Department of Computational Linguistics and Phonetics
Saarland University

MSc Thesis

Knowledge-based word lattice re-scoring in a dynamic context

Todd Shore
tshore@coli.uni-saarland.de

28th November 2011

Supervisors:
Prof. Dr Dietrich Klakow
Dipl.-Inf. Friedrich Faubel

Declaration

I hereby confirm that the thesis presented here is my own work, with all assistance acknowledged.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Saarbrücken, 15th November 2015

Signature:

Recent advances in automatic speech recognition (ASR) technology continue to be based heavily on data-driven methods, which means that the full benefits of such research are often not enjoyed in domains for which there is little training data, such as for air traffic control (ATC). Moreover, tractability is often an issue with these methods when conditioning for long-distance dependencies; This entails that many higher-level knowledge sources (KSs) such as situational knowledge cannot be easily utilised in classification.

In this thesis, it is shown that classification accuracy in ASR can be improved through lattice re-scoring by using dynamic, high-level situational knowledge about the context in which the ASR system is used. Although usage of higher-level knowledge has already been employed in lattice re-scoring, this thesis describes a novel approach for encoding dynamic, context-sensitive domain knowledge as dynamic declarative constraints which are then evaluated using a situational model of the given context, formulating knowledge-based re-scoring as a constraint satisfaction problem which is used to bias the original classification.

By using this method with a wide-coverage grammar, it was possible to reduce the word error rate (WER) by roughly 80% over the baseline for classification of spoken commands in an ATC simulation when using a combination of constraints encoding multiple KSs. Furthermore, this was accomplished with symbolic rather than data-driven methods, and so it can be used in any domain, regardless of the amount of training data available.

‘It is an old maxim of mine that when you have excluded the impossible, whatever remains, however improbable, must be the truth.’
— Sherlock Holmes, *The Adventure of the Beryl Coronet*
(Sir Arthur Conan Doyle, 1892)

Contents

List of Algorithms

List of Figures

List of Listings

List of Tables

Part I.

Introduction

1. Motivation

Despite the progress made in ASR in specific and probabilistic classification in general, these advances continue to be based heavily on data-driven methods, and the requirement for ever-increasing amounts of training data means that the best results often cannot be achieved for small domains with little training data and for which cross-domain adaptation may not work [47]. However, Moore [47] explained that human beings themselves prove that massive amounts of training data and computing time is not necessary for high-accuracy speech recognition, achieving 'near perfect' accuracy at a '-3dB signal-to-noise ratio' [47, p. 1177] largely independent of domain: By utilising an amount of redundancy present in discourse, humans can induce missing information so well that they are not only able to repair a noisy information signal but also to hyper-correct it, inferring information which was originally never actually communicated.

Moreover, many traditional methods in data-driven inference entail that the feature set used for conditioning is largely restricted to local features due to tractability concerns [44]: This means that long-distance and global dependencies cannot be effectively inferred, limiting the ability for such methods to model complex natural phenomena like human speech and dialogue.

Nevertheless, there has been much recent research into novel methods for utilising many different **knowledge sources** (KSs) [20] to improve accuracy, including incorporating such knowledge into lattice re-scoring methods for ASR [cf. 40, 61]. Similarly, such knowledge has also been successfully incorporated into inference through declarative methods such as integer linear programming [cf. 6, 7, 44, 51].

However, most of these methods are employed in the inference stage of classification, regardless of how exactly they incorporate knowledge into classification. Therefore, such methods still may not achieve optimal results in a domain where there is a severe lack of in-domain training data, for example, in the domain of air traffic control (due to the fact that proper annotation of ATC recordings is expensive and time-consuming due to noise and other issues). Moreover, the lack of training data for a given domain does not entail a lack of demand for high-accuracy classification in the domain. Furthermore, previously-unseen events must always be dealt with during probabilistic classification regardless of the size of the training data set available [43, pp. 198 sqq.].

Due to these reasons, it seems that the ability to utilise multiple dynamic higher-level KSs for knowledge-based reasoning would also be highly effective when applied at a stage after inference. Although some state-dependent systems such as dialogue systems employ a degree of dynamic knowledge [cf. 75], this remains largely unattempted in lattice re-scoring, which is already a ubiquitous method for improving classification accuracy in ASR.

2. Objective

This thesis describes an experiment to improve ASR classification accuracy in the domain of ATC. This is done by re-scoring the ASR lattice output using higher-level KSs specific to the domain: The two main sources of knowledge utilised are the knowledge of possible referents (i.e. aircraft) in a given ATC scenario and physical knowledge about next possible aircraft states given their current state. Knowledge from these KSs is encoded as declarative constraints in order to calculate a knowledge score [40] for a classification hypothesis which is dependent on the state of the ATC scenario at hand. This knowledge score based on dynamic situational knowledge is then combined with the probabilistic score of the acoustic and language models used in ASR to increase the accuracy in classifying utterances situated in a dynamic context.

Natural language is **situated** in context, being dependent on the situation in which a dialogue takes place [35, pp. 311-316]. Although such situational knowledge cannot be always easily incorporated into probabilistic models directly due to tractability issues, such information can be utilised to improve classification accuracy in non-probabilistic ways by using such knowledge to eliminate hypotheses which are implausible given the context in question, regardless of the probability of the hypothesis — in the words of Sherlock Holmes, '... when you have excluded the impossible, whatever remains, however improbable, must be the truth' [15].

This thesis is structured as follows: First, a review of previous related work is provided in chapter 3 on the following page. Then, chapter 4 on page 27 describes the methods used for obtaining a corpus of recorded ATC commands and information about the ATC scenario context in which they were given and the ASR system used for initial classification hypothesis scoring is expounded in chapter 5 on page 31. In chapter 6 on page 38, the re-scoring of the lattices output by the ASR system is explained and the metrics used for evaluating the effectiveness of re-scoring is described in chapter 7 on page 44. Chapter 8 on page 48 discusses the baseline classification accuracy for the ASR system, before re-scoring; The results after re-scoring are specified in chapter 9 on page 54. The interpretation of these results is detailed in chapter 10 on page 64 and chapter 10 on page 64 expounds the implications of these results for research in ASR, re-scoring techniques and probabilistic classification in general. The thesis conclusion is summarised in chapter 12 on page 75, with chapter 13 on page 76 detailing future work needed to adapt the methods used in this thesis into a production-quality prototype for practical use. Finally, the details of the ATC corpus are listed in appendix B on page 81 and those on the grammar and language models used for ASR classification in appendix C on page 84; The lattice search algorithm used for scoring is detailed in chapter D on page 95, and details on the baseline results are listed in appendix F on page 102 while details on the results after re-scoring are listed in appendix G on page 105.

3. Previous work

3.1. Air traffic control

Air traffic control (ATC) is a process by which aircraft in a given controlled airspace (e.g. near an airport) are directed by controllers (usually on the ground) in order to ensure safe and efficient travel through the airspace. This is principally done through maintaining separation of aircraft in the controlled airspace: In other words, aircraft must be guided to their destination while maintaining a safe distance from every other aircraft in the three-dimensional airspace, whether the destination is to land at the airport itself or to pass through the airspace in order to enter another one. This is typically done by recording **flight information** about individual aircraft (e.g. their position, speed and current goal(s)) and deciding an ideal pattern of air traffic for the airspace and how/ where each individual aircraft should fit into that pattern. Verbal (i.e. radio) commands are then issued to individual aircraft to set them on a course which fits in this pattern; A pilot is not allowed to change airspeed, altitude or heading without explicit clearance from ATC operations beforehand. However, fatigue is one of the biggest problems of these multi-modal input requirements when combined with the volume of even routine air traffic: This means that air traffic controllers must take frequent breaks in order to avoid fatigue-related performance problems, increasing the cost of ATC operations.

A large number of ATC scenarios are routine and so there exists a well-defined pattern template which can be used without requiring active problem-solving skills from the controller(s). However, there is a small number of low-frequency (typically unexpected) scenarios for which active problem-solving skills are essential [58, pp. 45–57]. Therefore, a large portion of a controller's energy is devoted to monitoring and directing situations which in fact do not utilise a controller's full cognitive abilities, causing fatigue and keeping him or her from focusing all of his cognitive resources on situations which do require all of them. Due to this fact, much effort in ATC-related research has been directed at developing automated systems to which a controller can choose to delegate routine functions, enabling him/her to direct more of his/her attention and cognitive resources to situations which require active problem-solving skills [58, p. 45].

3.1.1. ATC simulation

In an effort to improve the quality and cost of ATC training, the use of ATC simulators in the training of air traffic controllers has become widespread [58, p. 15]. These simulators generate a model of an airspace, in which information regarding aircraft is generated by the simulation, e.g. for visual display on a computer screen. This is in contrast to real-life ATC scenarios where such information is induced from sensor input (e.g. most principally

from radar). Although simulators can also facilitate training for and rehearsal of routine ATC scenarios, perhaps the main advantage of simulation-based training is the ability to safely and inexpensively generate specific situations which may have a low probability of occurring in real life yet for which controllers must be well-prepared.

Moreover, ATC simulation not only used for training but is also used for research: A simulation environment provides an environment for experimentation in which many variables can be more easily controlled than in real life [58, pp. 16–23]. Through experimentation, not only can controller behaviour be studied, but new technologies and controller interfaces (e.g. new workstation designs) can be tested in an effort to e.g. improve situational awareness and reduce controller fatigue.

3.1.2. Arrival management

One application of ATC research is into **arrival manager** (AMAN) systems, which assist air traffic controllers in managing the flow of arriving aircraft, i.e. aircraft which are scheduled to land in the airspace being controlled. Through the use of such systems, many duties which once were done manually can be automated; See figure 3.1 on the following page for an example of a graphical user interface (GUI) for such a system. The most notable feature of AMAN software for this experiment is the computerised management of aircraft flight progress: Tracking and predicting/planning current and future aircraft position, heading and altitude was once done by a combination of manually monitoring radar displays and using flight strips — paper indicators of aircraft flight progress which are maintained on a strip board representing all aircraft in an airspace. These systems are now generally computerised, allowing more efficient arrival management.

3.1.3. Automatic speech recognition in ATC

Automatic speech recognition (ASR) is another technology which has been used in many efforts to automate ATC operations: Through the automatic recognition of verbal commands issued to pilots, it is possible to render much information redundant which is traditionally recorded manually by the controller.

However, there are multiple problems which must be addressed in ASR-based ATC operations: Figure 3.2 on page 20 illustrates the control flow for the processing of controller input into ATC system output (e.g. a visual display). Each stage in this **noisy channel** is a potential source for errors, which are then propagated to the remaining sub-processes [60].

Non-relevant language

Controllers are trained to issue commands in a simple, regular manner with an ATC-specific **phraseology** standardised by the International Civil Aviation Organisation (ICAO) [2]. These standards also mandate that language which is not directly relevant to ATC be avoided. Such non-target language comprises not only fillers such as *uh* and *er* but also phatic expressions such as *hello* and *good morning* (see figure 3.2 on

Figure 3.1.: An example GUI for AMAN software used by air traffic controllers, in this case that of the 4D-CARMA RadarVision AMAN client [27]. Commands which are to be issued to aircraft appear on the advisory stack (labelled as a), which is positioned above an integrated flight information display (labelled as b), which displays radar data as well as other information related to arrival management.

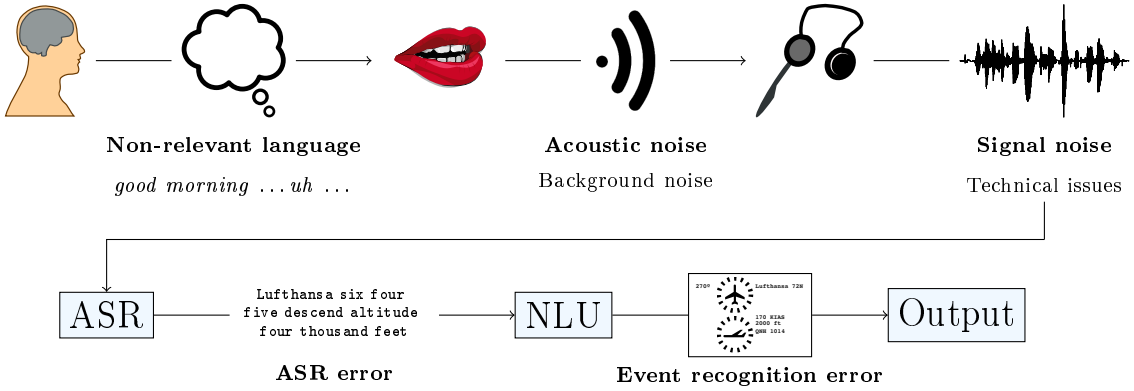
the following page) [42]. Finally, all ATC commands at an airport serving international flights must be issued in English.

However, in practice, controllers do not conform exactly to these standards [58, p. 20]. This non-relevant language must be part of the ASR acoustic and language models in order to avoid mis-recognition of non-relevant language as relevant language, thus increasing the size data set needed for training; This non-relevant data can be considered 'information noise' — information which is not relevant to the task at hand but can nevertheless cause errors in the recognition of relevant information.

Acoustic noise

Another problem is of acoustic noise, specifically background noise: Although environments in which ATC operations are performed are ideally quiet, maintaining an environment which is completely silent except for the issuing of ATC commands is infeasible. No matter how well background noise is filtered, it cannot be eliminated entirely. Therefore, an information stream composed of both relevant and non-relevant language will be

Figure 3.2.: The error propagation problem present in automatic event recognition for ATC operations, illustrating potential noise sources at each stage in the information channel for event recognition.



encoded as an acoustic signal with a significant portion of error-producing noise.

Signal noise

This informationally and acoustically noisy channel is then (imperfectly) converted into an electric signal by a microphone, which introduces the potential for further error. Furthermore, in a real-world scenario, information received by and from aircraft pilots is done so typically through very high frequency (VHF) radio; Errors in the VHF signal can be caused by e.g. electromagnetic atmospheric disturbances.

ASR errors

The noisy electromagnetic signal representing an acoustic signal (which is noisy in itself) may be mis-recognised by the ASR system, e.g. due to the model being imperfect. Although the model can be trained with data incorporating non-relevant language as well as background and electromagnetic noise, this subsequently increases the size of the domain being modelled, making recognition more difficult due to problems associated with such a relatively heterogeneous domain [cf. 25]. Furthermore, the amount of quality training data available in the domain of ATC is relatively small, especially when accommodating for the noise sources described above.

Event recognition errors

Finally, the (potentially incorrect) ASR label of the informationally, acoustically and electromagnetically noisy channel is interpreted by a natural language understanding (NLU) unit [cf. 32, pp. 858–861], recognising the ATC event denoted by the utterance. This recognised event is then e.g. converted to a visual representation for display or

transmitted to other systems. Although the event may be correctly recognised by a human listener, the amount of error (e.g. propagated from preceding processes) may cause the ATC event to be mis-recognised. In a sensitive task like ATC, a mis-recognition may have serious consequences, e.g. possibly leading to lack of aircraft separation (the main task in ATC operations) and therefore possibly leading to a dangerous in-flight incident which can disrupt normal airspace operations or, in extreme cases, to a mid-air collision.

Solutions

Although theses noise sources and subsequent causes of potential error can be mitigated, they cannot be eliminated entirely. As stated above, by adhering to the language specified by the ICAO [2] and maintaining a professional code of conduct within the ATC environment, non-relevant language and background noise can be minimised. Electromagnetic noise can be reduced through the use of high-quality recording and transmission equipment.

Reducing ASR errors is more complicated, however: The accuracy and performance of an ASR system is dependent on the size of the search space involved, just as many other problems related to information theory [cf. 43, pp. 61–78] — Generally, as the set of possible solutions increases, so does the perplexity of the system. However, there is little training data in the domain of ATC which can be used to create high-accuracy acoustic and language models for ASR, and this problem is exasperated by the need to incorporate the preceding noise sources as variables into the acoustic and language models used in ATC.

Nevertheless, much work has been done in improving ASR specifically for ATC, especially in e.g. cross-domain adaption to the domain of ATC [cf. 11]. More notably, however, is that KSs specific to ATC have been used in various ways to improve classification accuracy, such as separation patterns and general decision making in ATC [cf. 58].

3.1.4. Relevance

In the experiment done for this thesis, the accuracy of the automatic classification of ATC commands was improved through lattice re-scoring in the domain of a simulated ATC scenario, using AMAN software to generate and display the simulation to the recording participants, who read the commands prompted by the AMAN aloud for recording. The re-scoring method utilised the information from the simulation to incorporate situational knowledge specific to the ATC scenario in which the utterances were made, and thus KSs were used similar to the ones used by Schäfer [58].

3.2. Automatic speech recognition

Pattern recognition methods have been used to label spoken language since as early as the middle of the twentieth century, such as in the system developed by Davis, Biddulph and Balashek [12] for recognising numerals spoken in isolation. However, this first system

could only reliably recognise a single speaker, to whom the system had been configured [59, pp. 11 sqq.]. Since then, **automatic speech recognition** (ASR) has improved and matured as an industry, with many effective, robust ASR systems for many different platforms available both commercially [cf. 16] and in the open-source domain [cf. 56, 70, 73].

