3.2}$$

In Equation 3.2, v stands for the number of occurrences of the word, $L(p_{i_1} .. p_{i_n})$ denotes a 5-gram language model over phone sequences, whereas $A_i(o_j)$ indicates the acoustic score of the j^{th} occurrence of the word assuming pronunciation i, and $\delta(p_0, p_i)$ is the edit distance between the dictionary pronunciation and pronunciation p_i. The idea behind this computation is to select a pronunciation that acoustically agrees with most occurrences, while at the same time avoiding unlikely phone sequences through the use of a language model, as well as avoiding adding new lexicon entries due to minor pronunciation variations.

We then select $p^* = \arg\max_i SC(p_i)$ as the pronunciation to be recovered and add it to the lexicon, if it is not one of the existing dictionary pronunciations.

The language models, one for each different language, are trained with the SRILM toolkit [92] with modified Kneser-Ney smoothing [16]. As training data, one can use manually generated lexica for a given language, which is usually preferable if these are available, or automatically generated lexica.

	No pron. recovery			Pron. recovery		
Speech	Base	+acr	+acr+ool	Base	+acr	+acr+ool
DEVE	15.61	14.63	14.29	15.37	14.14	14.04
ENVI	14.40	14.15	13.54	13.94	13.69	13.07
IMCO	24.31	24.20	23.93	23.91	23.80	23.45
LEGAL	23.22	22.80	22.68	22.81	22.39	22.22
Average	20.05	19.68	19.35	19.65	19.24	18.92

Table 3.3: WER (%) for the system improvements. The factor that differs between the left and the right half of the table is whether pronunciation recovery is applied. The leftmost column of each half represents the WER after the baseline method has been applied, but with no acronym (acr) or OOL word recovery; the middle column indicates the WER with acronym recovery; and the last column presents the WER with both acronym and OOL word recovery.

3.5 Experimental evaluation

In this subsection, we evaluate the impact of the proposed improvements - OOL words, acronyms, and pronunciation recovery, on the recognition results, by comparing these with the results obtained in Section 3.3.5.

At this point we integrate the improvements of Section 3.4. In order to take the impact of pronunciation recovery into account, we added the recovered pronunciations into the recognition lexicon after the baseline algorithm had completed, and executed another iteration of the process. Table 3.3 shows the results of our experiments for these two cases (where we apply, and do not apply the pronunciation recovery algorithm in order to analyze its impact).

In Table 3.3, we see that the recovery of acronyms only seems to significantly improve results in the talk from the DEVE committee. In fact, of the four tested talks this is the one with the largest proportion of acronyms and abbreviations. In other talks, results are also slightly improved, which suggests that we are not recovering many spurious abbreviations. On average, acronym recovery improves recognition accuracy 2.4% relative. Also, the results of Table 3.3 demonstrate improvements in performance with OOL word recovery, in a more uniform way across all of the different speeches. The average relative improvement from OOL word recovery is 1.3% relative. Finally, the presence of pronunciation recovery appears to affect results in a additive manner relative to the other factors, and has a positive impact of

Operation	Parallel RTF	Sequential RTF
PT - lattice intersections	0.69	1.96
Init. align. generation	0.18	0.18
Abbreviation recovery	0.23	0.23
OOL word recovery	0.42	0.42
Pronunciation recovery	0.51	1.34
Alignment generation	0.19	0.19
Final decoding	0.86	2.12
Total	3.08	6.44

Table 3.4: Average RTF, over the four testing talks, of each of the main operations of the algorithm (for a single iteration). The first column indicates the parallel RTF whereas the second column indicates the sequential RTF.

1.5% relative. Combining all the methods, we achieved an overall relative WER improvement of 5.0%, when compared to the second iteration of the baseline system. This translates to a cumulative 24.8% when compared with speech recognition only (without running the baseline system).

3.5.1 Running time analysis

In this section we empirically analyze the computational overhead, in terms of running time, incurred by the algorithms described in this chapter, for the case of $N = 4$ speech streams (corresponding to one original and three interpreted languages).

We measured the overhead of each of the developed components. Table 3.4 summarizes the real time factors of the algorithm, averaged over the four testing speeches, for one full iteration. We distinguish the sequential RTF from the parallel RTF. The latter assumes that a number of operations can be executed in parallel, since they are independent of each other and non-overlapping, and considers only the running time of the longest among these operations. The operations that can be executed in parallel are intersecting multiple phrase table - lattice pairs, the final decoding steps - obtaining improved transcriptions is parallelizable for the various streams since there do not exist any dependencies between the instances of the search algorithm - and pronunciation recovery, which is done separately for each of the languages. Note that no additional effort is required to parallelize these operations, since they are already

independent from each other. Table 3.4 shows that the methods presented in this work are responsible for 1.99 xRT sequential, whereas a full iteration of the algorithm takes 6.44 xRT sequentially, but only 3.08 xRT if it can be executed in parallel.

The total time complexity grows quadratically with the number of different streams N, which would be intractable for large N. However, the most time-consuming part of the algorithm - running a series of phrase table-lattice intersections - can be paralellized, and so given sufficient computational resources this would not slow the system down. Furthermore, the number of phrase table-lattice intersections can be kept to a minimum by selecting a subset of the $\binom{N}{2}$ possible intersections, in such a way as to minimize the impact in result quality. A possible way of doing this would be to select phrase tables for language pairs that we believe have close linguistic connections, particularly in what concerns word order, since this will probably increase the number of phrase pairs extracted when compared to more distant languages, and therefore improve the quality of the generated alignments.

3.6 Combination of lecture speech and slides

In applications such as multimedia indexing and retrieval, one often encounters the problem of aligning lecture speech with the slides that were used to support its presentation. Previous work [66, 44, 19] has focused on this problem, using similarity measures such as the cosine distance between the automatically generated speech transcripts and the slides, or through the use of dynamic programming algorithms. Recently, the problem of correcting ASR transcripts with slides has been addresed [95]. In this work, the authors view the phonemes output by the ASR system as distortions of the slide words, and seek to recover the true phoneme sequence using an HMM model with different states for slide and non-slide phonemes. In this Section, we show how we adapted the method described in Section 3.3 for the purpose of lecture recognition. Compared to this previous work, our method has the advantage that it is more extensible: it can, in principle, use slides that are in a different language from the speech, and it allows speech to be combined not only with slides but also with other streams such as speech in a different language. It also enables the words in a particular slide to affect

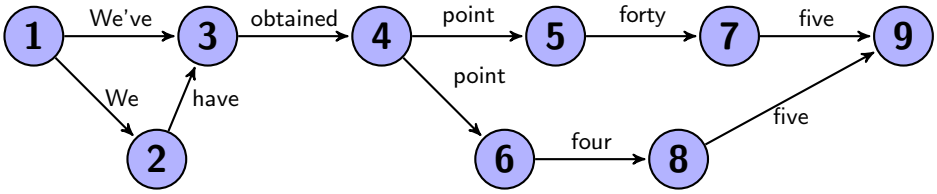

Figure 3.12: Lattice generated for the sample sentence "We've obtained .45". The alternatives "we have" and "we've" have been generated for the first word, whereas "point four five" and "point forty five" were generated for the third word.

the language model with higher precision, i.e., only when the speaker is using that slide as supporting material.

We considered the lecture speech to be one of the streams and the slides to be the other stream, and we dropped the initial decoding and rescoring steps for the slide stream, since we were not interested in recovering information from this stream. Note, however, that if that were our goal, we could have considered the slide stream as rescorable, and used the multistream combination process to learn, for instance, expansions for different abbreviations. The input to our system consists of the lecture audio files and the slides in Portable Document Format (PDF) format (which may contain slides from other lectures as well). Therefore, the slides have to be converted into lattices and we need to generate a phrase table connecting the two streams before our method can be applied. We next describe these modifications.

3.6.1 Converting from slides to lattices

The slides are first pre-processed by extracting the unnormalized text from the slides, using the *pdftotext* tool of XPdf [72]. The output produced by this tool contains some errors, and the sentences do not necessarily appear in a sequence that is related to the slide layout. Also, *pdftotext* is unable to extract text that is embedded in graphs or tables.

This text from the slides is then used to build a lattice which is used as input to the phrase table-lattice pair combination algorithm described in Section 3.3.1. To build the lattice, we keep a pointer to the most recently added node r. When processing a token t from the text,

we create a node n and add an edge labeled t to the lattice, which connects r and n. Certain tokens, such as numbers, are spelled out as multiple words. For instance, *12000* is spelled as *twelve thousand*, so in this case an additional intermediate node is added to the lattice. Still other tokens have multiple possible normalizations, depending on the speaker and context. In an equation, for example, the token $<$ can be spelled as *lower than*, *smaller than*, or *less than*. Since, in the absence of disambiguating information, we have no obvious way of choosing among them, we encode all of these as alternative paths between the nodes r and n, and use the flexibility present in the lattices, to allow the information in other streams to decide which of the alternatives to use. Figure 3.12 illustrates a lattice generated by this process for the sentence "We've obtained .45".