3.2.1. Current trends

With state-of-the-art methods, an ASR system can expect to have a WER of roughly 3% or lower and an SER of roughly 12% or lower when trained on a small, relatively homogeneous domain for which there is sufficient high-quality training data available [47]. However, systems designed for larger domains and more general use show significantly worse results [76, p. 131]. While ASR accuracy continues to improve, the amount training data required to achieve these improvements also continues to increase, leading to diminishing returns [47, pp. 1176–1179].

Due to these diminishing improvements, more research is beginning to incorporate information from more KSs [20, p. 218] than only acoustic information and the lexical/morpho-syntactic information represented in language models, including dialogue-state information and semantic/pragmatic information [cf. 20, 21, 34, 40, 47, 58, 61, 76, 77, 78].

3.2.2. Relevance

In the experiment done for this thesis, a state-of-the-art ASR system [73] was used for the classification of ATC commands, producing a set of classification hypotheses as word lattices (see section 5.5 on page 34) which was then re-scored (see section 3.4 on the following page) using dynamic knowledge-based reasoning to improve accuracy in a dynamic context, as described in chapter 6 on page 38.

3.3. Knowledge sources

The idea of using additional KSs for improving ASR accuracy has been actively pursued for some time. Although traditional ASR systems often use only acoustic and lexical, and syntactic knowledge to label input, a number of other KSs have been used for constraining ASR output, all of which have proved effective in reducing classification errors.

3.3.1. Dialogue state

Many systems use dialogue structure to constrain ASR output to utterances which are expected given the dialogue state [cf. 21, 34]. For example, Fink and Biermann [21] created a history-based expectation model from sequences of spoken utterances in dialogue and their associated actions; Similarly, Young [76] created a hierarchical dialogue tree of dialogue (sub-)goals which may be pursued and accomplished given the current dialogue state [76, p. 132].

3.3.2. Semantics

Many systems not only incorporate dialogue-state knowledge into their predictions but also incorporate constraints based on semantic inference. For example, not only did Young [76] created a frame language-based semantic model defining all domain objects and their attributes — for example, both *helicopter* and *ship* may be part of the model universe represented by the semantic model, but only *helicopter* has an *altitude* attribute. Using this semantic information, inferences about possible next commands can be made, constraining the next possible commands to those which are semantically and pragmatically valid. Similarly to as done by Young [76], Schäfer [58] utilised a reasoning system which uses semantic knowledge specific to ATC in predicting a controller's next command.

3.3.3. Situational knowledge

Some systems attempt to directly model the universe in which a dialogue takes place, making inferences about what can physically happen next: For example, Schäfer [58] used simulation data to predict what a given aircraft's future status will be (e.g. at what altitude and airspeed it will be flying at), and what possible conflicts may occur which must be resolved by the controller [58, pp. 59–60].

3.3.4. Relevance

In the experiment described in this thesis, KSs specific to ATC were used for improving in-domain ASR accuracy: The knowledge encoded in the re-scoring method used situational and semantic knowledge similar to some of that used by Schäfer [58], most notably using situational flight information about the aircraft in a given airspace.

3.4. Classification hypothesis re-scoring

In order to improve classification accuracy where tractability or lack of sufficient training data is a concern, one common practice is to **re-score** a set of initial classification hypotheses generated with a given probabilistic distribution using additional methods [49]. The main advantage of such a method is that additional knowledge can be incorporated into classification using previously-trained models: In this way, such knowledge can be applied even if it may be difficult to incorporate it directly into the original model(s) as features, e.g. if using an off-the-shelf classifier or for classification in domains for which it is difficult to acquire a large set of training data which incorporates all the features desired.

Re-scoring is often done by combining this original score with the score of a relatively more sophisticated model: For example, in the case of ASR, relatively simple acoustic and language models are often used for the initial decoding and are re-scored using more sophisticated versions [cf. 41]. Such scores can also be calculated using other methods, such as by optimising expected WER based on the posterior distribution [cf. 63]. Lastly,

it is possible to utilise additional KSs, which are not encoded by the features used by the original model(s), e.g. articulatory knowledge [cf. 40, 61].

3.4.1. Relevance

In the experiment done for this thesis, the accuracy of the ASR system described in section 3.2 on page 21 was improved through lattice re-scoring using KSs specific to ATC, as described in section 3.1 on page 17. However, unlike the methods described in the previous section, dynamic contextual knowledge was used for lattice re-scoring in addition to static knowledge such as that used by Li, Tsao and Lee [40].

3.5. Constraint programming

There is considerable recent interest in applying declarative programming methods to NLP tasks, most notably integer linear programming (ILP) [cf. 6, 7, 8, 22, 36, 44, 50] but also **constraint programming** (CP) [cf. 17, 24, 29]. Both of these paradigms are very similar in that they are used to find a solution to a problem subject to a set of constraints. Likewise, both are particularly useful for NLP in that they can easily encode non-local features which cannot be easily inferred from training data in a tractable way [8, 44]. Due to this property, they are useful for encoding knowledge from higher-level KSs into classification (see section 3.4 on the preceding page).

However, there are a number of differences between ILP and CP: Namely, in ILP, constraints may only be defined as either variable equalities or inequalities for integer values, while CP may incorporate mathematical constraints as well as logical constraints. Moreover, there is typically only one optimal solution for an ILP optimisation problem while there may be multiple valid solutions in a **constraint satisfaction problem** (CSP), in which at least one solution out of a set of multiple hypotheses is found which satisfies a set of constraints used in the definition of the problem [17, pp. 5 sqq., 24, 55, pp. 137–160].

A CSP is derived from a set of constraint evaluation functions $C \equiv \{c_i(\cdot)\}$, where the function $c_i \colon \mathcal{Z} \mapsto \mathbb{B}$ evaluates a vector of variables $z \in \mathcal{Z} \equiv \begin{bmatrix} z_1 & \cdots & z_n \end{bmatrix}$ against the semantic predicate encoded by the constraint [24]. Therefore, a solution to the CSP $\hat{z}$ is that which satisfies all the constraints $\sum_{c_i \in C} c_i(\hat{z}) = 0$.

3.5.1. Weighted constraints

Although constraints in a classic CSP are hard and thus a solution must satisfy all of them, it is possible to define a penalty **weight** p_i for violating each constraint c_i, therefore defining the most preferential solution $\hat{z}$ as that which minimises the product of the constraint functions and their weights $p^\intercal C(z)$ [13].

3.5.2. Dynamic constraints

Furthermore, a constraint evaluation function $c_i(\cdot)$ in a classic CSP is static: An input z will always evaluate to the same value $\mathbb{B}$ regardless of context. Nevertheless, it is possible to define **dynamic constraints** which encode context-sensitive predicates [cf. 14, 68]. In this way, it can be possible to evaluate knowledge which may change over time as a function $f_C(x, z)$ which is not only dependent on the solution itself z but also the context in which the solution is required x.

3.5.3. Relevance

In the experiment described in this thesis, lattice re-scoring was performed by defining a weighted CSP using declarative constraints which encode dynamic situational knowledge specific to ATC (see section 3.1 on page 17); This CSP was used during lattice re-scoring to bias the classification score of the ASR system used in the experiment in order to improve the accuracy of classification in the given context. Moreover, it not only utilises static knowledge such as done by e.g. Li, Tsao and Lee [40] but also dynamic contextual knowledge, and is therefore sensitive to the context in which the utterance to be classified was made.

Part II.

Methods

4. Corpus

For the experiment, a corpus of spoken ATC commands in the context of a dynamic ATC simulation was recorded, from which ASR recognition lattices were generated as output from an ASR system using the recordings as input (see section 5.5 on page 34). These lattices were used then re-scored using contextual knowledge (see chapter 6 on page 38).

4.1. Participants

Sixteen different participants recorded a total of 1,107 ATC commands, with a mean 69.1875 commands per participant; None of the participants had previous experience with ATC operations. Self-identified by native language, eight of the participants were German speakers, three were (American/Canadian) English speakers, and there were two Greek, one Malayalam, one Romanian and one Russian speaker. Twelve of participants were male and four were female. All of the participants were employees or students at Saarland University. The number of utterances attributed to each participant and to each native language and gender are listed in table B.1 on page 81.

4.2. Materials

In order to create a context in which to give ATC commands for recording, arrival manager (AMAN) software was used to prompt the participants to give pre-defined commands at pre-defined times in an ATC simulation. The same simulation was used for each participant, and, in addition to audio recording of the ATC commands, the time of each utterance was recorded. The AMAN logs simulation information at timed intervals (e.g. radar data indicating aircraft airspeed, heading, etc.); Recording the time of each utterance along logging such situational information allows the context in which each utterance is said to be inferred.

4.2.1. Software

The AMAN software used for simulation in the experiment was the 4-Dimensional Co-operative Arrival Manager (4D-CARMA) [28]. The simulation generated a dynamic advisory stack of commands which were to be read aloud within a certain time frame (generally within 30 seconds of the command being pushed into the advisory stack). The simulation information provided by the 4D-CARMA server was displayed via the 4D-CARMA RadarVision* client graphical user interface (GUI) [27]. The software displays

*RadarVision version 10.54 was used in this experiment.

an **advisory stack** visually in a moveable window on top of a display showing radar data along with other aircraft flight information (see figure 3.1 on page 19): Each item in the advisory stack represents a command an aircraft in the airspace is to execute.

Due to the fact that the AMAN software was used for recording ATC commands and data about the situational context in which they were issued, the procedure followed by the recording participants for using the system was different from that in true ATC operations [cf. 2, 27]: The participants were prompted to read the commands aloud as they were generated by the 4D-CARMA simulator and displayed on the advisory stack of the RadarVision GUI.

The advisory stack information as used for prompting ATC commands is defined as follows: The callsign of the aircraft is displayed (e.g. `AF418` (read as *Air France four one eight*) and `QFA6` (read as *Qantas six*) in figure 3.1 on page 19) along with the type of action the aircraft is to execute (e.g. `Descent` ≅ *descend* and `Turn Left` ≅ *turn left*) a value for the action (e.g. `FL70` ≅ *flight level seven zero* and `25R` ≅ *heading two five zero* — the heading the aircraft must turn toward *runway two five right*[†]). Lastly, a time is indicated in seconds which denotes the end of the time envelope in which the command should be issued (e.g. `+10` and `+19`). For example, the two commands on the advisory stack in figure 3.1 on page 19 would be read aloud as follows:

`+10 AF418 Descent FL70` *Air France four one eight descend flight level seven zero* (the command must be issued within 10 seconds)

`+19 QFA6 Turn Left 25R` *Qantas six turn left heading two five zero* (the command must be issued within 19 seconds)

4.2.2. Hardware

The recordings were made using a Dell Optiplex desktop PC with an Intel Core 2 Quad 2.83GHz CPU and 8GB of RAM running OpenSUSE version 11.3 AMD64 through a Sennheiser PC320 noise-cancelling headset. Recording was controlled through a push-to-talk interface in which hitting the enter key on a keyboard toggled recording on and off. The same PC was simultaneously used to run the 4D-CARMA server.

The RadarVision simulation software was run on a Lenovo Thinkpad X2005 notebook PC with an Intel Core Duo 1.86GHz CPU and 2GB of RAM running Microsoft Windows XP Professional 2002 SP3, displayed on a Dell 24" LCD monitor at 1920×1200 resolution. The RadarVision client was connected to the 4D-CARMA server by networking the two PCs in a two-PC LAN using a Netgear G5105 router.

[†]According to proper ATC procedure, a command to turn toward a runway for landing is issued as Callsign *turn* Direction *heading* Heading, *cleared runway* Runway, e.g. *Qantas six turn left heading two five zero, cleared runway two five right* [2]. However, in this experiment, the command is abridged to Callsign *turn* Direction *heading* Heading in order to simplify both the language and situation model.

4.2.3. Test simulation details

In this experiment, the AMAN simulation airspace featured in total 31 unique aircraft to which commands could be issued. However, at any one time there was a maximum of 11 and a mean of 8.76 aircraft in the air to which commands could be issued.

The duration of the entire simulation is approximately one hour, but the participants were required neither to say every command in the simulation (they were instructed to prefer reading out a smaller number of commands correctly over reading out a larger number of commands incorrectly) nor to continue recording until the end of the simulation. Likewise, the participants practiced with the simulation and recording set-up for a varying period of time before recording: Therefore, a varying amount of data was discarded from the start of each session. The maximum recording session length was 54:07 minutes of simulation time in duration and the mean recording session length was 39:49.5 minutes of simulation time. The total recording time was 10:37:12 hours of simulation time. For details on individual recording sessions, see table B.3 on page 82.

In addition to discarding the initial 'practice phase' of each session, the recordings were reviewed manually and all utterances were discarded which either were deemed to not comply to the ICAO phraseology or did not match the command prompted by the advisory stack at the given time.

4.2.4. Command format

The ATC commands which were read by the participants (in English) adhere to the ICAO standard phraseology as described in section 3.1.3 on page 18. Each command consists of an aircraft callsign (e.g. *Air France four one eight* ≅ `AF418`) followed by a goal action to execute (e.g. *descend* ≅ `Descent`) and a goal value to achieve during that action (e.g. *flight level seven zero* ≅ `FL70`). The ATC system grammar covers exactly such utterances and no more (see section 5.3 on page 32 for details).

4.2.5. Callsigns

Each command consists first of an aircraft **callsign** (e.g. *Air France four one eight* ≅ `AF418`), a sequence of words which unambiguously refers to a single aircraft in the airspace. Each callsign comprises an **airline designator**, e.g. *Air France* in the case of *Air France four one eight* ≅ `AF418` (i.e. denoted by `AF` in `AF[0-9]+`) or *Qantas* in *Qantas six* ≅ `QFA6` (i.e. denoted by `QFA` in `QFA[0-9]+`). Furthermore, each callsign has a **flight number** following the airline designator: This is a numeric expression of one to four numbers[‡] which denotes the individual flight, e.g. *four one eight* in the case of *Air France four one eight* ≅ `AF418` (i.e. denoted by `418` in `[A-Z]+418`).

[‡]A flight number may in fact be an alphanumeric of arbitrary length rather than comprising strictly one to four numerals (e.g. *bravo two three eight* ≅ `B238`) [1, p. 4-2-7]. However, in this experiment, flight numbers are restricted to numeric expressions of one to four numerals.

4.2.6. Command goals

Following an aircraft callsign is the **command goal**, a description of the action the given aircraft is to execute, e.g. *descend flight level seven zero* ≅ `Descent FL70` (i.e. the aircraft is to descend to flight level 70).[§] A command goal is composed of a **goal action**, such as *descend* ≅ `Descent` or *reduce speed* ≅ `Reduce`, and a following **goal value** which the aircraft is to achieve (e.g. *flight level seven zero* ≅ `FL70`).

4.2.7. Pronunciation

The participants were instructed to say the ATC commands clearly while still maintaining a 'normal' tone, volume and rate of speech and to minimise any non-relevant language. In addition, in accordance with ICAO standards, the participants were instructed to pronounce `3` ≅ *three* as *tree* /ˈtɹiː/ and `9` ≅ *nine* as *niner* /ˈnaɪ.nə(ɹ)/.

4.3. Procedure

The 4D-CARMA/recording server PC and RadarVision client PC were both set up on a computer desk in a room approximately 14m^2 in size. The participants read the ATC commands aloud as they appeared on the RadarVision advisory stack display. Each utterance was recorded in isolation (i.e. the recording was started and stopped once for each utterance): The participants were instructed to start recording for each command by pressing the enter key on the keyboard of the server PC about a second before the utterance, and to press enter again about a second after the end of the utterance in order to stop recording.

The participants were instructed in the command format and were told to focus on reading the commands out correctly over reading all the commands out, i.e. to favour fewer, more correct utterances over more, less correct utterances. They were allowed to practice with the system until they were ready to start the recording; all recordings during the practice session were later discarded. The recording session was then stopped after either one hour of simulation time (the approximate length of the entire simulation) or when the participant wished to stop.

[§] A **flight level** is a standard nominal altitude in hundreds of feet, calculated using international standard air pressure rather than the actual local air pressure.

5. ASR classification

The corpus recordings were used as input for an ASR system to output 2,214 weighted phone-to-word lattices, each representing the ASR network for an input x: Two different grammars were used (one with a relatively low perplexity and the other with a relatively high perplexity) to classify each of the 1,107 utterances, resulting in 1,107 lattices per grammar and 2,214 weighted phone-to-word lattices in total (see section 5.5 on page 34 for more details).

The ASR system used is based on *Millennium*, a set of dynamically-linkable shared-object libraries accessible through the Python interpreter [52, 73, pp. 497 sqq.]. This system extracts a time series of acoustic feature vectors from an input signal x and uses this as input to a series of cascaded context-sensitive HMMs in order to create a phone-to-word lattice represented as a finite state transducer [41, 57]. This lattice represents the classification score $f_\Phi(x, y)$ for a word label hypothesis y.

5.1. Definition

Classification using *Millennium* (as with most ASR systems) is formally a **linear classification** task [5, pp. 605–652]; The probabilistic model used for linear classification is composed of disjoint classes with discrete decision boundaries between them [5, p. 179], e.g. as in Hidden Markov Models (HMMs) [cf. 30, 47, 59]. Such a **linear model** is defined as a set of feature functions $\Phi \equiv \{\phi_i(\cdot)\}$ (where $\phi_i \colon \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$) with a corresponding weight vector w. According to such a model, the conditional probability $P(y \mid x)$ of a hypothesis y (typically a sequence of word labels) for an acoustic signal x is equivalent to the linear combination of these feature and weight vectors [7, p. 34, 63, p. 163]:

$$\begin{aligned} P(y \mid x) &\equiv f_\Phi(x, y) \\ &\equiv w^\intercal \Phi(x, y) \\ \Phi &\equiv \{\phi_i(\cdot)\} \qquad \text{where } \phi_i \colon \quad \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R} \end{aligned} \tag{5.1.1}$$

The advantage of linear methods over non-linear ones is their tractability and many tasks in NLP (including ASR) can be formulated as linearly-separable problems. However, non-local features cannot be easily incorporated into such models while maintaining tractability. Nevertheless, this linearity also means it is possible to re-score an input-output pair $\langle x, y \rangle$ by simply adding a constant term $f'_\Phi(x, y) \equiv f_\Phi(x, y) + p$; In this way, the constraint-based re-scoring function $f_C(x, y)$ can be easily incorporated (see chapter 6 on page 38 for details).

5.2. Vocabulary

The vocabulary Γ used by the two grammars is composed of 78 symbols, including ϵ. These symbols (with the exception of ϵ) primarily represent word labels corresponding to natural-language words such as `descend` $\cong$ *descend* and `air_france` $\cong$ *Air France*, meta-linguistic information such as `__pause__`. However, the vocabulary also includes a number of symbols representing various semantic concepts related to ATC, e.g. the XML tag set `<airspeed>` denotes an AIRSPEED concept, with the concept value represented by the word labels between the tags, e.g. `<airspeed> two two zero </airspeed>` $\cong$ 220 (knots) airspeed (see section 4.2.4 on page 29 for further details); These semantic concept labels are used to facilitate concept extraction for template-based semantic evaluation [cf. 71] during knowledge-based re-scoring (see section 6.2 on page 38 and appendix E on page 100). These semantic concept labels are omitted when measuring classification accuracy due to the face that they do not represent natural-language words (see chapter 7 on page 44).

Therefore, the vocabulary of output labels Γ comprises two sub-sets, of labels which correspond to natural-language words Γ^{word} and of labels which do not $\Gamma^{\overline{\text{word}}}$ ($\Gamma^{\overline{\text{word}}} \equiv \Gamma \setminus \Gamma^{\text{word}}$). The set of non-word labels $\Gamma^{\overline{\text{word}}}$ includes these concept symbols as well as other non-word labels such as ϵ and `__pause__`.

5.3. Grammars

The grammar used for training the language model used in ASR classification is a context-free grammar (CFG) described in the Nuance grammar format [48]. Two different grammars were used to train two different language models, each of which were then used to classify each utterance in the corpus: One grammar is relatively restricted with relatively narrow coverage and low perplexity, and the other is relatively unrestricted, with relatively broad coverage and high perplexity.

5.3.1. Restricted grammar

The grammar which was used to create one set of lattices is relatively restricted: The grammar covers only 35 unique callsigns — the 31 callsigns in the simulation plus four which were not used in the simulation (see section C.3 on page 88 for further details).

Perplexity

The restricted grammar has a relatively low perplexity of $6.19315214 \cdot 10^{17}$ when deriving perplexity from Shannon entropy (instead of cross entropy, as is often done) [43, p. 78] and using the maximum likelihood estimate (MLE) [43, pp. 197 sqq.] as the probability of a sentence $P(s \in S_{\text{restricted}})$ from the set of all unique sentences able to be produced by the restricted grammar $S_{\text{restricted}}$: See section C.1 on page 84 for further details on this derivation of grammar perplexity.

Purpose

The results of tests using this grammar represent the accuracy of the ASR system using the most-constrained grammar possible for the given domain at hand (i.e. for the single given ATC simulation in which the utterances were made). The accuracy of the system using this narrow-coverage system is the ideal goal during the re-scoring experiments: Given a wider-coverage system, the ideal goal of the re-scoring experiment is to achieve accuracy which is comparable to or better than the accuracy of the system using the narrow-coverage, restricted grammar.

5.3.2. Unrestricted grammar

The grammar which was used to create the second set of lattices is relatively unrestricted, covering callsigns comprising any combination of airline designator and flight number and numeric expressions comprising any combination of one to four numerals. Therefore, the grammar covers 244,220 unique callsign label sequences representing 122,210 unique callsigns — with 11 airline designators · (10 one-digit flight numbers + 100 two-digit flight numbers + 1,000 three-digit numbers + 10,000 four-digit flight numbers) · 2 for one variant with `__pause__` (e.g. `<callsign> <airline> adria </airline> __pause__ <flightnumber> two three </flightnumber> </callsign>`) and one without (e.g. `<callsign> <airline> adria </airline> <flightnumber> two three </flightnumber> </callsign>`) (see section C.4 on page 92 for further details).

Perplexity

The perplexity of the unrestricted grammar is $8.571322368 \cdot 10^{20}$, which is relatively high: This value is approximately 1,383 times that of the restricted grammar (see section 5.3.1 on the preceding page). Moreover, as stated previously, this difference in perplexity is attributed only to the restriction of callsigns in the restricted grammar to 35 pre-defined unique callsigns versus the wide coverage of 244,220 possible unique label sequences (which represent 122,110 unique callsigns) provided by the unrestricted grammar: This suggests that accuracy can be greatly improved simply by constraining the set of recognisable callsigns to those which represent valid aircraft in the simulation, as described in detail in section 6.2.1 on page 39.

Purpose

The results of tests using this grammar without any re-scoring represent the accuracy of the system with a wider-coverage grammar which would be used in a production system, one which is able to recognise most syntactically well-formed ATC commands and is constrained only by the vocabulary used (see the previous sub-section on vocabulary): In the re-scoring tests, the effectiveness of the re-scoring methods used is measured as the difference of the accuracy before and after re-scoring.

5.4. Acoustic training data

The acoustic model for the ASR system was trained using an amalgamation of several acoustic corpora of both spontaneous and scripted speech, including transcribed academic lectures, conferences and meetings [cf. 37] and news broadcasts [cf. 23]. Phonetic representations of the recording transcriptions were obtained by using the CMU Pronouncing Dictionary [38].

At the time of the experiment, there was no corpus of recordings dedicated specifically to ATC which were available and which had the contextual data required for the experiments at hand (see chapter 6 on page 38).

5.5. Word/phone lattices

Lattices are used in this experiment (as well as in many others) for representing ASR classification scores where multiple hypotheses and their scores are to be output in a single decoding pass [18, p. 1013]. For each acoustic signal input x, the ASR system outputs a phone-to-word lattice $\mathcal{L}(x) \mapsto \langle V, E \rangle$ (herein referred to simply as a **phone lattice**), which is a **connected directed graph** representing the ASR network, where V is a set of graph vertices and E is a set of directed edges $e \in E \equiv \langle s, e \rangle$ from one state $s \in V$ to another $e \in V$ [3]. From this lattice, all classification hypotheses and their corresponding scores can be calculated.

5.5.1. Definition

A phone lattice is formally a **weighted finite-state transducer** (WFST) over a semiring $\mathbb{K}^*$, receiving as input a sequence of symbols $\sigma_1 \dots \sigma_n$ from an input vocabulary Σ and outputting a corresponding sequence of symbols $\gamma_1 \dots \gamma_n$ from an output vocabulary Γ [18, 45]:

$$\begin{aligned} &\mathcal{T} \equiv \langle \Sigma, \Gamma, Q, I, F, \delta, \lambda, \rho \rangle && \text{where } I, F \subseteq Q \\ &\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \times \mathbb{K} \times Q \\ &\lambda \colon I \mapsto \mathbb{K} \\ &\rho \colon F \mapsto \mathbb{K} \end{aligned} \tag{5.5.1}$$

where Q is a set of states, $I \subseteq Q$ is a set of initial states (i.e. states with no incoming transitions) and $F \subseteq Q$ is a set of final states (i.e. states with no outgoing transitions). δ is a set of weighted transitions, each of which is a quintuple $\langle s, \sigma, \gamma, w, e \rangle$ where $s \in Q$ is the transition origin state, $e \in Q$ is the transition goal state, σ is the input symbol, γ is the output symbol and $w \in \mathbb{K}$ is the weight of the transition. Finally, for each initial state $i \in I$ there is a corresponding weight function $\lambda(i) \mapsto w \in \mathbb{K}$ and for each final state $f \in F$ there is a weight function $\rho(f) \mapsto w \in \mathbb{K}$.

*In the case of this experiment, the semiring used is the log-probability semiring $\mathbb{K} \equiv \langle \mathbb{R}^+ \bigoplus_{\log}, +, 0, 1 \rangle$ [74, p. 242].

5.5.2. Transition weights

In the case of ASR, a phone lattice $\mathcal{L}^{\text{phone}}(x)$ is generated for a time series of acoustic features x, denoting transition probabilities from multiple cascaded context-sensitive Hidden Markov Models used for classification by the ASR system [3, pp. 565 sqq., 30, 74]. From this lattice, the probability of a sequence of output word symbols $y \equiv \langle \gamma_1 \dots \gamma_n \rangle$ given a sequence of phone and/or word input symbols $\Sigma' \equiv \langle \sigma_1 \dots \sigma_n \rangle$ can be calculated: The weighted feature function score $f_\Phi(x, y)$ of a given input signal x being classified as a given sequence of word labels y (see formula 5.5.1 on the previous page) is the linear combination of the weight of each corresponding transition $f_\Phi(x, y) = \sum_{i=1}^{n} \delta(q_{i-1}, \sigma_i, \gamma_i, q_i)$. It should be noted, however, that this score includes the weight of multiple ϵ-transitions, which do not output an explicit word label. Therefore, the label sequence y output by the system as a labelling hypothesis is the output of each transition in the respective transition path minus any ϵ labels.

Weights as negative log probabilities

In this implementation, transition weights (and thus also scores) are based on negative log probabilities to minimise problems related to floating-point arithmetic. Therefore, the linear combination of the transition weights $\sum_{i=1}^{n} \delta(q_{i-1}, \sigma_i, \gamma_i, q_i)$ equals the negative logarithm of the joint probability of the transitions $-\log \prod_{i=1}^{n} P(q_{i-1}, \sigma_i, \gamma_i, q_i)$, and so $f_\Phi(x, y) = -\log P(x, y)$: The greater the weight, the lower the probability and the worse the score.

Another advantage of using negative log probabilities which is crucial to this re-scoring experiment is that they allow a classification score $f_\Phi(x, y) \equiv \sum_{i=1}^{n} \delta(q_{i-1}, \sigma_i, \gamma_i, q_i)$ to be easily re-scored in a linear fashion: These negative log probability weights are not normalised and so the hypothesis can be re-scored by simply adding a constant term p to the weighted feature function score retrieved from the phone lattice $f'_\Phi(x, y) \equiv f_\Phi(x, y) + p$ (see section 6.2.2 on page 42 for further details).

5.5.3. Word lattice projection

Rather than outputting only the best-scoring hypothesis $\hat{y} \equiv \arg\min_{y \in \mathcal{Y}} f_\Phi(x, y)$ for an input x, the *Millennium* ASR system outputs a **word lattice** $\mathcal{L}^{\text{word}}(x)$ representing a set of possible hypotheses. This word lattice is a sub-lattice of the corresponding phone lattice $\mathcal{L}^{\text{word}} \subseteq \mathcal{L}^{\text{phone}}$.

A word lattice $\mathcal{L}^{\text{word}}$ is derived through **projection** of the corresponding phone lattice *project*: $\mathcal{L}^{\text{phone}} \mapsto \mathcal{L}^{\text{word}}$ [41, 74], in which all information in the lattice is discarded except for each unique sequence of output labels. In this process, ϵ-transitions are merged with their following word label transitions, meaning that there are no ϵ-transitions in the word lattice. The resulting lattice is a directed acyclic graph [3], with a (weak) connective upper bound of $n(n-1)$. Likewise, the FST representing a word lattice $\mathcal{T}^{\text{word}}$ is deterministic.

This smaller lattice $\mathcal{L}^{\text{word}} \subseteq \mathcal{L}^{\text{phone}}$ is used to derive an n-best list of possible hypotheses $y^{\text{word}} \in Y^{\text{word}}$ by scoring each unique hypothesis represented by a word lattice

$Y^{\text{word}} \equiv \mathcal{Y}(\mathcal{L}^{\text{word}})$ and then ranking them according to their score (see chapter 7 on page 44 for further details). This represents a much more computationally-feasible task than scoring and re-scoring all the hypothesis represented by the phone lattice used for re-scoring $\mathcal{Y}(\mathcal{L}^{\text{phone}})$, which is a directed multigraph [3, pp. 565 sqq.] and has a connective upper bound of $n(nv)$, where n is the number of vertices and v is the vocabulary size (e.g. $v \equiv |\Gamma|$ in formula 5.5.1 on page 34).

5.5.4. Word lattice scoring

The weighted feature function score $f_\Phi(x, y)$ of a symbol sequence hypothesis $y \equiv \langle \gamma_1 \dots \gamma_n \rangle$ for the input x is the best-scoring sequence of transitions $\sum_{i=1}^{n} \delta(q_{i-1}, \sigma_i, \gamma_i, q_i)$ which outputs the given word symbol sequence $\gamma_1 \dots \gamma_n$; This score is the minimum transition score sum $\sum_{i=1}^{n} \delta(q_{i-1}, \sigma_i, \gamma_i, q_i)$ for all sequences of states $q_0 \dots q_{n-1}$ which output the given output symbol sequence $\gamma_1 \dots \gamma_n$ (represented by y) given any sequence of input symbols $\sigma_1 \dots \sigma_n$ (see formula 5.5.1 on page 34):

$$\begin{gathered} f_\Phi(x, y) = \min_{\substack{\sigma_1 \dots \sigma_n \\ q_0 \cdots q_{n-1}}} \sum_{i=1}^{n} \delta(q_{i-1}, \sigma_i, \gamma_i, q_i) \\ \text{where } y \equiv \langle \gamma_1 \dots \gamma_n \rangle \end{gathered} \tag{5.5.2}$$

However, the path through a phone lattice WFST which outputs the sequence of word symbols constituting a label hypothesis $y^{\text{word}} \equiv \langle \gamma_1^{\text{word}} \dots \gamma_m^{\text{word}} \rangle$ can also entail transitions which output non-word symbols such as ϵ and concept symbols (described in section 5.2 on page 32); An output symbol sequence from a phone lattice WFST $y \equiv \langle \gamma_1 \dots \gamma_n \rangle$ which corresponds to the word label sequence $y^{\text{word}} \equiv \langle \gamma_1^{\text{word}} \dots \gamma_m^{\text{word}} \rangle$ is that which is equivalent to the word label sequence minus any non-word symbols $\gamma \notin \Gamma^{\text{word}}$:

$$\mathcal{Y}^{\text{word}} \equiv \mathcal{Y} \cap \Gamma^{\text{word}} \qquad \text{where } \Gamma^{\text{word}} \equiv \Gamma \setminus \Gamma^{\overline{\text{word}}} \tag{5.5.3}$$

Therefore, in order to calculate the score $f_\Phi(x, y^{\text{word}})$ of a word label sequence y^{word}, the best-scoring sequence of states $q_0 \dots q_{n-1}$ must be found with a corresponding output label sequence y and any sequence of input symbols $\sigma_1 \dots \sigma_n$:

$$\begin{gathered} f_\Phi(x, y^{\text{word}}) \equiv \min_{y \in \mathcal{Y}'} f_\Phi(x, y) \\ \text{where } \mathcal{Y}' \equiv \{ y \in \mathcal{Y} : y \cap \Gamma^{\text{word}} = y^{\text{word}} \} \end{gathered} \tag{5.5.4}$$

Algorithm

In order to find the best-scoring path through the WFST representing a phone lattice $\mathcal{T}^{\text{phone}}$ which outputs a symbol sequence y corresponding to the word label sequence to be scored y^{word}, an iterative search algorithm inspired by the Viterbi algorithm [5, pp. 629–631, 69] was developed which finds the best-scoring sequence of states in the WFST $q_0 \dots q_{n-1}$ which outputs the symbol sequence y, performing an iterative breadth-first search for each word label $\gamma_i^{\text{word}} \in y^{\text{word}}$ by expanding intermediate edges which

output non-word symbols in a depth-first manner: Each word label $\gamma_i^{\text{word}} \in y^{\text{word}}$ is an intermediate goal in the path representing a sequence of word label goals $y^{\text{word}} \equiv \langle \gamma_1^{\text{word}} \dots \gamma_m^{\text{word}} \rangle$. In order to find the next word label $\gamma_{i+1}^{\text{word}}$, all such **goal edge** transitions $\langle s, \sigma, \gamma_i^{\text{word}}, w, e \rangle \in \delta$ (see formula 5.5.1 on page 34) which were reached during the previous search iteration i are expanded, searching each path from the state e in a depth-first manner by expanding every possible path until either it ends at the next goal label $\gamma_{i+1}^{\text{word}}$ or there are no more edges to expand which have not already been visited by a better-scoring path (thus avoiding cycles).