3.6.2 Modified alignment generation

As described in Section 3.3.1, our algorithm requests that we associate time stamps to each of the lattice nodes. However, the adaptation for lecture recognition means that this requirement must be relaxed since, unlike for speech streams, there is no time information directly available for slide words. On the other hand, we observe that lecturers usually proceed through the slides sequentially, with the occasional exception when it is necessary to revisit a topic. Under this assumption, it is logical to assign an approximate timing to the words in a slide. This would correspond to the time at which this slide was displayed by the speaker.

Therefore, our modified alignment generation procedure jointly computes the aligment and the time stamps of words and phrases in the slides. To do that, it ignores the time differences involving a slide stream in the first iteration. In particular, no phrase pairs are pruned during phrase-table / lattice intersection or the first iteration of the alignment generation process. Instead, at the end of the first iteration, we calculate the time stamps for the slides as follows: we take the speech-slide phrase pairs that were added to the alignment as *anchor points*, and to estimate time stamps for the remaining slide words and phrases, we linearly interpolate between the two closest such anchor points. We subsequently run a number of iterations of the algorithm, until the time stamps for the slide words converge or a predefined maximum

number of iterations is reached. The (progressively refined) time stamps, computed at the beginning of each iteration, are used as in the original procedure to calculate time distance features.

3.6.3 Phrase Table generation

The obvious way of generating a phrase table to serve as input to our algorithm would be to create identical phrase pairs for all of the phrases in the slides with less than a fixed number of words, and then to add these phrase pairs to the phrase table. However, this ignores the fact that the lecturer will often substitute the words in the slides with morphologically related words. Generating all the morphological variants of a word is beyond the scope of the present work, so we attempt to generate only the most common among these. Our approach consists of first obtaining the part-of-speech and lemma for each of the words in the slides. Then, for a singular noun such as *probability* we include its plural *probabilities* as a translation of *probability* in the generated phrase table. Analogously, we include the singular form for plural nouns. Similarly, for a verb such as *to work*, we add the past participle form *worked* as well as the gerund *working*. If we encounter, for example, the gerund form, then we add the infinitive and past participle forms, to the generated phrase table, as possible translations.

In order to test the performance of our algorithm, we compared the baseline, which consists of speech recognition only, with the proposed method. For evaluation purposes, we used the data set described in Section 3.2.3.2.

We then trained a domain-specific 4-gram language model using text extracted from a set of 10 computer science books. This language model was linearly interpolated with a 4-gram language model trained on the Hub4 text data, where the interpolation weight was estimated so as to optimize perplexity on our held out development set, creating language model A. We also trained a language model (language model B) that included both the computer science books and the supporting slides for all of the lectures in the development and testing sets. The text from the slides was extracted using the *pdftotext* tool and the resulting language model was interpolated with the 4-gram language model trained on the Hub4 data. By using

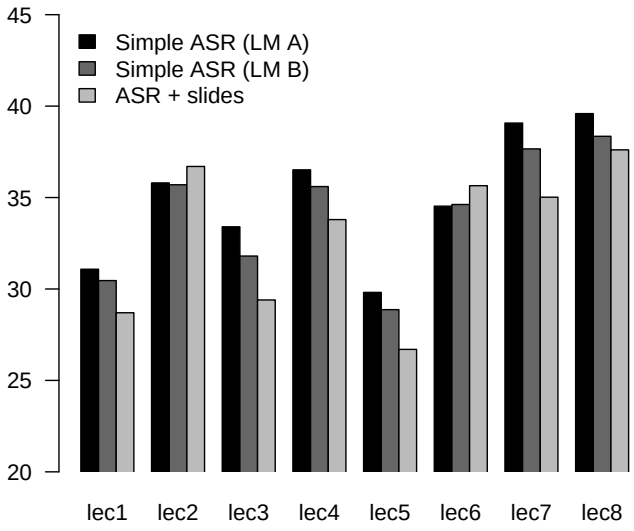

Figure 3.13: WER for each of the lectures in the test set, both when using speech recognition only and when combining the speech with presentation slides, for the two language models (LM A and LM B) used in the experiments.

a language model (language model B) which was trained on the same data as is available to our method, we are able to assess the improvement that is obtained by assigning a higher weight to words or phrases that are in the same or neighboring slides.

Both language model estimation and interpolation were carried out using the SRILM toolkit [92].

3.6.4 Evaluation

For each of the talks in the testing set, we ran the baseline speech recognition system with the two language models previously mentioned. We compared the results with the system developed to integrate the lecture speech with the slides, as shown in Figure 3.13. We observe an overall improvement in the results by using the proposed method, from a baseline WER of 35% to 32.9%, with an average relative WER improvement of 5.9%, when using language model A. If we use language model B, the impact of our method is smaller, as would be expected but there is still a relative WER reduction of 3.6%.

However, the obtained improvements vary significantly from lecture to lecture, and in lectures

2 and 6 our method slightly degrades performance, when compared to the original ASR transcripts. We attribute this to differences between the lectures and their supporting slides: some of the slides contained less text or a larger number of images, which our text extraction method is not able to process, and in other cases the lecturer deviated from the slides to discuss a topic not covered by these. In those cases, our method seems to have introduced a small number of errors by trying to combine the speech with unrelated slide words. An interesting possible line of future work would be to find ways to recover text and equations contained in images, in order to increase the amount of information for combination purposes.

3.7 Conclusions

In this chapter we proposed a generic approach for integrating multiple parallel streams of information. We combine sequences of words in the different streams (which can be of different types, such as speech or text) in order to improve speech recognition performance. The relations between the word sequences are encoded in the form of phrase tables, as are used in Statistical Machine Translation. The relations extracted from different pairs of streams, in the form of phrase pairs, are used to build an alignment across the different streams which is then used to rescore the streams containing speech.

This method is evaluated in two different scenarios: improving the recognition of simultaneously interpreted speech from the European Parliament Committees and using the information contained in slides used to support a lecture to improve the speech recognition of that lecture. We were able to achieve improvements in recognition accuracy in both scenarios. The magnitude of these improvements can optionally be improved, if one applies the OOL word and abbreviation recovery techniques that were described in this chapter.

In the future, we would like to extend this technique to further application areas. A possible application would be audio-visual speech recognition, where we would need to establish a mapping between sequences of visual features and words. Also, it would be interesting to apply this work to mutually assisted automatic speech recognition and handwriting recogni-

tion. In this case, we could, for instance, use the speech recognition output to inform better handwriting recognition about the same topic, and vice-versa.

4 Improved Rich Transcription across Multiple Streams

4.1 Introduction

In this Chapter, we consider the problem of enriching speech transcripts with meta-information, using the multistream alignments discussed in Chapter 3. We therefore discuss a method to place final marks in ASR output which takes advantage of the information given by an alignment of multiple parallel streams, and apply it to the simultaneous interpretation scenario of the European Parliament. Similarly, we propose a technique which capitalizes on the information available in these alignments, created from multiple parallel streams, in order to improve the detection of which of the words output by an ASR system are part of disfluent speech phenomena.

4.2 Improved Punctuation Recovery

In this section, we propose a method to recover punctuation that uses the information in the multistream alignments to do this more accurately. We do this by combining the probabilities given by a baseline classifier with the information that is extracted from the alignment, which constrains the possible sentence boundary locations.

4.2.1 Baseline systems

4.2.1.1 ASR and SMT systems description

We used four languages - English, Italian, Portuguese and Spanish - to develop our ASR and SMT systems. We used Audimus [63], a hybrid ANN-MLP WFST-based recognizer,

as the speech recognizer for this work. We trained 4-gram language models for each of the languages using the Europarl Parallel Corpus [47], and used our existing acoustic models and lexica for these four languages [65]. Phrase tables were created for the six possible language combinations (Portuguese-Spanish, Portuguese-English, Portuguese-Italian, Spanish-English, Spanish-Italian and English-Italian). We used the Moses toolkit [49] to train these phrase tables on the European Parliament Parallel data.

4.2.1.2 Punctuation System

The baseline punctuation system [5] consists of a maximum entropy classifier, which combines a number of features that are extracted from the region surrounding each potential sentence boundary (which corresponds to each word boundary).

These include word features, which capture information about which words or word bigrams are likely to occur close to a sentence boundary; speaker identity features, which use the information provided by a speaker clustering system to detect if the speaker has changed; part-of-speech (POS) features, which consider the tags assigned to words by a part-of-speech tagger; the segments produced by an acoustic segmenter, and the duration of the intervals between consecutive neighbouring words.