By using such a two-stage iterative search, this search algorithm is admissible [cf. 55, pp. 107–109], always finding the best-scoring path outputting the sequence $y^{\text{word}} \equiv \langle \gamma_1^{\text{word}} \dots \gamma_m^{\text{word}} \rangle$ if it exists in the lattice. For a detailed description of the algorithm, see appendix D on page 95.

6. Knowledge-based re-scoring

In order to perform re-scoring experiments on the ASR word/phone lattice output, the n-best hypotheses $y_1 \dots y_n$ are retrieved from the word lattice FST $\mathcal{T}^{\text{word}}(x)$ projected by the ASR system (see section 5.5.3 on page 35), and the best score for each hypothesis $f_\Phi(x, y)$ is retrieved from the phone lattice WFST $\mathcal{T}^{\text{phone}}(x)$ using the lattice search method described in section 5.5.4 on page 36. The resulting set of unique word label sequences and their scores sorted in ascending order according to score represents n-best classification hypotheses for x. This original score ranking is used to evaluate the effectiveness of various re-scoring methods by calculating the difference between metrics derived from this score and those from the score after re-scoring.

6.1. Definition

Re-scoring is then done by evaluating the semantic attribute vector representation of a hypothesis y against a set of weighted constraint penalty functions $f_C(x, y)$ which encode knowledge from higher-level KSs (see section 3.3 on page 22). This knowledge score is then combined with the original weighted feature-function score $f_\Phi(x, y)$ to calculate the new relative ranking of the input-output pair $\langle x, y \rangle$:

$$f_{\Phi,C}(x, y) \equiv f_\Phi(x, y) + f_C(x, y) \tag{6.1.1}$$

In this way, it can be said that the weighted feature-function score $f_\Phi(x, y)$ for an input-output pair $\langle x, y \rangle$ is biased by a parallel weighted dynamic CSP which penalises classification hypotheses y that do not satisfy the constraints defined in section 6.2.

6.2. Constraint-based knowledge scoring

The knowledge score of a hypothesis $f_C(x, y)$ is formulated as a weighted constraint penalty function, which represents the score of the hypothesis y given the utterance being classified and the context in which it is situated x; Therefore, in this formulation, the input x is not only a vector of acoustic signal features but also includes information about the contextual state (in this case, the state of the ATC simulation). In order to calculate this score, a vector of semantic attributes $z \equiv \begin{bmatrix} z_1 & \cdots & z_n \end{bmatrix}$ is derived from the hypothesis $z \cong y$ and this input-output pair $\langle x, z \rangle$ is evaluated against the knowledge encoded by a set of declarative constraints C as a weighted CSP $f_C(x, y) \cong f_C(x, z)$:

$$
\begin{aligned}
f_C(x, y) &\cong f_C(x, z) \\
&\equiv p^\intercal C(x, z) \\
C &\equiv \{c_i(\cdot)\} \qquad \text{where } c_i \colon \mathcal{X} \times \mathcal{Z} \mapsto \mathbb{B}
\end{aligned} \tag{6.2.1}
$$

See appendix E on page 100 for details on the implementation of concept parsing and evaluation against the ATC situation model.

6.2.1. Constraint knowledge

Two types of situational knowledge were encoded through declarative constraints for re-scoring: **Referential knowledge**, which is knowledge of which entities are valid possible referents denoted by an utterance in question, and **physical knowledge**, which is knowledge of valid next physical states given the current physical state of the context in which the utterance was made. In the domain of ATC, referents are aircraft to which commands are given, which can be uniquely identified by callsign. Likewise, physical knowledge in the domain of ATC includes knowledge of the next possible flight information values for an aircraft in question given its current flight information, the command given to the aircraft and the current state of the airspace in general.

Both the set of referential knowledge constraints and that of physical knowledge constraint can divided further into three sub-types of knowledge constraints: Firstly, those which encode **knowledge certainty**, i.e. which constrains valid attribute values to a single value; Secondly, those which encode **static knowledge**, i.e. which encode knowledge that does not change throughout the scenario; Thirdly, those which encode **dynamic knowledge**, i.e. knowledge which is dependent on the current state of the scenario and so constrains values differently depending on the current state of the ATC context.

Referential knowledge

The referential knowledge used in this experiment (about which aircraft are valid referents) was encoded by set-membership constraints on the valid callsign(s) denoted by an utterance in question: For each constraint defined, there is a set of valid callsigns V, and a label sequence hypothesis satisfies the constraint if the CALLSIGN concept value it denotes $y \to v$ is in the set of valid callsigns $v \in V$.

Three sub-types of constraints were defined which encode referential knowledge in the domain of ATC: One which statically constrains valid aircraft callsigns to those denoted by the gold standard label sequence, one which statically constrains valid callsigns to those which are possible in the simulation, and one which dynamically constrains valid callsigns to those which are valid in the simulation context in which the utterance was made.

Referential certainty First, one constraint $c^{\text{goldstd}}_{\text{callsign}}$ evaluates the aircraft callsign denoted by a hypothesis against that of the gold standard label sequence $V^{\text{goldstd}}_{\text{callsign}} \equiv \{v^{\text{goldstd}}\}$. This constraint was used to test the effectiveness of knowing the specific

aircraft to which a command is given, e.g. if the knowledge of which aircraft is being addressed is provided non-verbally, such as being selected by the air traffic controller with a mouse and/or keyboard.

Static referential knowledge Another constraint $c^{\text{possible}}_{\text{callsign}}$ evaluates the aircraft callsign denoted by a hypothesis against that of the set of the callsigns of all aircraft which can occur in the given scenario; In this case, this is 35 callsigns which are covered by the ASR restricted grammar, representing all the aircraft which could occur in the simulation, i.e. $|V^{\text{possible}}_{\text{callsign}}| = 35$ (see table B.4 on page 83 for further details). This constraint was used to test the effectiveness of knowing the specific aircraft which can occur in a given scenario, e.g. all aircraft scheduled to enter a given airspace on a given day.

Dynamic referential knowledge The third callsign constraint $c^{\text{context}}_{\text{callsign}}$ uses the situational model created from the 4D-CARMA server log file to evaluate the aircraft callsign denoted by a hypothesis against that of the set of aircraft for which there is flight information within the time envelope $\tau - 5 \ldots \tau$, where τ is the time at which the utterance was made, measured in seconds (see section E.2 on page 101). In other words, the set $V^{\text{context}}_{\text{callsign}}$ is the subset of all possible callsigns $V^{\text{context}}_{\text{callsign}} \subseteq V^{\text{possible}}_{\text{callsign}}$ for which each aircraft denoted by a member thereof $v \in V^{\text{context}}_{\text{callsign}}$ is in the simulation airspace at the time of the utterance within 5 seconds. Therefore, this constraint dynamically constrains the set of valid utterances to those which denote a valid aircraft callsign in a particular context.

Physical knowledge

A number of constraints in the experiment encode common-sense knowledge about the physical world and the interface between ATC utterance semantics and the physical constraints on aircraft. They compare the flight information denoted by a given hypothesis to the last given flight information of the aircraft in question (e.g. ALTITUDE denoted by `<s> <callsign> <airline> lufthansa </airline> <flightnumber> four three nine </flightnumber> </callsign> <command_type="descend"> descend altitude <altitude> four thousand </altitude> feet </command> </s>`). In order to do this, the function $H(\beta)$ gets the last given flight information for an aircraft β as a tuple $H(\beta) \mapsto \langle \text{alt}, \text{spd}, \text{hdg} \ldots \rangle$ denoting the last flight information of the aircraft according to the situation model (see section E.2 on page 101). The goal value denoted by a hypothesis y (e.g. `4000` for the concept ALTITUDE as denoted by the previous example) is then compared to the last given flight information value for the aircraft β denoted by the command in order to evaluate its validity. This is done by getting the value of the CALLSIGN attribute from the corresponding semantic attribute vector $\mathit{callsign}(\cdot) \mapsto \beta$, and using this value to get the corresponding aircraft flight information $H(\beta) \mapsto \langle \text{alt}, \text{spd}, \text{hdg} \ldots \rangle$:

$$
\begin{aligned}
c_{\alpha}^{\text{fi}}(\cdot) &\equiv c'(z, h_{\alpha}) \qquad \text{where } h \equiv H(callsign(z)) \\
callsign &\colon \mathcal{Z} \mapsto \mathcal{B} \\
H &\colon \mathcal{B} \mapsto \mathcal{I} \qquad \text{where } i \in \mathcal{I} \equiv \langle \text{alt}, \text{spd}, \text{hdg} \ldots \rangle
\end{aligned} \tag{6.2.2}
$$

where $c'(z, h_{\alpha})$ evaluates either an equality or inequality of z and h_{α}, for example $c'(z, h_{\text{alt}}) \equiv [\![4000 \geq h_{\text{alt}}]\!]$ in the case of a constraint constraining valid altitudes for a DESCEND command given the hypothesis `<s> <callsign> <airline> lufthansa </airline> <flightnumber> four three nine </flightnumber> </callsign> <command_type="descend"> descend altitude <altitude> four thousand </altitude> feet </command> </s>` (further details later in this section).

Physical certainty Only one physical constraint was defined which encodes physical certainty: That which constrains valid aircraft headings to 250°, that of the single runway in the simulation: In the given ATC simulation, a TURN command directs an aircraft to a runway which is parallel to heading 250° (the only runway in the simulation scenario); Therefore, a constraint was defined which restricts the HEADING of a TURN command to 250° $c_{\text{hdg}}^{\text{rwy}}(\cdot) \equiv [\![z_{\text{hdg}} \neq 250]\!]$. This constraint was used to test the effectiveness of knowing a static heading to which aircraft are regularly directed in a specific situation, e.g. in this case during approach for landing.

Static physical knowledge Two constraints were defined which encode static physical knowledge, constraining valid HEADING values for a TURN command to a value between 0 and 360 $c_{\text{hdg}}^{\text{min}}(\cdot) \equiv [\![z_{\text{hdg}} < 0]\!]$ and $c_{\text{hdg}}^{\text{max}}(\cdot) \equiv [\![z_{\text{hdg}} \geq 360]\!]$. These constraints are defined because the grammars used cover all label sequences for HEADING concepts which correspond to the CFG production rule left-hand side `NUMBER NUMBER zero` despite the fact that heading values outside the range 0 to 360 are not semantically valid as aircraft headings (see appendix C on page 84 for further details).

Dynamic physical knowledge Two types of flight information were constrained by encoding dynamic physical knowledge as dynamic constraints: aircraft airspeed and altitude/flight level.

Airspeed Given a command to reduce airspeed to a given speed z_{spd}, the aircraft to which the command was given has a last given airspeed h_{spd} which is greater than the airspeed to which it is to reduce $z_{\text{spd}} < h_{\text{spd}}$. This information was encoded by a constraint which constrains the goal airspeed denoted by COMMAND concepts of the subtype REDUCE to values which are less than the value of the last given airspeed (e.g. `<s> <callsign> <airline> qantas </airline> <flightnumber> six </flightnumber> </callsign> <command_type="reduce"> reduce speed <speed> two hundred </speed> knots </command> </s>`). In other words, this knowledge is encoded as a constraint $c_{\text{spd}}(\cdot) \equiv [\![z_{\text{spd}} \geq h_{\text{spd}}]\!]$ which evaluates the expression $z_{\text{spd}} \geq h_{\text{spd}}$.

Altitude/flight level Given a command to descend to a given altitude or flight level, the aircraft to which the command was given is at an altitude h_{alt} or flight level h_{fl} which is greater than the altitude/flight level to which it is to descend ($z_{\text{alt}} < h_{\text{alt}}$ or $z_{\text{fl}} < h_{\text{fl}}$, respectively). This information was encoded into two constraints: The first constrains the goal FLIGHTLEVEL denoted by DESCEND concepts to values which are less than the value of the last given aircraft flight level h_{fl} (e.g. `<s> <callsign> <airline> qantas </airline> <flightnumber> six </flightnumber> </callsign> <command_type="descent"> descend flight level <flightlevel> two two zero </flightlevel> </command> </s>`); Analogously to the airspeed constraint $c_{\text{spd}}(\cdot)$, the constraint $c_{\text{fl}}(\cdot) \equiv [\![z_{\text{fl}} \geq h_{\text{fl}}]\!]$ evaluates the expression $z_{\text{fl}} \geq h_{\text{fl}}$.

Likewise, another constraint $c_{\text{alt}}(\cdot) \equiv [\![z_{\text{alt}} \geq h_{\text{fl}}]\!]$ constrains goal ALTITUDE values to those which are less than the last given aircraft altitude (e.g. `<s> <callsign> <airline> qantas </airline> <flightnumber> six </flightnumber> </callsign> <command_type="descent"> descend altitude <altitude> four thousand </altitude> feet </command> </s>`). However, due to the fact that the 4D-CARMA server logs aircraft altitude only in flight level measures, and that flight level is derived from altitude $fl = alt/100$, this constraint is implemented as $c_{\text{alt}}(\cdot) \equiv [\![z_{\text{alt}} \geq 100h_{\text{fl}}]\!]$.

6.2.2. Constraint weighting

All of the constraints defined in section 6.2.1 on page 39 are intended to be hard constraints in these experiments. However, an arbitrary positive finite constant value g^* was chosen as a weight for all constraints because a hypothesis ranking is required even in cases where no hypothesis in the set satisfies all constraints: Applying truly 'hard' constraints (i.e. whhich have an infinite-value weight $w_i \in w \equiv \infty$) would entail that, in such a situation, the score of all hypotheses would be the same (i.e. $f_\Phi(x, y) + f_C(x, y) = \infty$). Therefore, this constant value g represents an arbitrary upper bound for scores of hypotheses which are considered reasonable; By not satisfying at least one constraint, a hypothesis will have a score equal to or greater than g.

6.2.3. Knowledge score application

The knowledge score of a hypothesis y given the context in which the corresponding utterance was made $f_C(x, y)$ is calculated as the dot product of the constraint penalty functions and their respective weights $p \cdot C(x, y) = p^\intercal C(x, y)$ (see formula 6.2.1 on page 39). However, each constraint has the same weighting $w_i \in w \equiv g$, as described in section 6.2.2: Therefore, the constraint penalty score equals the scalar product of the maximum penalty sum g and the constraint evaluation functions $f_C(x, y) = g \cdot C(x, y)$.

6.3. Test sets

The effectiveness of each constraint set defined in section 6.2 on page 38 was tested by testing each possible combination of constraints which are deemed semantically plausible:

*In this implementation, $g \equiv 32767$, the maximum value of a 16-bit signed integer.

A semantically plausible combination of constraints is one in which the predicate which a constraint encodes $c_i \cong \varphi$ does not entail the predicate encoded by any other constraint in the set $c_j \cong \psi$:

$$\underset{c_i \in C}{\forall} \underset{c_j \in C}{\forall} \varphi \nrightarrow \psi \qquad \text{where } c_i \neq c_j, c_i \cong \varphi, c_j \cong \psi \tag{6.3.1}$$

For example, for this reason, there are no constraint test sets which have all three HEADING constraints defined in section 6.2.1 on page 40 $\{c^{\text{min}}_{\text{hdg}}, c^{\text{max}}_{\text{hdg}} c^{\text{rwy}}_{\text{hdg}} \dots\} \notin TESTS$: The predicate encoded by $c^{\text{rwy}}_{\text{hdg}}$ restricts valid HEADING values to 250, which entails the predicates encoded by $c^{\text{min}}_{\text{hdg}}$ and $c^{\text{max}}_{\text{hdg}}$, which encode the knowledge that a valid HEADING value is respectively greater than or equal to 0 and less than 360 (see section 6.2.1 on page 41 for details). Therefore, the effects of the constraints in a given test set should be orthogonal.

However, a semantically-plausible combination of constraints must also encode a set of predicates which define a KS completely: For example, the presence of $c^{\text{min}}_{\text{hdg}}$ or $c^{\text{max}}_{\text{hdg}}$ in a test set entails the presence of the other, because both are necessary to fully encode the knowledge that an aircraft heading is in degrees (i.e. a value between 0 and 360).

In conclusion, a total of 72 different constraint combinations were tested including the case where the set of constraints is empty $C = \emptyset$ — the results thereof representing the baseline, i.e. the accuracy before re-scoring, using only the weighted feature function score $f_\Phi(x, y)$:

$$TESTS \equiv \left\{ \begin{matrix} \{c^{\text{goldstd}}_{\text{callsign}}\} \\ \{c^{\text{possible}}_{\text{callsign}}\} \\ \{c^{\text{context}}_{\text{callsign}}\} \\ \emptyset \end{matrix} \right\} \times \left\{ \begin{matrix} \{c_{\text{spd}}\} \\ \emptyset \end{matrix} \right\} \times \left\{ \begin{matrix} \{c_{\text{alt}}\} \\ \{c_{\text{fl}}\} \\ \{c_{\text{alt}}, c_{\text{fl}}\} \\ \emptyset \end{matrix} \right\} \times \left\{ \begin{matrix} \{c^{\text{min}}_{\text{hdg}}, c^{\text{max}}_{\text{hdg}}\} \\ \{c^{\text{rwy}}_{\text{hdg}}\} \\ \emptyset \end{matrix} \right\} \tag{6.3.2}$$

$$|TESTS| = 72$$

In conclusion, the four constraint sets defined in section 6.2.1 on page 39 represent four different parameters of the scoring function $f_{\Phi,C}(x, y)$; The 72 different constraint combinations defined in formula 6.3.2 represent all valid parameter value combinations tested.

7. Evaluation

In order to evaluate the effectiveness of constraint-based lattice re-scoring, the baseline results for classification using each grammar were calculated; The effectiveness of a set of constraints C is defined as the difference in accuracy of before and after re-scoring for the respective grammar.

For each classification test, the weighted feature function score $f_\Phi(x, y)$ was calculated for each hypothesis retrieved from each word lattice FST $y \in \mathcal{Y}(\mathcal{T}^{\text{word}}(x))$ in the test dataset, as described in section 5.5.4 on page 36. An n-best list of the hypotheses was then created by sorting the hypotheses by their respective score. This list was used to evaluate classification accuracy for an utterance before any re-scoring is performed.

Likewise, for each constraint test set $C \in TESTS$ (see formula 6.3.2 on the previous page), the knowledge-based constraint penalty function score $f_C(x, y)$ was computed for each hypothesis retrieved from each word lattice FST in the test dataset $y \in \mathcal{Y}(\mathcal{T}^{\text{word}}(x))$, as described in section 6.2 on page 38. For each hypothesis, the combined score $f_{\Phi,C}(x, y) \equiv f_\Phi(x, y) + f_C(x, y)$ was calculated, and an n-best list was created using this combined score; This list was used to evaluate classification accuracy after re-scoring using the given set of constraints C.

The classification results for a test (either of the baseline or for re-scoring) were evaluated by comparing the accuracy of the best-scoring label sequence to the gold standard (as is typically done in ASR) for the utterance as well as by evaluating the rank of the gold standard label sequence in the corresponding n-best list of hypotheses (as is typically done in a retrieval task).

7.1. Best-classification evaluation

For each test, the word error rate (WER) and sentence error rate (SER) was calculated for the best-scoring hypothesis $\hat{y} \equiv \arg\min_{y \in Y} f(x, y)$ out of all hypotheses represented by the corresponding word lattice FST $Y \equiv \mathcal{Y}(\mathcal{T}^{\text{word}}(x))$; This was done by comparing the best-scoring hypothesis $\hat{y}$ to the gold standard word label sequence for the utterance $gold\_standard(x)$. WER and SER are often cited in percent values [cf. 79], as is also done in this thesis.

7.1.1. Word error rate

The most commonly-used metric for evaluation in ASR is WER, which is a metric of the distance between the word label sequence output by the ASR system and the gold standard s (which was actually said) [32, pp. 362–364]. WER is defined as a derivation of Levenshtein distance [39]:

$$WER(s) \equiv \frac{\text{ins}(s) + \text{del}(s) + \text{sub}(s)}{W(s)} \tag{7.1.1}$$

where $ins(s)$ is the number of word insertions, $del(s)$ is the number of deletions and $sub(s)$ is the number of substitutions needed to align the two sequences. $W(s)$ is the amount of words in the reference [64, pp. 2091–2092]. Therefore, the WER of a set of sentences S is:

$$WER(S) \equiv \frac{\sum_{s \in S} \text{ins}(s) + \text{del}(s) + \text{sub}(s)}{\sum_{s \in S} W(s)} \tag{7.1.2}$$

7.1.2. Sentence error rate

Another commonly used for evaluation in ASR is SER, which is the rate of sentences having at least one error (i.e. the rate of sentences not being recognised perfectly) [32, p. 363, 64, pp. 2091–2092]:

$$SER(S) \equiv \frac{\sum_{s \in S} \text{sgn}(\text{ins}(s) + \text{del}(s) + \text{sub}(s))}{|S|} \tag{7.1.3}$$

Although WER and SER are often related, this is not always the case [63]: Generally, SER increases with WER, but one cannot be inferred from the other. Therefore, both metrics can be useful.

7.2. Retrieval-task evaluation

However, these experiments were not designed to directly improve the accuracy of only the best-scoring hypothesis but rather to evaluate the effectiveness of re-scoring each hypothesis based using additional knowledge; Therefore, re-scoring was also evaluated as a retrieval task, in which the rank of the correct answer (in this case, the gold standard word label sequence) in an ordered set of answers is evaluated [32, p. 821, 53]. In order to measure this, the mean reciprocal rank (MRR) of the gold-standard classification for each utterance in the test set was calculated given the corresponding n-best hypothesis list for the utterance.

7.2.1. Mean reciprocal rank

MRR is a metric commonly used for measuring accuracy in retrieval tasks (e.g. question-answering systems) which is derived from the reciprocal rank of the first correct answer in the results of a query [32, p. 821, 53]; The **reciprocal rank** of a hypothesis $RR(y)$ is the multiplicative inverse of its rank $R(y)$ in the set of hypotheses or zero if the rank $RR(y)$ is zero (i.e. if the set of ranked hypotheses does not contain the given one) [32, p. 821]:

$$RR(y) \equiv \begin{cases} 1/R(y) & \text{if } R(y) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{7.2.1}$$

Therefore, the mean reciprocal rank for a set of hypotheses is the sum of the ranks of the hypotheses normalised by the size of the set:

$$MRR(Y) \equiv \frac{1}{|Y|} \sum_{y \in Y} \frac{1}{R(y)} \tag{7.2.2}$$

7.3. Statistical significance

In this experiment, the difference of two test results was estimated to be statistically significant if their error ranges $ERR \equiv \{x : ERR_{\min} \leq x \leq ERR_{\max}\}$ (largely) do not overlap, where the error range is derived from the standard deviation $ERR_{\min} \equiv \max(MEAN - SD/2, 0)$ and $ERR_{\max} \equiv \min(MEAN + SD/2, 100)$ in the case of WER and $ERR_{\max} \equiv \min(MEAN + SD/2, 1)$ in the case of MRR; An error measure is not calculated for SER due to the fact that SER is defined for an entire set of utterances being evaluated.

Although this is not a rigorous test of significance [4], the results varied by a relatively large amount between tests and thus rigorous statistical significance testing was not required for effectively interpreting the results.

Part III.

Results

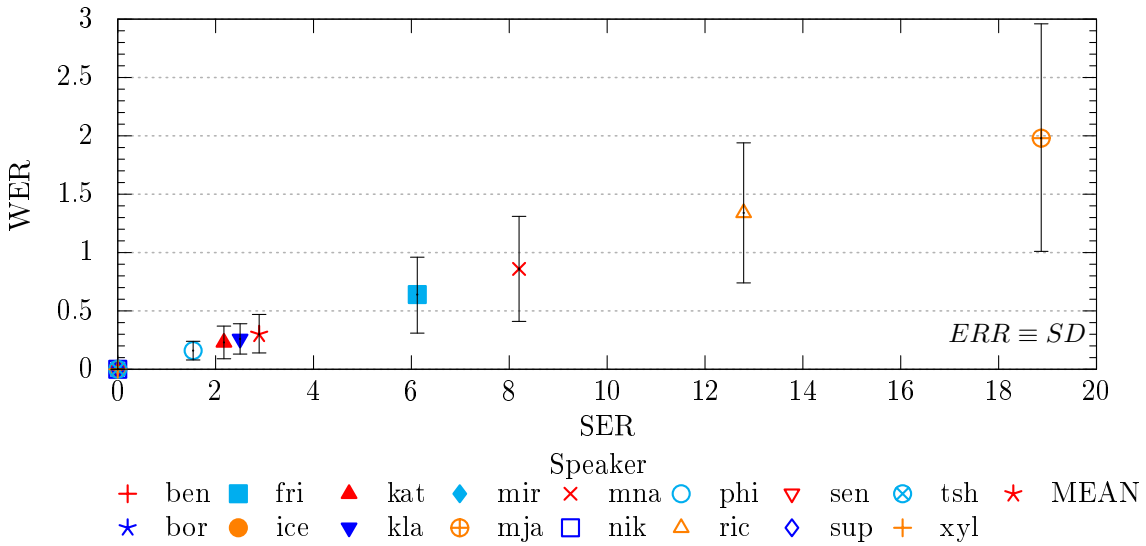
8. Baseline results

8.1. Restricted grammar

8.1.1. WER and SER

When testing on all lattices with the restricted grammar, the mean WER for all utterances was 0.30 and the mean SER was 2.89. The minimum WER and SER was 0.00 and the maximum WER and SER was 1.98 and 18.87, respectively; As can be seen in figure 8.1 there was a large variation in the WER and SER of classifying the set of utterances for each speaker.

Figure 8.1.: Baseline WER and SER results by speaker for classification using the restricted grammar.



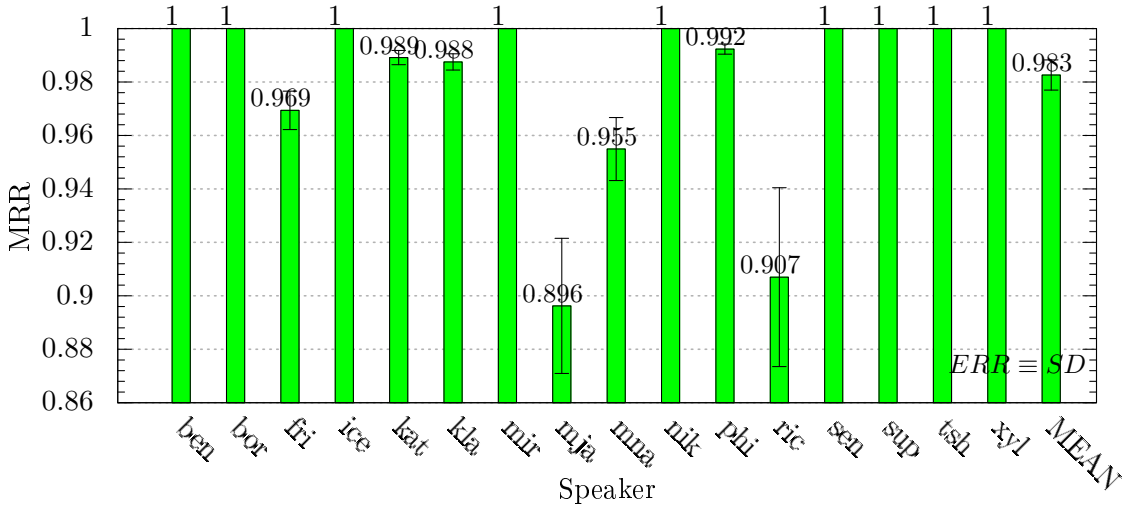
Moreover, there was not only large variation between speaker groups but also within them: The mean WER standard deviation (SD) was 0.33, but had a minimum value of 0.0 and a maximum value of 1.95.

Therefore, even when using a relatively restricted grammar, there were some speakers for which the ASR system performs significantly different than for others. For details on the results, see table F.1 on page 102.

8.1.2. MRR

Similarly to the large variance seen in the WER and SER of the speaker groups, there was also a large variation in the MRR among the groups, with a mean value of 0.983 but a minimum of 0.897 and a maximum of 1.0.

Figure 8.2.: Baseline MRR results by speaker for classification using the restricted grammar.



Just as in the case of the WER and SER, there was both a large variation between speaker groups and within speaker groups (as shown in figure 8.2), with a mean MRR SD of 0.0118, a minimum of 0.0 and a maximum of 0.0669; For details, see table F.2 on page 103.

8.1.3. Error analysis

When classifying the utterances using the restricted grammar, the only type of word errors made were in classifying numbers, as shown in figure 8.3 on the next page and table 8.1 on the following page: For example, out of 411 utterances denoting a DESCEND COMMAND with an ALTITUDE value of `4000` (feet) (e.g. *Lufthansa six four five descend altitude four thousand feet*), thirteen were classified as denoting a value of `5000` (feet), e.g. `lufthansa six four five descend altitude five thousand feet`. However, although aircraft flight numbers are composed of numbers (see section 4.2.5 on page 29), there were no word errors for words which form part of a flight number.

Irrecoverable sentences

Out of the word lattices $\mathcal{L}^{\text{word}}(x)$ which have an incorrect label sequence as the highest-scoring one $\hat{y} \neq gold\_standard(x)$, six of them did not contain the gold standard label

Figure 8.3.: Baseline errors made during classification using the restricted grammar.

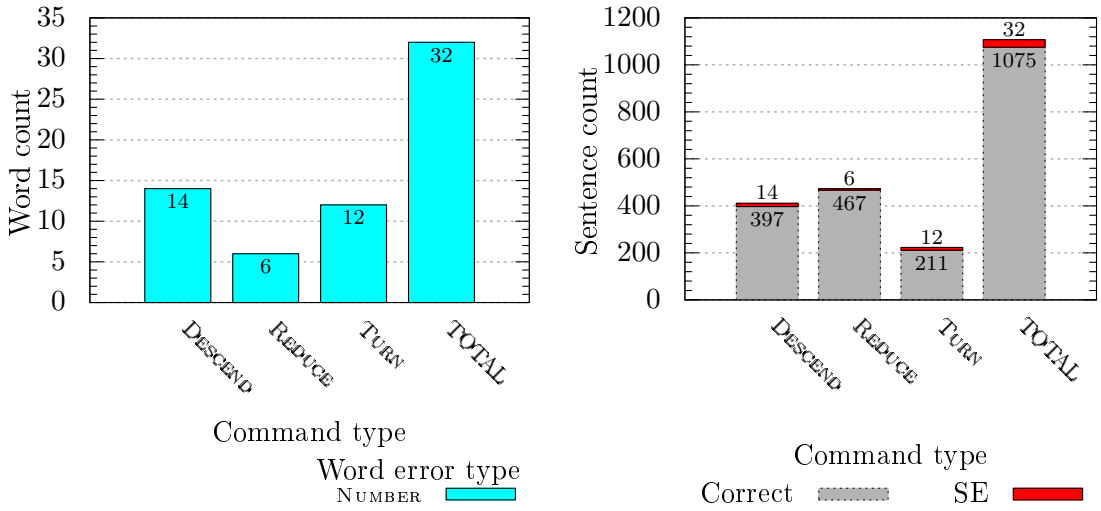


Table 8.1.: Baseline errors made during classification using the restricted grammar.

Command Type	S	SE	W	WE	NUMBER
DESCEND	411	14	3691	14	14
REDUCE	473	6	4676	6	6
TURN	223	12	2201	12	12
TOTAL	1107	32	10568	32	32

sequence $gold\_standard(x) \notin \mathcal{Y}(\mathcal{L}^{\text{word}}(x))$.

Therefore, the minimum possible WER for classification with the lattices created using the restricted grammar was 0.05; Likewise, the minimum SER was 0.54. The maximum possible MRR for the restricted grammar lattices was 0.995 (see table 8.2 on the following page).

8.2. Unrestricted grammar

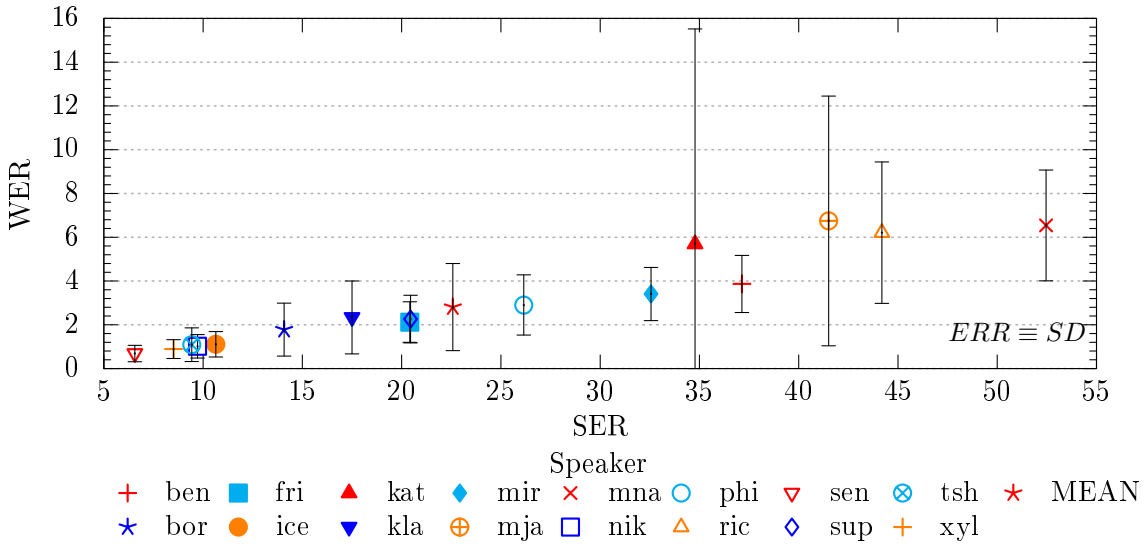
8.2.1. WER and SER

When testing on all lattices with the unrestricted grammar, the mean WER for all utterances was 2.81 and the mean SER was 22.58. The minimum WER and SER was 0.69 and 6.56 and the maximum WER and SER was 6.75 and 52.46, respectively (see figure 8.4 on the next page; For further details on the WER and SER for classification using the unrestricted-grammar lattices, see table F.3 on page 104.