When recovering punctuation, most of these features are obtained from the automatically generated speech transcripts or from the output of the audio pre-processor module. The Portuguese and English punctuation systems were trained using manually annotated broadcast news data.

4.2.2 Proposed Method

The main insight behind the proposed method is that we can find an approximate correspondence between sentences in different streams. That is, we expect to find, for a sentence in one stream, a sentence containing equivalent information in each of the other streams. Of course, this may not always be true, but we expect it to be a reasonable approximation. For

example, in the case of simultaneous translation of speech, the interpreter may drop a part of a sentence due to being unable to keep up with the speaker, or they may split a sentence into several sentences. In the latter case, there would no longer be a one-to-one equivalence between sentences in the two streams, so in our method we assume this to be a relatively rare occurrence.

In light of this assumption, our method consists of minimizing the function f in Equation 4.1, where the binary vector $s_i = w_{i1} \dots w_{im}$ represents the segmentation for stream i, such that $w_{ij} = 1$ if there is a full stop before the j^{th} word of stream i, and $w_{ij} = 0$ otherwise, and p_{ij}, when available, is the probability that there is a full stop before the j^{th} word of stream i, given by the baseline classifier.

$$f(s_1 \dots s_n) = \alpha_1 \delta(s_1 \dots s_n) + \alpha_2 \tau(s_1 \dots s_n) + \alpha_3 \gamma(s_1 \dots s_n) + \alpha_4 \left(\sum_{w_{ij}=1} p_{ij} + \sum_{w_{ij}=0} (1 - p_{ij}) \right) \tag{4.1}$$

The function $\delta(s_1 \dots s_n)$ tries to incorporate the information that we obtained from the multistream alignment. It does so by computing the number of distinct conflicts in the sentence segmentation $s_1 \dots s_n$. Consider two phrase pairs, pp_1 and pp_2, which are in the streams s and t. Then pp_1 and pp_2 are said to be in conflict, if there is a sentence boundary between the corresponding phrases in one of the streams but not in the other. For example, suppose we have the two phrase pairs pp_1 = "eles tinham || they had" and pp_2 = "tentado || tried", in the English and Portuguese streams, and that, in the current segmentation, pp_1 and pp_2 are in different sentences in one of the streams but not in the other. Then the function $\delta(s_1 \dots s_n)$ penalizes this fact, trying to bring the phrase pairs to equivalent sentences in the two streams.

Also, the component $\tau(s_1 \dots s_n)$ assigns a penalty which is proportional to the duration of an interword pause (the difference between the start time of the current word and the end time of the previous word) to word boundaries that are not preceded by a full stop in the

current segmentation. The component $\gamma(s_1 \dots s_n)$ represents, for each stream, the n-gram score variation introduced by adding full stops at the locations defined by segmentation $s_1 \dots s_n$. For example, if the original sentence for stream i is "Thank you chair we will now proceed." and the suggested punctuation is "Thank you chair. We will now proceed.", then the contribution of stream i to $\gamma(s_1 \dots s_n)$ is the difference between the language model scores of the second and the first punctuations. Finally, the last component assigns a higher score to punctuations that agree with the baseline classifier (when its information is available).

To optimize function f, we start with some initial joint sentence segmentation $s_1 \dots s_n$, and perform an applicable local operation - a valid operation which improves the total punctuation score given by function f - in order to produce a new sentence segmentation. We then iteratively apply one of these operations until reaching a local minimum (none of the operations can be applied) or a predefined maximum number of iterations. We therefore jointly optimize the punctuations of the streams, in a hill-climbing manner. The two types of applicable operations are the following:

- Merging two or more sentences in the same stream - two or more consecutive sentences are merged into one, therefore implicitly removing the full stops between them.

- Splitting a sentence, in a given stream, into two sentences, where the possible splitting locations are the word boundaries inside that sentence. This operation corresponds to inserting a full stop in the corresponding punctuation.

All the instantiations of the above operations are sorted by decreasing order of Δf. If more than one operation decreasing f is available, then the one which decreases it the most is selected at each step.

The parameters $\alpha_i > 0$ of Equation 4.1 are manually selected to minimize SER, averaged over the resulting segmentations $s_1^* \dots s_n^*$, in a held-out development set.

4.2.3 Experimental evaluation

4.2.3.1 Experimental Setup

We collected and manually transcribed four speeches, in English, from the DEVE, ENVI, LEGAL and IMCO Commitees of the European Parliament, as well as their respective interpreted versions in three other languages (Italian, Portuguese and Spanish). Our development set consists of two other speeches, in the same four languages, that we also collected and transcribed from the European Parliament Comittees.

We automatically transcribed and generated ASR lattices for each of the speeches, and then executed the multistream combination algorithm of Section 3.3, in order to generate an alignment of the four streams. This alignment was subsequently used as input to the proposed method.

4.2.3.2 Results

To evaluate the impact of our proposed method on punctuation recovery performance, we computed four different performance metrics: SER [59], which represents the number of insertions, deletions and substitutions of punctuation marks divided by the total number of reference punctuation marks, precision (P), recall (R) and F-measure (F1). In order to be able to do this, we first align the automatic transcripts with the manual reference, using a dynamic-programming edit distance algorithm. Note that this alignment may be non-trivial due to the fact that there may be recognition errors close to the sentence boundaries, making it unclear to which hypothesis words we should assign the punctuation marks in the reference.

The values of these metrics for the proposed method were compared with our two baselines. The first baseline was our Speech / Non-Speech (SNS) component [64], which splits speech into segments passed to the speech recognizer, based mostly on speech activity; we considered these segments to be implicitly delimited by full stops. The second baseline was the punctuation and capitalization system described in Section 4.2.1.2; we add a full stop to the output wherever

	SNS				Baseline				Prop. method			
	P	R	F1	SER	P	R	F1	SER	P	R	F1	SER
PT	0.259	0.817	0.393	2.540	0.372	0.365	0.365	1.281	0.632	0.604	0.616	0.761
ES	0.209	0.762	0.323	3.259	-	-	-	-	0.618	0.575	0.587	0.819
EN	0.338	0.946	0.489	2.205	0.558	0.693	0.614	0.897	0.710	0.718	0.710	0.608
IT	0.323	0.828	0.455	2.172	-	-	-	-	0.557	0.609	0.574	0.943
All	0.283	0.838	0.415	2.544	0.465	0.529	0.490	1.089	0.629	0.626	0.622	0.783

Table 4.1: The different performance metrics that were evaluated, for each of the tested methods, across Portuguese (PT), Spanish (ES), English (EN) and Italian (IT), and averaged over the four speeches in the test set. The table entries corresponding to Spanish and Italian are not available, since we lacked training data to generate instances of the classifier for these two languages. Therefore, the values in the "All" row are not directly comparable between the Baseline method and the other techniques.

the probability generated by this system is greater than 0.5. This baseline was only available for the Portuguese and English languages. The output of the SNS component was also used to initialize the search algorithm described in Section 4.2.2. All the comparisons were carried out for each of the four languages considered (English, Italian, Portuguese and Spanish).

The main results are summarized in Table 4.1. Overall, the method that produced the poorest results was - unsurprisingly, since it was not designed for the purpose of punctuation - the SNS component. It had the highest average recall among all the three tested methods together with the highest SER, which suggests that it split speech into a very large number of sentences, creating many spurious insertions of full stops. Also, the proposed method has the best results across all of the languages that we considered. Compared with the baseline method in Portuguese, there is a 40% reduction in SER and a 68% improvement in F1, and in English there is a 32% reduction in SER and a 15% improvement in F1.

We observe that all of the three compared methods are more effective in the original speech (always given in English), across all the considered metrics, than in the interpreted versions. Not only do the interpreters often pause at locations that are unrelated to the sentence boundaries, they sometimes also produce larger numbers of disfluencies which may be confusing as to the location of sentence boundaries. Also, the spontaneous speech they produce is usually recognized at a higher word error rate, and this disrupts features that are based on word

	SER (no baseline prob.)	SER (with baseline prob.)
PT	0.780	0.761
ES	0.832	0.819
EN	0.652	0.608
IT	0.966	0.943
All	0.807	0.783

Table 4.2: Comparison between the average SER of the different languages. The factor that varies between the columns is whether or not the baseline classifier probabilities, for Portuguese and English, are used as features.

identities, such as language model scores.

It is also interesting to note that the proposed method performs slightly worse in the interpreted languages (Spanish and Italian) for which the probabilities from the baseline classifier are unavailable, when compared to Portuguese, which suggests that including these as features had a positive impact on the proposed method. In fact, by inspecting Table 4.2, we find that this is actually the case: the use of these probabilities improves SER about 2.4% absolute and, while the improvements of the greatest magnitude are for the English language, all the languages show improvements, even those (Spanish and Italian) for which the probabilities of the baseline classifier were not available.