Table 8.2.: The count of lattices which incorrectly classify their respective utterance for each grammar and the best SER and MRR achievable given the irrecoverable sentences.

Grammar	Total	Incorrect	Irrecoverable	Min. WER	Min. SER	Max. MRR
Restricted	1107	32	6	0.05	0.54	0.995
Unrestricted	1107	250	23	0.31	2.07	0.979

Figure 8.4.: Baseline WER and SER results by speaker for classification using the unrestricted grammar.



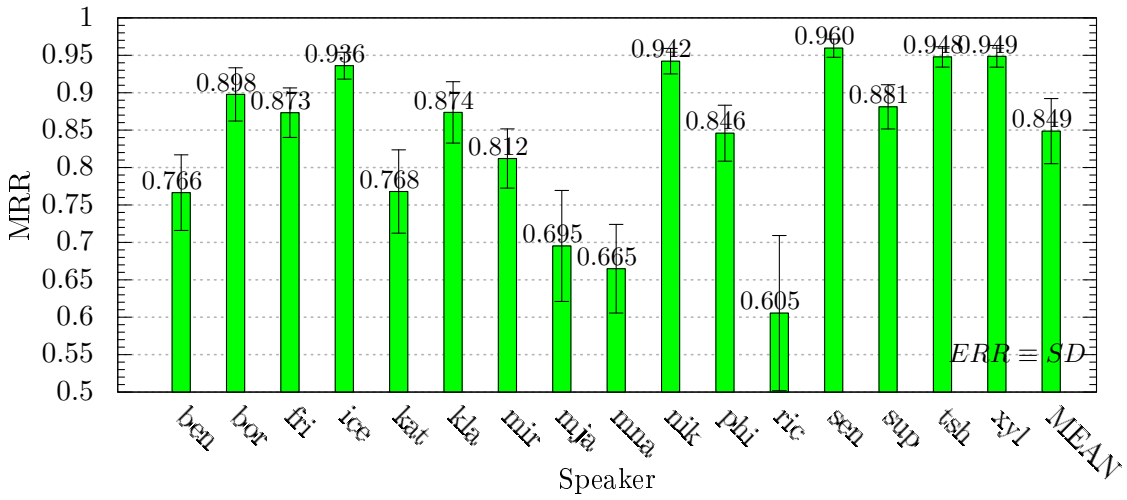
8.2.2. MRR

Finally, just as seen in the previous cases, there was a large variance in the MRR of the speaker sub-groups when using the unrestricted grammar, as seen in figure 8.5 on the next page: The mean MRR was 0.849 while the minimum was 0.605 and the maximum was 0.949. Likewise, the variance within the speaker groups was large, with a mean SD of 0.0871, a minimum of 0.0243 and a maximum of 0.2073; For further details, see table F.4 on page 104.

8.2.3. Error analysis

Just as when using the restricted grammar, there was a relatively high rate of word error when classifying numbers, as shown in figure 8.6 on page 53 and table 8.3 on the following page. However, unlike classification using the restricted grammar, there was also a significant amount of word errors in classifying callsigns: There were a total of 37

Figure 8.5.: Baseline MRR by speaker for classification using the unrestricted grammar.



incorrectly-labelled airlines.

Additionally, there were 4 miscellaneous word errors in classification which are of neither airlines nor of numbers: Each of these errors was in fact from the same utterance, the correct classification for which not being recoverable from the respective lattice used for (re-)scoring (see section 8.2.3 for further details).

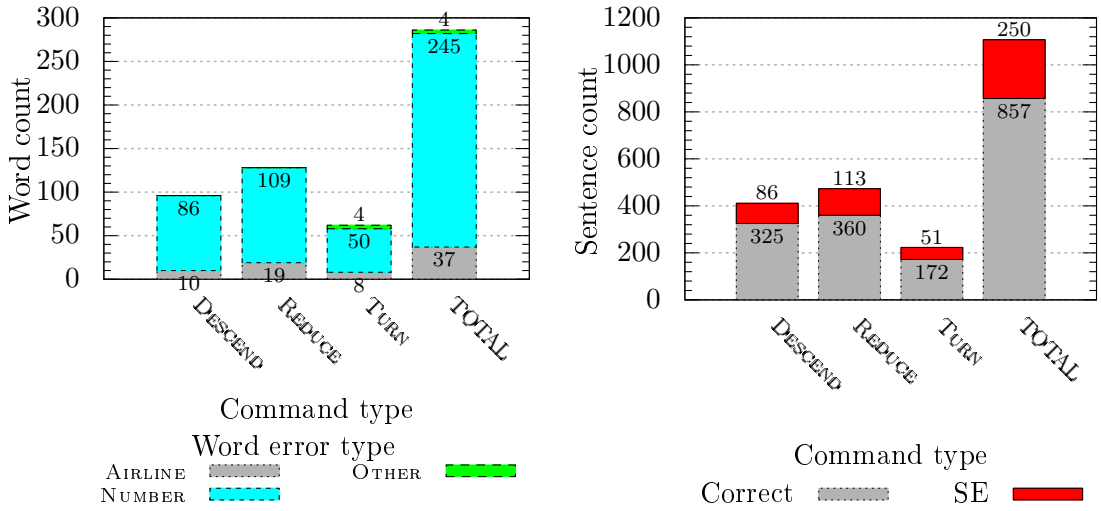
Table 8.3.: Baseline errors made during classification using the unrestricted grammar.

Command Type	S	SE	W	WE	Airline	Number	Other
Descend	411	86	3691	96	10	86	0
Reduce	473	113	4676	128	19	109	0
Turn	223	51	2201	62	8	50	4
TOTAL	1107	250	10568	286	37	245	4

Irrecoverable sentences

In the test set of lattices created using the unrestricted grammar, 23 out of 250 incorrect classifications were irrecoverable. Therefore, the minimum WER for classification using the unrestricted-grammar latices was 0.31 and the minimum SER was 2.07; The maximum MRR was 0.979 (see table 8.2 on the preceding page).

Figure 8.6.: Baseline errors made during classification using the unrestricted grammar.



8.3. Comparison

When classifying with the unrestricted grammar, the WER and SER is significantly higher than when classifying with the restricted grammar: The WER while using the unrestricted grammar is 9.37 that of classification while using the restricted grammar and the SER while using the unrestricted grammar is 7.81 that of the restricted grammar, with an absolute WER difference of 2.51 and SER difference of 19.69. Likewise, there was a 13.63% decrease in MRR when classifying with the unrestricted grammar compared to the results obtained when using the restricted grammar.

9. Re-scoring results

As described in section 6.3 on page 42, 72 different constraint combinations were tested for re-scoring the test lattices using context-specific knowledge; The results of constraint sets which are described in this chapter are those which are notable.

In this chapter, the graph label 'alt' represents the set of constraints $\{c_{\text{alt}}, c_{\text{fl}}\}$ constraining target valid ALTITUDE and HEADING values to those less than the current value for the aircraft in question and 'valHdg' represents the set of constraints defining valid HEADING values $\{c^{\text{min}}_{\text{hdg}}, c^{\text{max}}_{\text{hdg}}\}$ (see section 6.3 on page 42). Other constraint sets have labels similar to their names as defined in chapter 6 on page 38, e.g. 'rwy' $\cong c^{\text{rwy}}_{\text{hdg}}$.

9.1. Restricted grammar

The WER and SER for classification using the restricted grammar after re-scoring is shown in figures 9.1 to 9.2 on pages 54–55 for each of the notable constraint sets tested; For further details on the WER/SER for the individual constraint sets, see table G.1 on page 105.

Figure 9.1.: WER results while re-scoring using constraints for the restricted grammar.

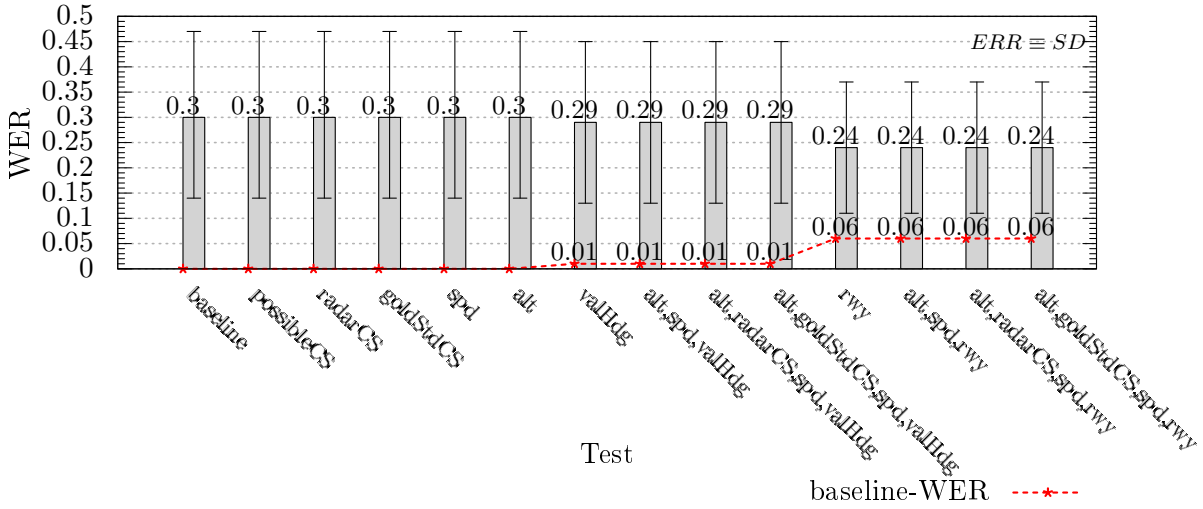
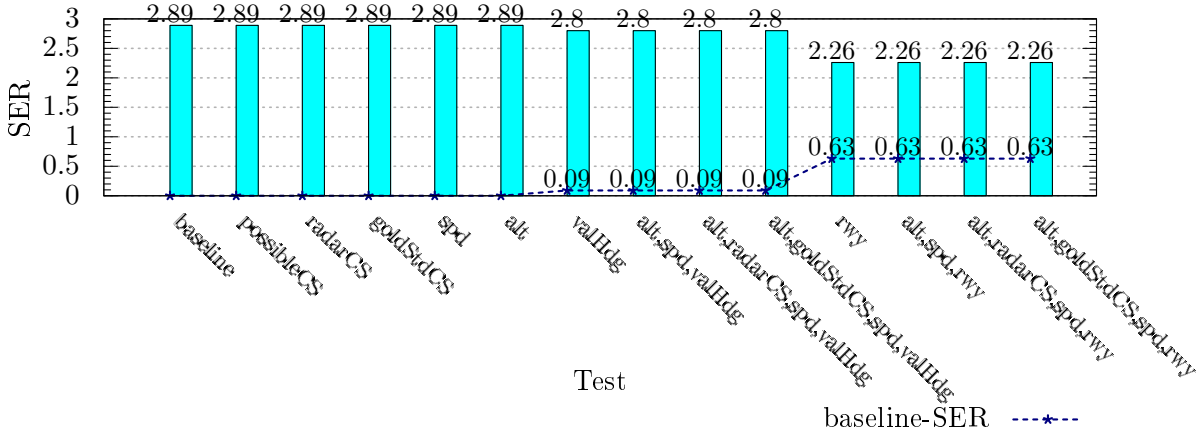
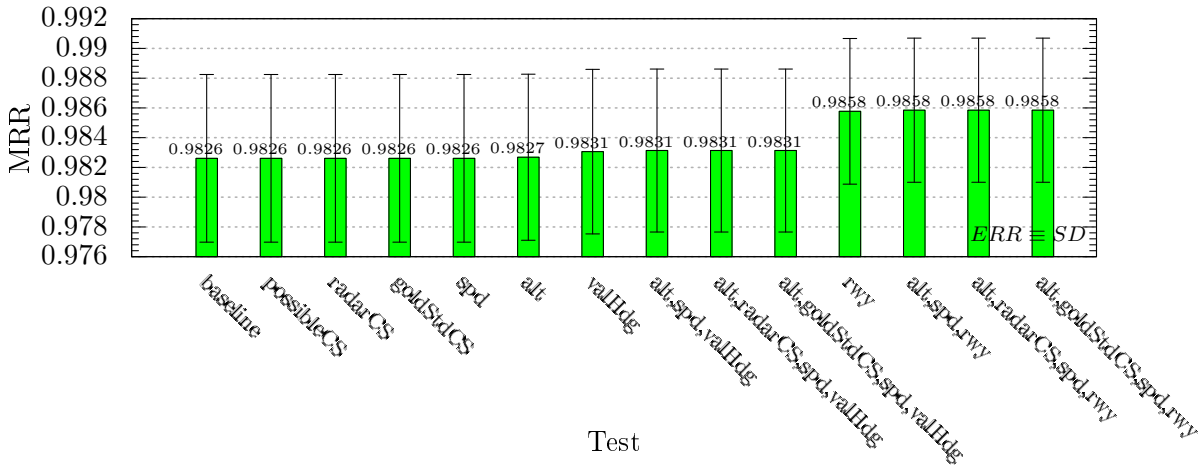


Figure 9.2.: SER results while re-scoring using constraints for the restricted grammar.



Likewise, the MRR for each respective re-scoring constraint test set using the restricted-grammar lattices is shown in figures 9.3 to 9.4 on pages 55–56; For further details, see table G.2 on page 106.

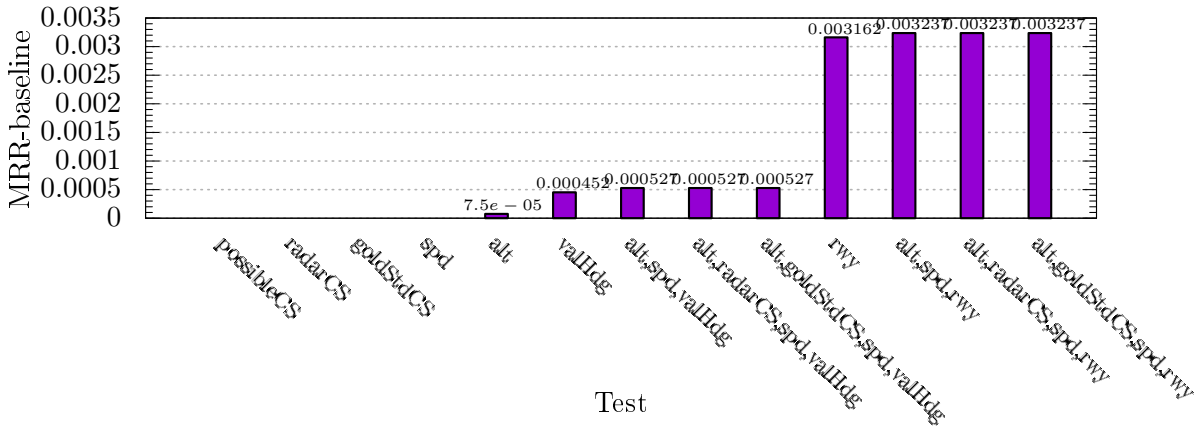
Figure 9.3.: MRR results while re-scoring using constraints for the restricted grammar.



9.1.1. Referential knowledge

Due to the fact that there were no errors in classifying callsigns (see section 8.1 on page 48), there were no improvements in accuracy over the baseline results after re-scoring using the referential knowledge constraints defined in section 6.2.1 on page 39.

Figure 9.4.: MRR results while re-scoring using constraints for the restricted grammar, relative to the baseline results.



9.1.2. Physical knowledge

When re-scoring with the flight information constraints defined in section 6.2.1 on page 40, a slight improvement in accuracy was observed for the lattices created using the restricted grammar.

The flight information constraint which achieved the greatest improvement after re-scoring the restricted- and unrestricted-grammar lattices was the runway heading constraint $c^{\text{rwy}}_{\text{hdg}}(\cdot) \equiv [\![z_{\text{hdg}} \neq 250]\!]$, which restricts valid HEADING values to a single value, 250 (see section 6.2.1 on page 41): Re-scoring the restricted-grammar lattices with this constraint resulted in a WER reduction of 20% (from 0.30 to 0.24) and a SER reduction of 19.29% (from 2.80 to 2.26), as well as a 0.32% increase in MRR (from 0.983 to 0.986).

WER and SER

As can be seen in figures 9.1 to 9.2 on pages 54–55, re-scoring using the airspeed constraint c_{spd} and altitude/flight heading constraints c_{alt}, and c_{fl} (represented by 'alt') had no effect on the WER and SER of the restricted grammar lattices. Likewise, the valid heading constraints $c^{\text{min}}_{\text{hdg}}$ and $c^{\text{max}}_{\text{hdg}}$ (represented by 'valHdg') did not have a significant effect on the WER and SER.

The only constraint by applying which a significant improvement was seen while re-scoring was the runway heading constraint $c^{\text{rwy}}_{\text{hdg}}$ (represented by 'rwy'): By applying this constraint, a 20% reduction in WER (to 0.24) and a 21.80% reduction in SER (to 2.26) was achieved.

MRR

Just as in the case of the WER and the SER, there was no significant improvement in MRR from re-scoring using the altitude, flight level or airspeed constraints, despite that the mean MRR was slightly greater while using the altitude constraint (see figures 9.3 to 9.4 on pages 55–56). Likewise, a small but insignificant improvement was achieved by applying the valid heading constraints $c^{\text{min}}_{\text{hdg}}$ and $c^{\text{max}}_{\text{hdg}}$, and a significant improvement was achieved by applying the runway heading constraint $c^{\text{rwy}}_{\text{hdg}}$.

9.1.3. Error analysis

Just as observed in the baseline results, the word errors are in classifying numbers (cf. section 8.1.3 on page 49); See figure 9.5 and table 9.1 on the following page for details.

Figure 9.5.: Errors made during classification after re-scoring for the restricted grammar, using (one of) the constraint set(s) which achieved the greatest accuracy ($\{c^{\text{goldstd}}_{\text{callsign}}, c_{\text{alt}}, c_{\text{fl}}, c_{\text{spd}} c^{\text{rwy}}_{\text{hdg}}\}$).

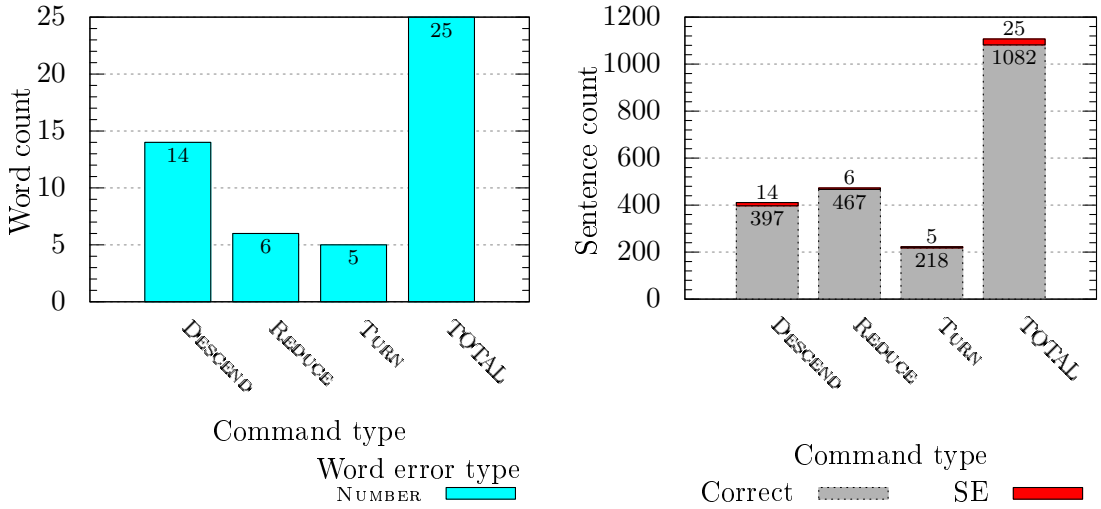


Table 9.1.: Errors made during classification after re-scoring for the restricted grammar, using (one of) the constraint set(s) which achieved the greatest accuracy ($\{c^{\text{goldstd}}_{\text{callsign}}, c_{\text{alt}}, c_{\text{fl}}, c_{\text{spd}} c^{\text{rwy}}_{\text{hdg}}\}$).

Command Type	S	SE	W	WE	Number
Descend	411	14	3691	14	14
Reduce	473	6	4676	6	6
Turn	223	5	2201	5	5
TOTAL	1107	25	10568	25	25

9.2. Unrestricted grammar

The WER and SER for classification using the unrestricted grammar after re-scoring is shown in figures 9.6 to 9.7 on pages 58–59 for each of the notable constraint sets tested; For further details on the WER/SER for the individual constraint sets, see table G.3 on page 106.

Figure 9.6.: WER results while re-scoring using constraints for the unrestricted grammar.

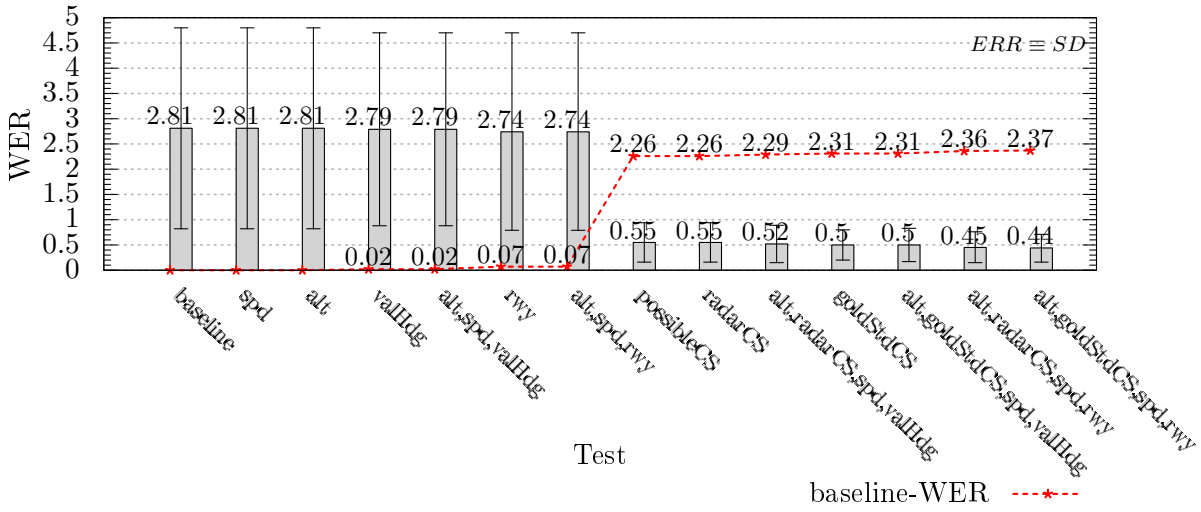
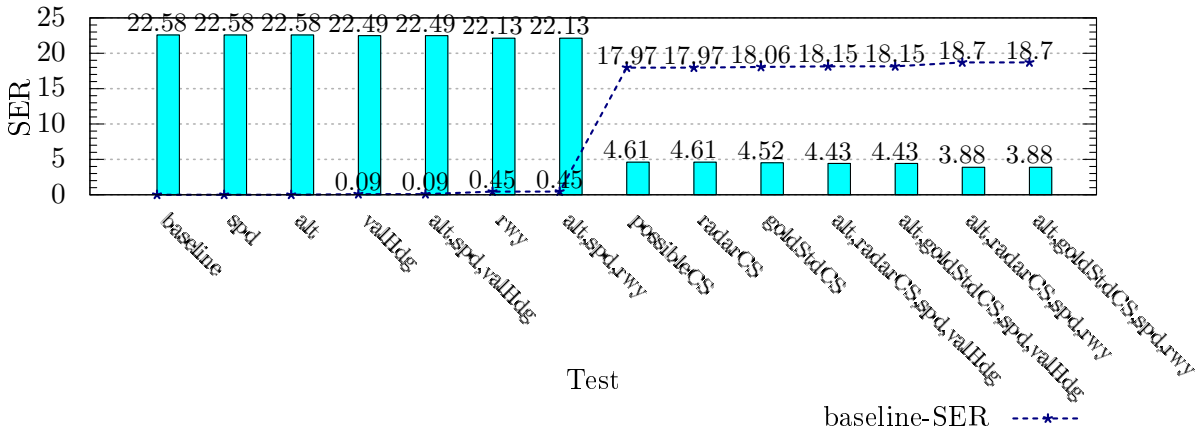
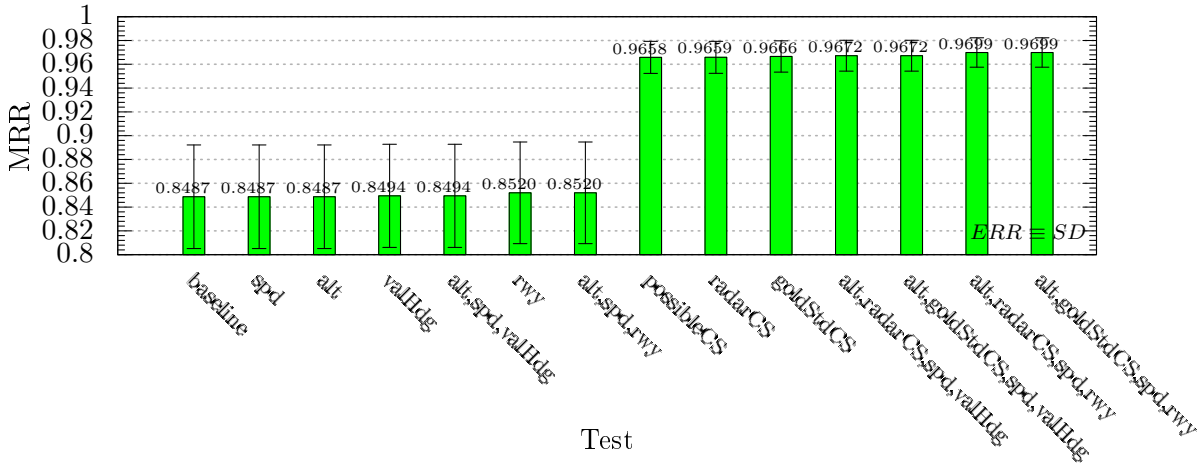


Figure 9.7.: SER results while re-scoring using constraints for the unrestricted grammar.



Likewise, the MRR for each respective re-scoring constraint test set using the unrestricted-grammar lattices is shown in figures 9.8 to 9.9 on pages 59–60; For further details, see table G.4 on page 107.

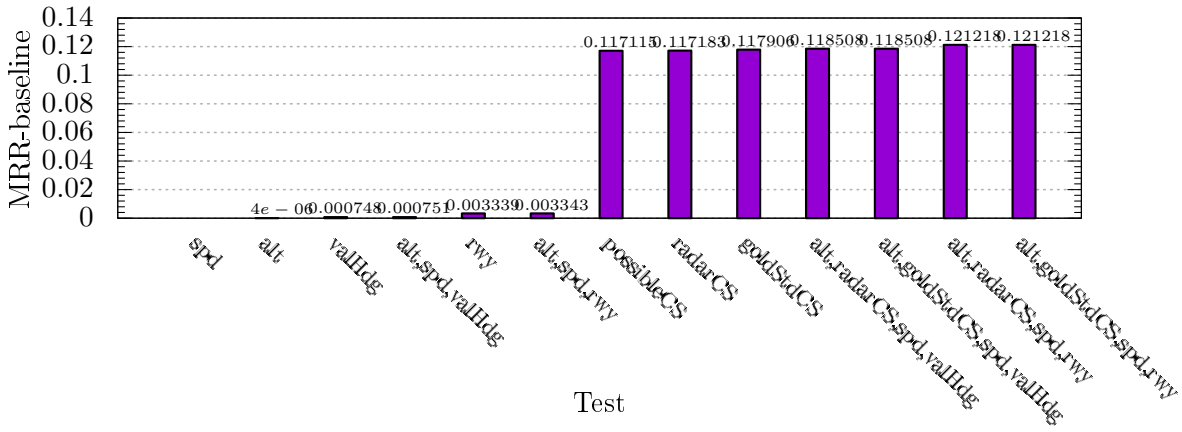
Figure 9.8.: MRR results while re-scoring using constraints for the unrestricted grammar.



9.2.1. Referential knowledge

When re-scoring the lattices created using the unrestricted grammar by applying callsign constraints, a significant improvement in classification accuracy was observed, achieving a significantly lower WER and SER and a significantly higher MRR.

Figure 9.9.: MRR results while re-scoring using constraints for the unrestricted grammar, relative to the baseline results.



WER and SER

As can be seen in figures 9.6 to 9.7 on pages 58–59, the WER was reduced at most by 82.21% to 0.5 (an absolute difference of 2.31) by applying the gold standard callsign constraint $c^{\text{goldstd}}_{\text{callsign}}$ and the SER was reduced by 79.98% to 4.52 (an absolute difference of 18.06). Moreover, the improvement seen while using either the scenario-specific possible callsign constraint $c^{\text{possible}}_{\text{callsign}}$ or the dynamic context-sensitive callsign constraint $c^{\text{context}}_{\text{callsign}}$ was nearly as great, achieving reductions of 80.43% in WER (to 0.55) and 79.58% in SER (to 4.61).

MRR

Just as in the case of the WER and SER, a significant improvement in MRR over the baseline for the unrestricted grammar was achieved by applying callsign constraints: As can be seen in figures 9.8 to 9.9 on pages 59–60, there was a significant improvement in MRR while using any of the callsign constraints, with the biggest improvement being 13.89%, achieved by applying the gold standard callsign constraint $c^{\text{goldstd}}_{\text{callsign}}$. Nevertheless, the other constraints also showed significant improvements. Moreover, unlike the case with the WER and SER, there was a slight improvement in MRR when applying the dynamic context callsign constraint $c^{\text{context}}_{\text{callsign}}$ instead of the static possible callsign constraint $c^{\text{possible}}_{\text{callsign}}$.

9.2.2. Physical knowledge

Just as the case with the restricted grammar as described in section 9.1.2 on page 56, a relatively small improvement in accuracy was observed when re-scoring the unrestricted-grammar lattices using the flight information constraints.

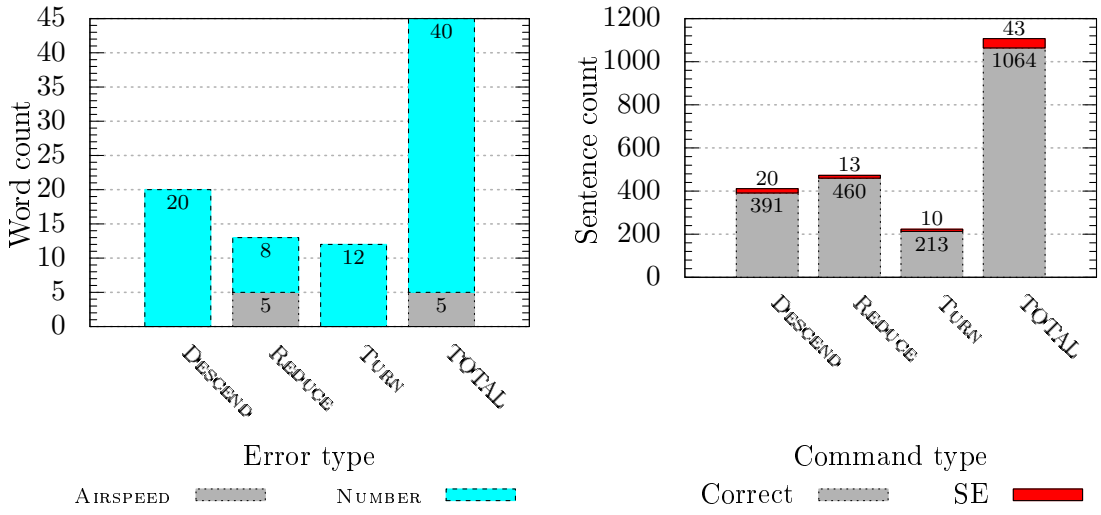
Again, the most effective constraint was the runway heading constraint $c^{\text{rwy}}_{\text{hdg}}$, achieving a 2.49% reduction in WER (from 2.81 to 2.74), a 2% reduction in SER (from 22.58 to 22.13) and a 0.39% inrease in MRR (from 0.849 to 0.852).

9.2.3. Error analysis

There was a significant reduction in word error both in the classification of airlines and numbers, as shown in figure 9.10 and table 9.2 on the next page. The largest improvement was in AIRLINE classifications, in that there were 37 such errors given the baseline results and there were none after re-scoring using the best combination of constraints $\{c^{\text{goldstd}}_{\text{callsign}}, c_{\text{alt}}, c_{\text{fl}}, c_{\text{spd}} c^{\text{rwy}}_{\text{hdg}}\}$ (see table 8.3 on page 52 and figure 8.6 on page 53).

Additionally, there were 5 word errors in classifying the word *speed* after re-scoring although such errors were not made before re-scoring: Each of these utterances involved the word *speed* classified as `airspeed`, meaning that they do not represent semantically-erroneous errors in classification. Moreover, the correct classification for each of these 5 utterances was not recoverable from the respective lattice used for (re-)scoring (see section 8.2.3 on page 52).

Figure 9.10.: Errors made during classification after re-scoring for the unrestricted grammar, using (one of) the constraint set(s) which achieved the greatest accuracy ($\{c^{\text{goldstd}}_{\text{callsign}}, c_{\text{alt}}, c_{\text{fl}}, c_{\text{spd}} c^{\text{rwy}}_{\text{hdg}}\}$).



9.3. Comparison

Applying the callsign constraints while re-scoring resulted in a significant improvement in accuracy when re-scoring the unrestricted-grammar lattices, as described in section 9.2.1

Table 9.2.: Errors made during classification after re-scoring for the unrestricted grammar, using (one of) the constraint set(s) which achieved the greatest accuracy ($\{c^{\text{goldstd}}_{\text{callsign}}, c_{\text{alt}}, c_{\text{fl}}, c_{\text{spd}} c^{\text{rwy}}_{\text{hdg}}\}$).