It does not require any training data, apart from a small development set which is used to tune a number of parameters of the algorithm. We evaluated our method in a test set consisting of European Parliament Committee speeches, in four languages (English, Italian, Spanish and Portuguese). We obtained an average 37% improvement in SER, when compared with a maximum entropy baseline, considering the two languages (Portuguese and English) for which this baseline was available.

4.3 Disfluency Detection with multiple streams

The occurrence of disfluencies is usually not detected directly by ASR systems, apart from the presence of filled pauses, which many systems model as specific entries in their pronunciation dictionaries. In spontaneous and conversational speech, filler words and false starts may

greatly impair the performance of an ASR system, both by disrupting its language model context and by adding spurious words. Their presence also contributes to reduce the performance of natural language understanding tasks that depend on the accuracy of the automatically generated transcripts. In fact, disfluencies have been found to hamper parsing techniques [29], since in their presence sentences are no longer well-formed.

In order to mitigate this problem, several methods have been developed to automatically detect and, if possible, remove such disfluencies. This is specially important in spontaneous or conversational speech.

An important problem with a number of disfluency detection methods is that they assume that an error-free transcription of the speech is available. As a result, their performance degrades significantly when working with automatic speech transcripts which contain errors, which is the case in practice. Our technique aims to be robust to the presence of speech recognition errors, as these will be present in most realistic situations.

In order to improve disfluency detection, we will again make use of redundant information that may be available across multiple streams. From these data we may be able to infer, for example, that a given word sequence is a phrase fragment corresponding to an edit disfluency, or detect a filled pause surrounded by a pair of aligned words.

In the remainder of this section, we first propose a disfluency detection framework upon which we build a disfluency detection system for the Portuguese language. We then show how the performance of this system can be improved by drawing upon information present in multiple parallel streams, in the context of the European Parliament Committee data.

4.3.1 Baseline systems

4.3.1.1 Disfluency detection system for Portuguese

Deciding whether words are part of a disfluency is an imbalanced classification problem, since even in highly spontaneous speech only 5-15% of the words correspond to disfluencies. Classi-

```
...na realidade penso <enfim> [] que teremos <de da de> [de] refocar...
...verdade que <a sua> [as suas] formas de luta neste problema ficam...
...<mas %fp> [mas] esse erro impede <%fp> que se obtenham muitas das...
...<eu gostaria> [eu gostaria] que se pudesse chegar a um acordo nos...
...de acordo com a ideia <das das> [dos] deputados terem de analisar...
...<este digamos> [estas] perguntas essenciais que asseguram a nossa...
...a serem verdade <o o a os> os factos teremos de extrair destes as...
```

Figure 4.1: Examples of candidate disfluencies generated for the second stage classifier. The reparandum is inside the angled brackets, and the rectangular brackets contain the proposed repair of each candidate disfluency

fiers are often learned in a way that minimizes the number of incorrectly classified instances. Under these circumstances, simply assigning the majority class to every instance leads to a high accuracy, but does not separate instances of the minority class from those of the majority class, as intended. This results in a low recall in the detection of the underrepresented class. The most common approaches to this problem include under and over sampling techniques, such as Synthetic Minority Over-sampling Technique (SMOTE) [14], or cost-sensitive learners, which assign different costs to different types of errors, in order to ensure that the recall is increased.

In order to overcome the problem of label imbalance, our baseline system uses a problem-specific approach consisting of two classifiers applied in sequence. The first classifier assigns a label to each word, indicating whether that word is part of a disfluency, by examining only local context. In the second stage, we use a set of rules, described below, to find candidate word sequences that are likely to be part of disfluencies in the recognized text. Figure 4.1 shows some candidate disfluencies generated by this method.

These candidate disfluencies are then validated in a second classifier, which assigns a label to each proposed disfluency, indicating whether all the words in its deletable part actually correspond to disfluencies. The second stage classifier is effectively presented with a much more balanced problem, because these rule-generated candidates are much more likely to contain disfluencies than arbitrary text spans. The resulting system sets as disfluencies the words that are classified as disfluencies by either the first or the second classifier (or both).

This hopefully leads to an increase in recall without a decrease in precision.

Both of the classifiers use, where applicable, several sets of features:

- *Duration features*: we compute the duration of silent pauses before and after the reparandum and the repair, the duration of words around the interruption point, the duration of phones in the reparandum and the repair. We also measure the duration differential between identical pairs of phrases in the reparandum and repair, since in repetitions there is often a prolongation of the former.

- *Language model features*: these determine to what extent removing the deletable part of a potential disfluency impacts the sentence's likelihood under an n-gram language model. An increase in likelihood may signal the presence of a disfluency.

- *Sentence segmentation features*: we use a sentence segmentation classifier [5] to estimate the probability that a word boundary corresponds to a sentence boundary. If the two parts of a candidate disfluency effectively correspond to different sentences, then it cannot be a disfluency.

- *Lexical features*: these capture common word sequences that often signal the occurrence of a disfluency. The repetition of words, for instance, often indicates the occurrence of a disfluency (but not always, since they are sometimes used for emphasis). Several other patterns, such as the repetition of function words but in different number (e.g. plural rather than singular) and / or gender also indicate the occurrence of disfluencies, because they imply a revision of the message being transmitted by the speaker.

- *Lattice-based features*: due to the fact that there are misrecognized words in the automatic transcripts, we also include features measuring the confidence of each word in the disfluency, extracted from lattices generated by the speech recognizer. For each word in the disfluency, we compute its posterior probability given the lattice, as well as the number of alternative hypotheses, and its average acoustic score.

- *Speaker ID features*: if there is a speaker change within the scope of a potential disfluency, then that provides indication that this is probably not a disfluency.

- *First classifier probabilities*: the probabilities assigned to each word inside a candidate disfluency by the first stage classifier are also used as features in the second stage classifier.

We use a number of rules in order to generate candidate disfluencies for the second classifier, as exemplified in Figure 4.1. Among the generated disfluencies, we include all the repetitions of single words (unigrams), pairs of words (bigrams), sets of three words (trigrams) and of four words (quadrigrams). We allow other morphological variants of these words to be included, such as words that differ only in the gender. An arbitrary number of filled pauses are also allowed to appear inside the disfluency. Words known to be often used as fillers, such as `digamos, enfim` in Portuguese, are also allowed in the generated candidate list. Finally, we include pairs of word sequences which, while not being identical, are close enough in terms of an edit distance between their respective pronunciations. The idea behind this is to capture small word fragments which do not significantly change the disfluency structure. For instance, the disfluency `<os nossos que> os nossos` would be added to the candidate disfluency list.

As the first classifier, we used a Multilayer Perceptron and as the second classifier we used a decision tree, trained using the C4.5 algorithm with the Weka machine learning toolkit [33].

4.3.2 Proposed Method

In multiple parallel streams that are approximate translations of each other, disfluencies should in principle be randomly distributed. In other words, the locations where disfluencies occur in different streams should not be correlated. Therefore, unmatched words or phrases in an alignment of multiple streams are more likely to be part of disfluencies, such as filled pauses or edits, which were misrecognized as words. However, the existence of unmatched words or phrases may only indicate that a rephrasing which is not captured by the translation model has taken place. Therefore, we wish to incorporate this information, in a principled manner,

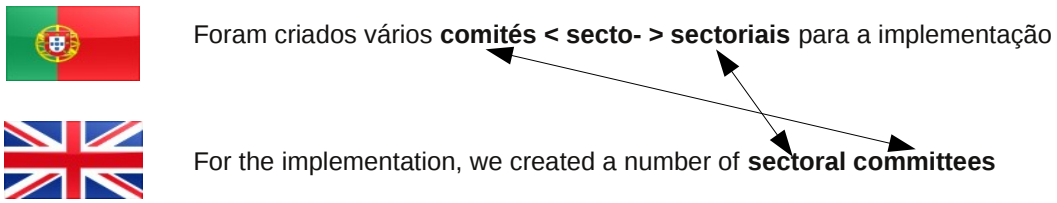

Figure 4.2: In this diagram, a word fragment "*secto-*" in a Portuguese stream occurs between two words which are aligned to consecutive words in an English stream. Because, in this case, the English stream contains the original speech, and the word fragment is not aligned to any words in other streams, this indicates that it is likely part of a disfluency.

within the framework of our disfluency detection classifier, in order to improve disfluency detection in the multistream framework.

Our approach consists of first running the baseline disfluency classifier, described in Section 4.3.1.1. From the output of this classifier, we obtain, for each word, the posterior probability that it is part of a disfluency. We then proceed by extracting alignment related features for each word in the transcription:

- The total score of the phrase pairs in the alignment which overlap with the current word, both partially and completely. The idea is that if a word is matched with words in one or several other streams, then that decreases the probability that this word is part of a disfluency in that stream. On the other hand, if the only existing matches with phrase pairs are partial, it may still be the case that part of the word is unmatched and therefore corresponds to a word fragment or a filled pause.

- If a word is unmatched, then we investigate whether the words or phrases that precede or follow it are matched to phrases in the same stream, and if those phrases are consecutive. If that is the case, as can be seen in Figure 4.2, then it is more likely that such a word is part of a disfluency.

- We also try to detect repetitions, or substitutions with similar phrases, by looking at whether there is more than one phrase that could be aligned with a given set of phrases

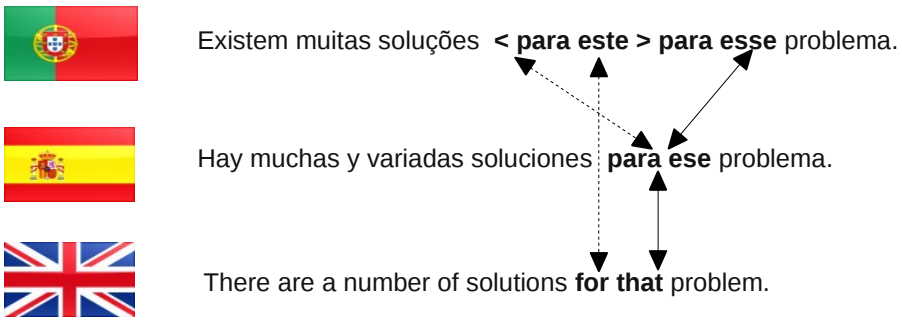

Figure 4.3: The diagram shows three parallel streams in three different languages. The solid arrows represent phrase pairs which are part of the alignment built for these streams, while the dashed arrows link together phrase pairs that were extracted during phrase table-lattice intersection, but which did not make it to the final alignment. A phrase pair which is on the end of only dashed arrows is probably part of an edited out word sequence.

in different streams. Figure 4.3 illustrates this situation. Note that the edited phrase "para este", as well as the following reparandum, in Portuguese, can both potentially be aligned to the same Spanish and English phrases. This is what allows us to determine that "<para este> para esse" is a potential repetition.

- The probability that this word is part of a disfluency, as obtained from the baseline disfluency classifier.

We train a logistic regression classifier, using the abovementioned features and a small amount of training data, in order to better separate words that are part of disfluencies from those which are not.

4.3.3 Experimental evaluation

4.3.3.1 Dataset

The dataset we used for training and testing the baseline classifier consists of about 40 hours of conversational speech, extracted from 40 Portuguese TV shows. The corpus was manually transcribed, including information about the location of disfluencies as well as the different

	TV Shows			European Parliament Committee (PT)		
	Words	Disf	Baseline WER	Words	Disf	Baseline WER
Training	227284	18355	39.60%	-	-	-
Development	24391	2449	36.40%	2888	142	31.62%
Testing	76923	5925	38.94%	11183	790	29.15%
Total	328598	26729	38.31%	14071	932	29.70%

Table 4.3: Statistics for the used corpora, in terms of the number of words (left column of each half), number of words corresponding to disfluencies (center column of each half) and ASR WER (right column of each half) for the training, development and test sets.

speakers in each turn. Most of these programs are political debates, and so there are many regions containing overlapping speech. Those segments without a predominant speaker or containing words which could not be transcribed were excluded from further processing. We randomly split the corpus into three parts: 70% of the programs were used for training each of the classifiers described in Section 4.3.1.1, 10% for evaluation and 20% for testing.

We force aligned the corpus to the reference transcription in order to obtain the time intervals at which disfluencies occur. Subsequently, we decoded each of the training, evaluation and test sets with our speech recognizer and labeled each word in the automatic transcripts as whether it is part of a disfluency or not. To do that, we marked a word in the automatic transcripts as a disfluency if its time interval overlapped with that of any word marked as a disfluency in the force aligned transcripts.

Table 4.3 summarizes the statistics of the corpus used.

For evaluating the proposed method, we have used the European Parliament Committee dataset described in Section 3.2.3.1. This dataset was manually annotated to include information about which words are part of disfluencies, and it was only split into development and testing subsets, since it is not used to train the baseline classifier. We followed a similar procedure as for the Portuguese TV show dataset previously described: we first force aligned the corpus to the disfluency annotated transcriptions to obtain disfluency time locations, and then transferred this information to the automatic transcripts generated for these speechs. Additionally, we automatically transcribed and generated ASR lattices for each of the speeches, and then executed the multistream combination algorithm of Section 3.3, in

	TV Shows		European Parliament Committee	
	Auto. transcript	Manual transcript	Auto. transcript	Manual transcript
Precision	79.8%	85.5%	74.0%	81.2%
Recall	46.5%	56.1%	45.9%	52.6%
F1	58.8%	67.7%	56.7%	63.8%
SER	65.3%	53.4%	70.2%	59.5%

Table 4.4: Results for classifying the words as whether or not they are part of disfluencies, both using the automatic transcripts and the reference. The left half of the table refers to the test set of the Portuguese TV corpus, whereas the right half refers to the European Parliament Commitee corpus

order to generate an alignment of the four streams. This alignment was subsequently used as input to the proposed method described in Section 4.3.2. Although we use information from four available streams (the Portuguese, Spanish, English and Italian versions of the speech) to construct this alignment, throughout the remainder of the current chapter all the results reported for disfluency detection refer exclusively to the Portuguese stream.

4.3.3.2 Results

First, we applied the baseline method described in the previous section to both corpora, and computed the precision, recall, F1 and SER (equivalent to the NIST error rate) on the test set, using both the reference and automatic transcripts. These results are displayed in Table 4.4.

We observe that the classification performance is better, as expected, when we have access to the reference transcripts, than when we rely on the ASR output, in the TV shows as well as in the European Parliament. However, the degradation in both F1 and SER is about 10% absolute, which indicates that our method is relatively robust to errors in the automatic speech transcripts. In both cases, the obtained precision is significantly higher than the recall. This can be justified by the fact that a considerable fraction of the disfluencies are very hard to identify, since in many cases the reparandum has no obvious similarity with the repair and can easily be mistaken for a fluent phrase.

In order to evaluate the impact of removing the disfluencies on the accuracy of our recognizer,

TV Shows			European Parliament Committee		
Baseline	Oracle	Proposed	Baseline	Oracle	Proposed
38.9%	32.4%	37.3%	29.2%	23.7%	28.0%

Table 4.5: Testing set WER before removing disfluencies, on the left column, compared with the WER using the oracle to determine disfluency locations, on the center column, and using the proposed method to generate disfluency locations, on the right column.

we took the automatically generated information concerning the disfluency locations and used it to perform a new ASR step, again considering both corpora. We modified our Viterbi based decoder in order to have it ignore the time regions corresponding to words that were identified as disfluencies. To do that, whenever a disfluency is encountered, the tokens corresponding to a word end that are active in the last frame before the disfluency are copied directly to the frame after the end of the disfluency, therefore effectively "ignoring" the acoustic observations in between. We compared this to the best achievable results, assuming that we had access to an oracle that provided the exact disfluency locations. This comparison can be seen in Table 4.5.

The absolute reduction in WER (1.6%) is, in this case, considerably smaller than the reduction obtained with the oracle version (6.5%). To a certain extent, this would be expected since the oracle is perfect, i.e, it corresponds to a precision and recall of 100%. However, some of the difference can also be attributed to the fact that, during decoding, some of the filled pauses are "absorbed" by words that begin or end with similar sounds. For instance, in the phrase "tentativa de %fp golpe de estado", the filled pause often has the same sound as the last phone of the word "de", and therefore the recognizer just interprets it as a lengthening of that word. Since the word as a whole is marked as a disfluency, then all of it, rather than just the filled pause, is removed. Another problem are word fragments which are of sufficiently small duration to be incorporated into the beginning or end of neighboring words. Both of these issues lead to deletion errors in the updated ASR output. They also interfere with the language model context of the recognizer, therefore reducing the improvement achieved by correctly detecting other disfluencies.

Also, the improvement in WER using the oracle (6.5%) is not as large as the proportion of

words marked as disfluencies in the testing set (7.7%). The main reason for this is connected with the words which absorb filled pauses or word fragments, but are otherwise recognized correctly by the ASR system, and therefore do not count as ASR errors.

Subsequently, we used the multistream alignments, with four different languages (Portuguese, English, Italian and Spanish), available for the European Parliament Committee corpus, to test our proposed method described in Section 4.3.2.

We obtained a recall of 53.4% and a precision of 73.6% (F1 = 61.9%, SER = 65.7%) in detecting which words are part of disfluencies. We observe that there was a considerable increase in recall, with the corresponding decrease in SER. From this, it can be seen that most of the recovered errors were deletions, i.e, sequences of words that had not previously been labeled as disfluencies. In particular, we were able to correctly identify many word fragments, as well as word sequences that are part of the reparandum of disfluencies, because these represented gaps in the alignment. These are often very hard to distinguish from fluent word sequences, since for example word fragments are often misrecognized as small words.

We also wanted to assess the impact of recovering from these disfluencies on the ASR WER, using the method that takes advantage of multiple parallel streams. There is an improvement in terms of WER of 0.3% absolute (1.1% relative), compared with the baseline disfluency detection method and of 1.5% absolute (5.1% relative) when we consider no disfluency detection.

It appears that we are closer to the WER improvement provided by an oracle than it would be expected if one extrapolates directly from the recall. As previously noted, some of the disfluencies have no direct impact on the WER, since they are short duration filled pauses which are confused with silences by the recognizer. These constitute most of the disfluencies that we were unable to detect.

4.4 Conclusions and Future Work

In this chapter, we have presented a method that can be used to combine multiple information streams, in order to improve automatic punctuation recovery in these streams. Our method extracts the information contained in an alignment that joins phrases in the various streams, and uses it to guide the better placement of end-of-sentence marks. We have shown that, in the Europarl Committee domain, it is possible to improve significantly over an existing maximum-entropy baseline, and to provide results even for languages for which such a baseline system is not available. In future work, we would like to extend this work to incorporate the recovery of different punctuation marks, such as the comma, question mark or exclamation mark. For example, recovery of the question mark, which tends to be easier in languages in which interrogatives are identified by cues such as subject-verb inversion, could be made significantly more effective by sharing this information across different languages.

We have also explored how to increase the performance of disfluency detection techniques, using the information available in multiple parallel streams. The alignments across phrases in different streams are taken as evidence, which helps us determine whether a given word should be considered part of a disfluency. In the European Parliament domain, it is possible to improve disfluency detection accuracy, when compared to a baseline disfluency classifier. In the future, we would like to expand the current work to propagate bidirectional feedback to the alignment building algorithm: the information gathered about the locations of disfluencies can be used to guide the alignment building process, since spurious phrase pairs containing a phrase that overlaps with a disfluent region no longer need to be considered. In a similar way, the information recovered by the punctuation recovery techniques developed in this chapter could be used to exclude cross-sentence phrase pairs from the alignment, which in turn could lead to better results.

5 Conclusions

In this thesis, we have presented a method for combining multiple parallel information streams, which can be applied to take advantage of redundant information in a number of Natural Language Processing tasks, such as simultaneous or consecutive interpretation, Computer Aided Translation, or recognition of lecture speech supported by slides. For the purposes of this thesis, we defined streams as symbol sequences, associated with time information that indicates where these symbol sequences occur, so that is possible to model the locality of redundancy across streams. Such streams may not be fully observable, containing a certain degree of uncertainty which reflects the imperfect methods used in their processing. Therefore, we proposed encoding the variability of each stream, as given, for example, by an automatic speech recognizer, as word lattices, and bridging the gap between different streams by using phrase tables, which map word sequences from one stream to another.

The main contribution of this thesis is, therefore, a technique to generate alignments of multiple parallel streams, which are then used to improve speech processing of those streams. These multistream alignments are non-conflicting sets of phrase pairs which indicate which symbol sequences in a given stream correspond to which symbol sequences in the others. We created an algorithm to generate an alignment out of several streams, which first performs a pairwise intersection of all the possible stream combinations, and then selects a non-conflicting subset of the generated phrase pairs, so as to maximize the expected quality of the alignment. This multistream alignment was used as a basis for improving speech recognition, punctuation recovery and disfluency detection in the speech streams considered, by extracting different sets of constraints from it. Our algorithm is also able to recover words that are not present in the generated lattices (*out-of-lattice words*), which is an important feature since lattices are finite. In the case of streams containing speech, that leads to increased robustness, enabling

one to deal with variability beyond that which can be handled by the recognizer.

In order to explore the applicability of the developed multistream combination algorithms, we applied them to two different domains: the simultaneously interpreted sessions of the European Parliament Committees and lectures in English supported by slides. In the first case, we integrated the original speeches in English with interpreted versions in Portuguese, Spanish, Italian and German and we were able to obtain improvements in recognition accuracy that increased with the number of languages used. For the latter application, the algorithms used to create the alignment were modified so as to work with the slide streams, which do not have explicit time information associated. We observed that this led to improvements in lecture recognition accuracy, both when compared with the baseline recognizer and with a recognizer using a language model which was trained on the slides texts. This result demonstrates that the information in the time alignments can be used to better constrain the slide words which match with a given speech segment.

We also considered how the method presented in this thesis can be applied to other speech processing tasks, such as rich transcription. In this way, we were able to improve the effectiveness of punctuation recovery and disfluency detection, when compared to baseline classifiers, by harnessing the information present in the multistream alignment. Full stop placement is improved by noting that sentence boundaries should be equivalent across streams, i.e., the phrases of a phrase pair should be in equivalent sentences in the respective streams. Similarly, disfluency detection is made more accurate by the observation that the disfluencies are independent events which are, in principle, not matched in streams other than the one in which they occur. These methods were applied to the simultaneously interpreted speech in the European Parliament committee speeches and we observed improvements, in terms of precision and recall, in both punctuation recovery and disfluency detection.

5.1 Future Work

The work in this thesis has led to a number of interesting open problems, which can be explored in future research.

Regarding the combination of multiple streams of information, it would be interesting to analyze the potential application of our method to sets of streams which have a looser coupling than the quasi-parallel relation that was defined in this thesis. Rather than considering that two streams are a translation of each other, we could extend the acceptable stream pairs to include streams that are simply referring to the same event. For instance, if several TV and radio stations are simultaneously broadcasting a sports event, in a different or the same language, they will be describing the same occurrences, using a set of expressions and linguistic constructions that are expected to overlap. Therefore, it can be expected that, using the methods that were presented in Chapter 3, it will be possible to obtain speech processing improvements by harnessing names and other expressions expected to be present across the different streams. However, additional techniques will probably be needed to avoid degrading performance significantly in the situations in which the streams contain unrelated information. In fact, to a certain extent this problem was observed in this thesis when combining speech with certain slides which content was more loosely coupled to the speech.

In this thesis, we focused on recovering information from speech streams. However, the streams used by the methods proposed in this thesis are not, in any way, restricted to speech or text and can be used whenever the variability or uncertainty present in the streams is representable in the form of a lattice. Therefore, an interesting direction for further research would be to use these techniques in other applications, with different stream types, which could be combined with speech streams. An example of this would be to enhance OCR with speech corrections, which would mutually benefit both tasks: not only would the latter be used to correct the output of the OCR system, but there would be a bidirectional information flow between the two streams, therefore improving the ASR system results as well.

Bibliography

[1] Kartik Audhkhasi, Panayiotis G Georgiou, and Shrikanth S Narayanan. Reliability-weighted acoustic model adaptation using crowd sourced transcriptions. In *INTERSPEECH*, pages 3045–3048, 2011.

[2] Kartik Audhkhasi, Panayiotis G Georgiou, and Shrikanth S Narayanan. Analyzing quality of crowd-sourced speech transcriptions of noisy audio for acoustic model adaptation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4137–4140. IEEE, 2012.

[3] N. Bach, M. Eck, P. Charoenpornsawat, T. Köhler, S. Stüker, T. Nguyen, R. Hsiao, A. Waibel, S. Vogel, T. Schultz, and A. W. Black. The CMU TransTac 2007 eyes-free and hands-free two-way speech-to-speech translation system. In *Proceedings of the IWSLT*, 2007.

[4] S. Bangalore, G. Bordel, and G. Riccardi. Computing consensus translation from multiple machine translation systems. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 351–354, 2001.

[5] F. Batista, H. Moniz, I. Trancoso, and N. J. Mamede. Bilingual Experiments on Automatic Recovery of Capitalization and Punctuation of Automatic Speech Transcripts. *IEEE Transactions on Audio, Speech and Language Processing*, pages 474–485, 2012.

[6] N. Bertoldi, R. Zens, and M. Federico. Speech translation by confusion network decoding. In *Proceedings of the ICASSP*, pages 1297–1300, Honululu, HI, USA, 2007.

[7] Nicola Bertoldi, Patrick Simianer, Mauro Cettolo, Katharina Wäschle, Marcello Federico, and Stefan Riezler. Online adaptation to post-edits for phrase-based statistical machine translation. *Machine Translation*, 28(3-4):309–339, 2014.

[8] K. Boakye, B. Trueba-Hornero, O. Vinyals, and G. Friedland. Overlapped speech detection for improved speaker diarization in multiparty meetings. In *Proceedings of the ICASSP*, pages 4353–4356, 2008.

[9] H. Boulard and N. Morgan. *Connectionist Speech Recognition - A Hybrid Approach*. Kluwer Academic Publishers, Massachussets, EUA, 1994.

[10] P. F. Brown, V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311, 1993.

[11] P. F. Brown, S. F. Chen, and S. A. Della Pietra. Automatic speech recognition in machine-aided translation. *Computer Speech and Language*, 8(3):177–187, 1994.

[12] Michael Buhrmester, Tracy Kwang, and Samuel D Gosling. Amazon's Mechanical Turk: a new source of inexpensive, yet high-quality, data? *Perspectives on psychological science*, 6(1):3–5, 2011.

[13] C. Callison-Burch and R. S. Flournoy. A program for automatically selecting the best output from multiple machine translation engines. In *Proceedings of MT Summit VIII*, 2001.

[14] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[15] L. Chen, L. Lamel, and J. L. Gauvain. Lightly supervised acoustic model training using consensus networks. In *Proceedings of the ICASSP*, 2004.

[16] S.F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393, 1999.

[17] Y. Chen, A. Eisele, C. Federmann, E. Hasler, M. Jellinghaus, and S. Theison. Multi-engine machine translation with an open-source decoder for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 193–196, 2007.

[18] D. Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the ACL*, Ann Arbor, Michigan, 2005.

[19] W.T. Chu and H.Y. Chen. Toward better retrieval and presentation by exploring cross-media correlations. *Multimedia systems*, 10(3):183–198, 2005.

[20] G. E. Dahl, D. Yu, L. Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech and Language Processing*, 20(1):30–42, 2012.

[21] S. B. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4), 1980.

[22] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1): 1–38, 1977.

[23] Y. Ding and M. Palmer. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of the ACL*, Ann Arbor, Michigan, 2005.

[24] B. J. Dorr. Interlingual Machine Translation: A Parameterized Approach. *Artificial Intelligence*, 63:429–492, 1993.

[25] A. Eisele, C. Federmann, H. Saint-Amand, M. Jellinghaus, T. Herrmann, and Y. Chen. Using Moses to integrate multiple rule-based machine translation engines into a hybrid system. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 179–182, 2008.

[26] Maxine Eskenazi, Gina-Anne Levow, Helen Meng, Gabriel Parent, and David Suendermann. *Crowdsourcing for Speech Processing: Applications to Data Collection, Transcription and Assessment.* John Wiley & Sons, 2013.

[27] J.G. Fiscus. A post-processing system to yield reduced word error rates: Recognizer Output Voting Error Reduction (ROVER). In *Proceedings of the ASRU*, Santa Barbara, USA, 1997.

[28] R. Frederking and S. Nirenburg. Three heads are better than one. In *Proceedings of the Fourth conference on Applied Natural Language Processing.* Association for Computational Linguistics, 1994.

[29] Fredrik Jørgensen. The Effects of Disfluency Detection in Parsing Spoken Language. In *Proceedings of the 16th Nordic Conference of Computational Linguistics NODALIDA-2007*, Tartu, Estonia, 2007.

[30] M. Galley, Jonathan G., K. Knight, D. Marcu, S. Deneefe, W. Wang, and I. Thayer. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the ACL*, 2006.

[31] K. Georgila, N. Wang, and J. Gratch. Cross-domain speech disfluency detection. In *Proceedings of the SIGdial Workshop on Discourse and Dialogue*, Tokyo, Japan, 2010.

[32] D. Graff. The 1996 broadcast news speech and language-model corpus. In *Proceedings of the 1997 DARPA Speech Recognition Workshop*, 1996.

[33] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1), 2009.

[34] J. H. Hansen, R. Huang, P. Mangalath, B. Zhou, M. Seadle, and J. R. Deller Jr. SPEECHFIND: spoken document retrieval for a national gallery of the spoken word. In *Proceedings of the Nordic Signal Processing Symposium (NORSIG)*, 2004.

[35] T. J. Hazen. Automatic alignment and error correction of human generated transcripts for long speech recordings. In *Proceedings of the Interspeech*, 2006.

[36] P. Hedelin and J. Skoglund. Vector quantization based on Gaussian mixture models. *IEEE Transactions on Speech and Audio Processing*, 8(4):385–401, July 2000.

[37] H. Hermansky. Perceptual linear predictive (PLP) analysis of speech. *Journal of the Acoustical Society of America*, 87(4):1738–1752, 1990.

[38] H. Hermansky and N. Morgan. RASTA processing of speech. *IEEE Transactions on Speech and Audio Processing*, 2(4):578–589, 1994.

[39] H. Hermansky, D. P. W. Ellis, and S. Sharma. Tandem connectionist feature extraction for conventional HMM systems. In *Proceedings of the ICASSP*, 2000.

[40] Matthias Honal and Tanja Schultz. Correction of disfluencies in spontaneous speech using a noisy-channel approach. In *Proceedings of Interspeech*, 2003.

[41] T. Izumitani, R. Mukai, and K. Kashino. A background music detection method based on robust feature extraction. In *Proceedings of the ICASSP*, pages 13–16, 2008.

[42] S. Jayaraman and A. Lavie. Multi-engine machine translation guided by explicit word matching. In *Proceedings of the ACL 2005 on Interactive poster and demonstration sessions*, pages 101–104, 2005.

[43] S. E. Johnson, P. Jourlin, G. L. Moore, K. S. Jones, and P. C. Woodland. The Cambridge University spoken document retrieval system. In *Proceedings of the ICASSP*, 1999.

[44] G. Jones and R. Edens. Automated alignment and annotation of audio-visual presentations. *Research and Advanced Technology for Digital Libraries*, pages 187–196, 2002.

[45] S. Khadivi and H. Ney. Integration of Speech Recognition and Machine Translation in Computer-Assisted Translation. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(8):1551–1564, 2008.

[46] B. E. D. Kingsbury. *Perceptually inspired signal processing strategies for robust speech recognition in reverberant environments.* PhD thesis, University of California, Berkeley, 1998.

[47] P. Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, 2005.

[48] P. Koehn, F. J. Och, and D. Marcu. Statistical Phrase-Based Translation. In *Proceedings of HLT-NAACL*, 2003.

[49] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the ACL*, 2007.

[50] J. Kominek, C. L. Bennett, and A. W. Black. Evaluating and correcting phoneme segmentation for unit selection synthesis. In *Proceedings of the Interspeech*, 2003.

[51] A. Lambourne, J. Hewitt, C. Lyon, and S. Warren. Speech-Based Real-Time Subtitling Services. *International Journal of Speech Technology*, 7(4):269–279, 2004. ISSN 1381-2416.

[52] L. Lamel, J. Gauvain, and G. Adda. Lightly supervised and unsupervised acoustic model training. *Computer Speech and Language*, 16:115–129, 2002.

[53] Alon Lavie, Michael Denkowski, and Chris Dyer. Learning from post-editing: Online model adaptation for statistical machine translation. In *EACL 2014*, 2014.

[54] E. Levin, S. Narayanan, R. Pieraccini, K. Biatov, E. Bocchieri, G. Di Fabbrizio, W. Eckert, S. Lee, A. Pokrovsky, M. Rahim, P. Ruscitti, and M. Walker. The ATT-Darpa Communicator Mixed-Initiative Spoken Dialog System. In *Proceedings of the ICSLP*, 2000.

[55] M. Liberman, J. Yuan, A. Stolcke, W. Wang, and V. Mitra. Using multiple versions of speech input in phone recognition. In *Proceedings of the ICASSP*, 2013.

[56] Y. Liu, E. Shriberg, A. Stolcke, and M. Harper. Comparing HMM, maximum entropy, and conditional random fields for disfluency detection. In *Proceedings of the Interspeech*, pages 3313–3316, Lisbon, Portugal, 2005.

[57] J. Lööf, C. Gollan, S. Hahn, G. Heigold, B. Hoffmeister, C. Plahl, D. Rybach, R. Schlüter, and H. Ney. The RWTH 2007 TC-STAR Evaluation System for European English and Spanish. In *Proceedings of Interspeech 2007*, pages 2145–2148, Antwerp, Belgium, 2007.

[58] A. Lopez. Statistical Machine Translation. *ACM Computing Surveys*, 3(8), 2008.

[59] J. Makhoul, F. Kubala, R. Schwartz, and R. Weischedel. Performance measures for information extraction. In *Proceedings of the DARPA Broadcast News Workshop*, pages 249–252, 1999.

[60] L. Mangu, E. Brill, and A. Stolcke. Finding consensus in speech recognition: word error minimization and other applications of confusion networks. *Computer Speech and Language*, 14(4):373–400, 2000.

[61] E. Matusov, N. Ueffing, and H. Ney. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *Proceedings of the EACL*, Trento, Italy, 2006.

[62] E. Matusov, D. Hillard, M. Magimai Doss, D. Hakkani-Tür, M. Ostendorf, and H. Ney. Improving speech translation with automatic boundary prediction. In *Proceedings of the Interspeech*, 2007.

[63] H. Meinedo and J. P. Neto. Automatic speech annotation and transcription in a broadcast news task. In *Proceedings of the ISCA Workshop on Multilingual Spoken Document Retrieval*, Macau, China, 2003.

[64] H. Meinedo and J. P. Neto. Audio segmentation, classification and clustering in a broadcast news task. In *Proceedings of ICASSP*, Hong Kong, China, 2003.

[65] H. Meinedo, A. Abad, T. Pellegrini, I. Trancoso, and J. Neto. The L2F Broadcast News Speech Recognition System. *Proceedings of FALA 2010, Vigo, Spain*, 2010.

[66] D. Mekhaldi. Multimodal document alignment: towards a fully-indexed multimedia archive. In *Proceedings of the Multimedia Information Retrieval Workshop, SIGIR*, Amsterdam, the Netherlands, 2007.

[67] H. Misra, H. Bourlard, and V. Tyagi. New Entropy Based Combination Rules In HMM/ANN Multi-Stream ASR. In *Proceedings of ICASSP*, Hong Kong, China, 2003.

[68] M. Mohri, F. C. Pereira, and M. Riley. Weighted Finite-State Transducers in Speech Recognition. *Computer Speech and Language*, 16(1):69–88, 2002.

[69] Günter Mühlberger, Johannes Zelger, and David Sagmeister. User-driven correction of OCR errors: combing crowdsourcing and information retrieval technology. In *Digital Access to Textual Cultural Heritage (DATeCH 2014)*, pages 53–56, Madrid, Spain, 2014.

[70] S. Nakamura, K. Markov, H. Nakaiwa, G. Kikui, H. Kawai, T. Jitsuhiro, J. Zhang, H. Yamamoto, E. Sumita, and S. Yamamoto. The ATR multilingual speech-to-speech translation system. *IEEE Transactions on Audio, Speech, and Language Processing*, 14 (2):365–376, 2006.

[71] T. Nomoto. Multi-engine machine translation with voted language model. In *Proceedings of the ACL*, Barcelona, Spain, 2004.

[72] D. Noonburg. Xpdf: A C++ library for accessing PDF, 2009. URL http://www.foolabs.com/xpdf/.

[73] Franz Josef Och and Hermann Ney. Statistical multi-source translation. In *Proceedings of the MT Summit VIII*, 2001.

[74] M. Paulik and A. Waibel. Extracting Clues from Human Interpreter Speech for Spoken Language Translation. In *Proceedings of ICASSP*, Las Vegas, USA, 2008.

[75] M. Paulik, S. Stüker, C. Fügen, T. Schultz, T. Schaaf, and A. Waibel. Speech Translation Enhanced Automatic Speech Recognition. In *Proceedings of the ASRU*, San Juan, Puerto Rico, 2005.

[76] M. Paulik, S. Rao, I. Lane, S. Vogel, and T. Schultz. Sentence Segmentation and Punctuation Recovery for Spoken Language Translation. In *Proceedings of ICASSP*, 2008.

[77] D. Picó, J. González, F. Casacuberta, D. Caseiro, and I. Trancoso. Finite-state transducer inference for a speech-input Portuguese-to-English machine translation system. In *Proceedings of Interspeech*, Lisbon, Portugal, 2005.

[78] P. Placeway and J. D. Lafferty. Cheating with imperfect transcripts. In *Proceedings of the ICSLP*, 1996.

[79] P. Placeway, S. Chen, M. Eskenazi, U. Jain, V. Parikh, B. Raj, M. Ravishankar, R. Rosenfeld, K. Seymore, M. Siegler, R. Stern, and E. Thayer. The 1996 Hub-4 Sphinx-3 System. In *DARPA Speech Recognition Workshop*, 1997.

[80] L. R. Rabiner. *A tutorial on Hidden Markov Models and selected applications in speech recognition*, pages 267–296. Morgan Kaufmann, San Francisco, USA, 2000.

[81] S. Rao, I. Lane, and T. Schultz. Improving spoken language translation by automatic disfluency removal: Evidence from conversational speech transcripts. In *Machine Translation Summit XI*, 2007.

[82] A. Raux, B. Langner, D. Bohus, A. W. Black, and M. Eskenazi. Lets go public! taking a spoken dialog system to the real world. In *Proceedings of Interspeech 2005*, Lisbon, Portugal, 2005.

[83] A. Reddy and R.C Rose. Integration of statistical models for dictation of document translations in a machine-aided human translation task. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(8):2015–2027, 2010.

[84] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.

[85] D. Rybach, S. Hahn, P. Lehnen, D. Nolden, M. Sundermeyer, Z. Tske, S. Wiesler, R. Schlter, and H. Ney. RASR - The RWTH Aachen University Open Source Speech Recognition Toolkit. In *Proceedings of ASRU*, Hawaii, USA, 2011.

[86] S. Saleem, S.-C. Jou, S. Vogel, and T. Schultz. Using Word Lattice Information for a Tighter Coupling in Speech Translation Systems. In *Proceedings of the ICSLP*, Jeju Island, Korea, Oct 2004.

[87] J. Schroeder, T. Cohn, and P. Koehn. Word lattices for multi-source translation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 719–727, 2009.

[88] E. Shriberg, R. Bates, and A. Stolcke. A prosody-only decision-tree model for disfluency detection. In *Proceedings of the Eurospeech*, Rhodes, Greece, 1997.

[89] M. Snover, B. Dorr, and R. Schwartz. A lexically-driven algorithm for disfluency detection. In *Proceedings of HLT-NAACL 2004*, Boston, Massachusetts, 2004.

[90] P. Somervuo, B. Chen, and Q. Zhu. Feature Transformations and Combinations for Improving ASR Performance. In *European Conference on Speech Communication and Technology*, 2003.

[91] M. Sperber. Efficient speech transcription through respeaking. Master's thesis, Karlsruhe Institute of Technology, 2012.

[92] A. Stolcke. SRILM - an extensible language modeling toolkit. In *Proceedings of the ICSLP*, 2002.

[93] A. Stolcke, E. Shriberg, D. Z. Hakkani-Tür, and G. Tür. Modeling the prosody of hidden events for improved word recognition. In *Proceedings of the Eurospeech*, Budapest, Hungary, 1999.

[94] A. Stolcke, H. Bratt, J. Butzberger, H. Franco, V. R. Rao Gadde, M. Plauché, C. Richey, E. Shriberg, K. Sönmez, F. Weng, and J. Zheng. The SRI March 2000 Hub-5 conversational speech transcription system. In *Proceedings of the NIST Speech Transcription Workshop*, 2000.

[95] R. Swaminathan, M. E. Thompson, S. Fong, A. Efrat, A. Amir, and K. Barnard. Improving and aligning speech with presentation slides. In *Proceedings of the International Conference on Pattern Recognition*, Istanbul, Turkey, 2010.

[96] S. E. Tranter and D. A. Reynolds. An overview of automatic speaker diarization systems. *IEEE Transactions on Audio, Speech and Language Processing*, 14(5):1557–1565, 2006.

[97] Unbabel. Unbabel. http://www.unbabel.com, 2015.

[98] A. Venkatamaran, A. Stolcke, W. Wang, D. Vergyri, V. R. R. Gadde, and J. Zheng. An efficient repair procedure for quick transcriptions. In *Proceedings of the ICSLP*, 2004.

[99] Keith Vertanen and David J. C. MacKay. Speech dasher: fast writing using speech and gaze. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 595–598. ACM, 2010.

[100] Ngoc Thang Vu, Franziska Kraus, and Tanja Schultz. Cross-language bootstrapping based on completely unsupervised training using multilingual A-stabil. In *Proceedings of the ICASSP*, pages 5000–5003, 2011.

[101] Mike Wald. Crowdsourcing correction of speech recognition captioning errors. In *W4A*. ACM, 2011.

[102] W. Wang, G. Tür, J. Zheng, and N. F. Ayan. Automatic disfluency removal for improving spoken language translation. In *Proceedings of ICASSP*, 2010.

[103] S. Young. Large vocabulary continuous speech recognition: A review. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, Snowbird, Utah, 1995.

[104] C. Zhai and J. D. Lafferty. A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. In *Proceedings of SIGIR*, pages 334–342, 2001.