Command Type	S	SE	W	WE	AIRSPEED	NUMBER
DESCEND	411	20	3691	20	0	20
REDUCE	473	13	4676	13	5	8
TURN	223	10	2201	12	0	12
TOTAL	1107	43	10568	45	5	40

on page 59. However, no improvement was seen when applying the callsign constraints while re-scoring the restricted-grammar lattices (see section 9.1.1 on page 55).

As described in section section 9.1.2 on page 56, the accuracy of classification using the restricted-grammar lattices was improved by using the flight information constraints. When re-scoring the unrestricted-grammar lattices using flight information constraints, however, the improvement in accuracy was relatively small, achieving a 20% reduction in WER for the restricted-grammar lattices versus only a 2.49% reduction for the unrestricted-grammar ones; This improvement was even smaller in comparison to the improvement in accuracy seen when re-scoring the unrestricted-grammar lattices with the callsign constraints (see section 9.2.2 on page 60).

In conclusion, classification benefited from re-scoring through the application of contextual constraints regardless of the grammar used for the initial scoring. Nevertheless, the improvement in accuracy seen when re-scoring the unrestricted-grammar lattices was much greater than the improvement seen when re-scoring the restricted-grammar lattices: The best classification accuracy while using the unrestricted grammar was roughly half of that as with the restricted grammar (e.g. a WER of 0.26 for the resticted grammar vs. 0.44 for the unrestricted one) while retaining the greater coverage of the unrestricted grammar over the restricted one.

Part IV.

Discussion

10. Interpretation of results

10.1. Grammar comparison

As stated in section 5.3 on page 32, the unrestricted grammar has a significantly greater coverage than the restricted grammar: This allows the recognition of a far greater range of utterances but it also entails that the perplexity of the unrestricted grammar is relatively high compared to that of the restricted grammar.

10.1.1. Baseline accuracy

The baseline accuracy of classification while using the unrestricted grammar was significantly worse than that while using the restricted grammar in regards to WER and SER; This is unsurprising considering the relatively high perplexity of the unrestricted grammar compared to the restricted one [10].

Likewise, the MRR was lower when classifying with the unrestricted grammar compared to the restricted grammar. Although this in itself is unremarkable, these results combined with the relatively few unrecoverable sentences in the corpus suggest that there is significant room for improvement of classification with the unrestricted grammar through knowledge-based lattice re-scoring in comparison to the restricted grammar (see section 8.2.3 on page 51).

10.1.2. Re-scoring accuracy

The claim that the unrestricted grammar leaves more potential for improvement was substantiated by the difference in improvements achieved between re-scoring the restricted and unrestricted grammars, as described in chapter 9 on page 54: When re-scoring the restricted-grammar lattices, a maximum 20% reduction in WER and a maximum 21.80% reduction in SER and a maximum 0.33% increase in MRR was achieved; When re-scoring the unrestricted-grammar lattices, however, an 84.34% reduction in WER and an 82.82% reduction in SER was achieved and a maximum 14.28% increase in MRR was achieved.

10.1.3. Conclusion

In conclusion, classification with the restricted grammar resulted in a lower error rate before any re-scoring is performed. However, the benefits of re-scoring lattices created with such a grammar was much less than that with a relatively unrestricted, wider-coverage grammar.

10.2. Situational knowledge effectiveness

Two types of situational knowledge were used for re-scoring in the domain of ATC: Referential knowledge of valid aircraft callsigns and physical knowledge of valid physical aircraft states given their last state.

10.2.1. Referential knowledge

As explained in section 9.1.1 on page 55, no improvements in classification accuracy were achieved by re-scoring the restricted-grammar lattices with the callsign constraints defined in section 6.2.1 on page 39: This is due to the fact that the restricted grammar has a very narrow coverage of callsigns in comparison to the unrestricted grammar, as described in section 5.3 on page 32. Therefore, the restricted grammar already implicitly constrains callsigns to a significant extent, entailing that the perplexity of callsign classification is relatively low when classifying using the restricted grammar in comparison to when classifying with the unrestricted grammar.

Understandably, then, a large improvement in accuracy was achieved when re-scoring the unrestricted-grammar lattices by applying the callsign constraints, as described in 9.2.1. The effectiveness of these constraints suggest that knowledge of entity reference [32, pp. 729 sqq.] can be highly effective for use in lattice re-scoring.

Referential certainty

The greatest improvement in accuracy while re-scoring with the callsign constraints was when applying the gold standard callsign constraint $c^{\text{goldstd}}_{\text{callsign}}$, the most semantically-specific one. However, this constraint relies on the knowledge of the entity which is denoted by a given utterance: In this case, this is the knowledge of which aircraft to which the command in question is directed. Therefore, the effective implementation of this constraint in a real-world situation may be difficult.

These results suggest that, if a referent is unambiguously specified (e.g. in a different modality), this knowledge can be used to greatly improve accuracy.

Static referential knowledge

Among the constraints encoding referential knowledge, the gold standard callsign constraint $c^{\text{goldstd}}_{\text{callsign}}$ achieved the greatest improvement. However, the possible callsign constraint $c^{\text{possible}}_{\text{callsign}}$ achieved nearly as much improvement, the WER after re-scoring using the possible callsign constraints being only 9.19% higher than after re-scoring using the gold standard callsign constraint (0.55 vs. 0.50, respectively). Likewise, the SER was only 1.95% higher (4.61 vs. 4.52, respectively). The difference between the MRR of re-scoring with the two constraints was even smaller, with the MRR of the possible callsign constraint being 0.08% lower than that of the gold standard callsign (0.965786 vs. 0.966576, respectively).

These results suggest that even by applying much-less restrictive constraints, accuracy can be increased significantly for a domain in which certain referential knowledge is

certain — for example, in this case, the knowledge of which aircraft callsigns are to be expected in an ATC scenario.

Dynamic referential knowledge

The dynamic context-specific callsign constraint $c^{\text{context}}_{\text{callsign}}$ which constrains valid callsigns to those which are present in the 4D-CARMA simulation in a given timeframe achieved the same WER and SER improvements that the possible callsigns $c^{\text{possible}}_{\text{callsign}}$ constraint did. Likewise, there was no significant improvement in MRR, with the MRR after re-scoring with the dynamic context callsign constraint being only 0.007% greater than that after re-scoring with the static possible callsign constraint. This suggests that, given the simulation in which the utterances were made, incorporating this dynamic contextual knowledge is not very effective in reducing accuracy.

However, improvements may be achieved by using this dynamic constraint over the static possible callsign constraint given a more complex ATC scenario, in which the set of possible callsigns is much larger: In the simulation used to generate the test data, there were a total of 31 different aircraft (and thus 31 possible callsigns) and there were a mean 8.76 aircraft in the air at one time (see appendix B on page 81 for details). In a real-world scenario, however, both the number of possible callsigns and the average number of aircraft in the air at one time can be much greater. Nevertheless, while there is no limit to the number of possible callsigns, the average number of aircraft in the air is limited by the capabilities of the control tower(s) controlling the airspace in question. Therefore, this number is in almost all cases much less than the number of possible callsigns that an aircraft may have at any time in the scenario.

Conclusion

In conclusion, the accuracy improvements seen from re-scoring with the callsign constraints shows that contextual referential knowledge can be used for lattice re-scoring: Using knowledge of a static context is highly effective, such as that encoded by the possible callsign constraint $c^{\text{possible}}_{\text{callsign}}$; Likewise, using knowledge of a dynamic context is possible, such as that encoded by the dynamic contextual callsign constraint $c^{\text{context}}_{\text{callsign}}$: Such dynamic knowledge has the potential to be even more effective than static referential knowledge.

However, the benefits of knowledge-based re-scoring can only be enjoyed with a relatively wide-coverage grammar, one which does not already implicitly entail similar constraints on recognition.

10.2.2. Physical knowledge

A small improvement in accuracy was observed after re-scoring both the restricted- and unrestricted-grammar lattices by applying the flight information constraints, defined as linear (in)equalities as described in section 6.2.1 on page 40. Due to the fact that the grammars do not differ in the flight information values they cover, this similarity is unremarkable (cf. listings C.1 to C.2 on pages 88–92).

Physical certainty

As described in sections 9.1.2 on page 56 and 9.2.2 on page 60, the most effective constraint which encodes physical knowledge was the runway heading constraint $c_{\text{hdg}}^{\text{rwy}}$; This constraint encodes the knowledge that, in this simulation, a turn heading always is intended to orient the aircraft towards the single runway in the simulation in preparation for landing. Given a real-word scenario, however, it is likely that trying to apply similar constraints would not be as effective: Firstly, many airspaces in which aircraft arrivals (i.e. landings) must be managed contain multiple runways to which a given aircraft may be directed; Secondly, in a real-world scenario, a TURN command does not (always) entail that an aircraft is to approach a runway which has the same heading, or that an aircraft is to approach any runway at all — for example, many aircraft pass through an airspace without landing in it, and often are issued commands to direct them through the airspace while maintaining separation (see section 3.1 on page 17). Therefore, such a restrictive static constraint is effective in only a relatively constrained domain such as that provided by the simple simulation used in this experiment.

In conclusion, just as the case with referential certainty as described in section 10.2.1 on page 65, certainty about the next physical state can be used for effective re-scoring but such knowledge cannot be certain in many situations.

Static physical knowledge

As described in sections 9.1.2 on page 56 and 9.2.2 on page 60, there was a small improvement in accuracy when re-scoring using the static valid heading constraints $\{c_{\text{hdg}}^{\text{min}}, c_{\text{hdg}}^{\text{max}}\}$. However, this improvement was insignificant when re-scoring the restricted-grammar lattices. Likewise, when re-scoring the unrestricted-grammar lattices, the improvements achieved were small in comparison to those achieved by re-scoring with the callsign constraints (see section 10.2.1 on page 65).

These results suggest that static physical knowledge and reasoning about the physical world can be used for lattice re-scoring: For example, in the case of the static valid heading constraints, a valid heading value is in degrees between 0° and 360°. Although such knowledge about valid headings can also be incorporated into the grammar (and thus language model) itself, others feature long-distance dependencies which cannot be easily encoded in a CFG: For example, in the case of ATC, knowing the minimum and maximum allowed altitude for any given aircraft in a given airspace can be advantageous for classification. However, these values may depend on e.g. the specific type of aircraft in question: For example, a larger aircraft may have a higher minimum allowed altitude than a smaller one.

Therefore, it seems that with more sophisticated static knowledge about the domain, greater improvements can be achieved after re-scoring than with relatively simple knowledge.

Dynamic physical knowledge

The constraints which encode dynamic physical situational knowledge c_{spd} and $\{c_{\text{alt}}, c_{\text{fl}}\}$ were not very effective in improving accuracy during re-scoring, as described in sections 9.1.2 on page 56 and 9.2.2 on page 60. These results suggest that the range of possible values $\{x \in \mathbb{R} : z_\alpha \leq x < h_\alpha\}$ between the value of a given type of flight information h_α and the value of the related concept z_α is simply too large to derive an effective constraint from: For example, out of 411 utterances denoting a DESCEND COMMAND with an ALTITUDE value of 4,000 (feet) (e.g. *Lufthansa six four five descend altitude four thousand feet*), thirteen were classified as denoting a value of 5,000 (feet), e.g. `lufthansa six four five descend altitude five thousand feet`. However, at the time in the scenario in which these utterances were made, the altitude of the given aircraft $h_{\text{fl}}(\beta)$ was often very great: In the simulation log data for the scenario used for creating the lattices, the mean flight level for any aircraft was 127.83 (1,2783 feet) out of a population of n simulation updates:

$$127.83 = \frac{1}{n} \sum_{\substack{h \in H \\ h(\beta) \in h}}^{n} h_{\text{fl}}(\beta) \tag{10.2.1}$$

Given that both grammars restrict altitude values to either 2,000, 3,000, 4,000, or 5,000 feet (cf. listings C.1 to C.2 on pages 88–92), the probability is very low that an erroneous classification will denote an ALTITUDE with a value higher than the current ALTITUDE value $h_{\text{fl}}(\beta)$ for the given aircraft β.

Therefore, it seems that relatively sophisticated physical situational knowledge and reasoning may be required to effect an improvement during re-scoring: Given the restrictiveness of the grammars used and the relatively large physical domain in which the utterances were made, there is little overlap between what can be recognised and what is physically implausible with such simple physical knowledge.

Conclusion

The only flight information constraint which achieved a significant improvement in accuracy during re-scoring was the runway heading constraint $c_{\text{hdg}}^{\text{rwy}}$, which is also the most restrictive (and is perhaps too restrictive): The other constraints were not restrictive enough to effect a significant change during re-scoring.

In conclusion, it seems that the flight information (in)equality constraints, which encode physical situational knowledge in ATC, do not constrain the values of their respective concepts enough to achieve a significant difference in ranking during re-scoring of either the restricted- or unrestricted-grammar lattices. Although such knowledge has the potential to be of great use during re-scoring. Therefore, it seems that more-sophisticated methods of reasoning about physical context must be used in order to effectively use physical knowledge for re-scoring.

10.2.3. Conclusion

Improvements in classification accuracy were achieved after re-scoring using situational knowledge, both of valid referents (section 10.2.1 on page 65) and of valid physical states (section 10.2.2 on page 66). When using either knowledge source, the more certain the knowledge used, the more effective re-scoring using the knowledge was. However, given a relatively large domain, such certainty may not be possible. Likewise, the effectiveness of re-scoring using dynamic contextual knowledge is limited by the amount that reasoning about the domain can reduce the amount of possible next states in the context given the current state.

However, in general, referential knowledge was more effective than physical knowledge for use in re-scoring: Re-scoring using the set of constraints which encodes the least-specific referential knowledge $\{c^{\text{possible}}_{\text{callsign}}\}$ (the possible callsign constraint) achieved an 80.43% reduction in WER as opposed to 2.49% when re-scoring with the set of constraints which encode the most specific physical knowledge. Similarly, there was a large difference between the SERs and MERs of the two re-scoring tests. These results suggest that referential knowledge is a much simpler knowledge source to take advantage of for re-scoring: In layman's terms, it is much simpler to (dis)prove the existence of an entity than to (dis)prove the physical state of that entity.

10.3. Classification errors

The vast majority of word errors observed were from classifying numbers, both for the restricted grammar before and after re-scoring (see sections 8.1.3 on page 49 and 8.2.3 on page 51, respectively) and for the unrestricted grammar before and after re-scoring (see sections 9.1.3 on page 57 and 9.2.3 on page 61, respectively). When using the unrestricted grammar, these classification errors were made both for numbers which are part of flight numbers (e.g. *Air France four one eight* classified as `air_france five one eight`) and for those which are part of flight information values (e.g. *Lufthansa six four five descend altitude four thousand feet* classified as `lufthansa six four five descend altitude five thousand feet`).

The callsign constraints effected a significant reduction of such errors in classifying callsigns; However, despite the similarity of the classification errors, similar improvements were not seen for the classification of flight information after re-scoring with the flight information constraints. This suggests that, even though the physical knowledge constraints did not achieve significant improvements in accuracy in this experiment, there is great potential for such constraints to effect such an improvement. Given better-engineered physical constraints, similar improvements in flight information classification may be achieved.

10.4. Conclusion

By re-scoring the lattices created with the unrestricted grammar, a minimum WER of 0.44 (an 84.34% reduction over the baseline) was achieved, while re-scoring the restricted-grammar lattices achieved a minimum WER of 0.24 (a 20% reduction over the baseline). Similar improvements in SER were achieved, and, similarly, there were significant improvements in MRR. Although the minimum WER and SER of classification using the unrestricted-grammar lattices after re-scoring was roughly 50% greater than that using the resticted-grammar lattices, these significant improvements were achieved while retaining the relatively wide coverage coverage of the unrestricted grammar: If perplexity is used as a comparison of grammar coverage, the best results while using the unrestricted grammar were only roughly 50% greater than that of the restricted grammar despite that the perplexity of the unrestricted grammar is roughly 1,383 times that of the restricted grammar (see section 5.3 on page 32).

In conclusion, by encoding contextual knowledge as constraints in a lattice re-scoring task, the benefits of a wide-coverage grammar can be used while minimising classification errors.

11. Implications of results

11.1. Lattice re-scoring

There are already many successful methods for lattice (and n-best list) re-scoring, both using data-driven and symbolic methods [cf. 40, 41, 49, 61, 63]. In this experiment, it has been shown that dynamic contextual knowledge can be used for re-scoring by encoding them as linear and logical constraints. In this way, lattices can re-scored according to their specific context at runtime.

11.1.1. Dynamic vs. static knowledge

There have already been successful attempts at knowledge-based lattice re-scoring, such as using articulatory information [cf. 40, 61]. However, in this experiment, the knowledge used to successfully re-scored lattices was not only static in relation to the domain of classification but also dynamic: For example, the articulatory knowledge used by Li, Tsao and Lee [40] is robust to speech variations [40, p. I-837]. Likewise, in the case of a single ATC scenario (such as in the context of a given airspace on a given day), the knowledge about which aircraft callsigns can be expected is static for the given domain (see section 10.2.1 on page 65). However, the callsigns which can be expected given a current time in the scenario is dependent on the time; No analogously-dynamic knowledge was incorporated into the articulatory classifier used by Li, Tsao and Lee [40]. Although the results of using such dynamic constraints were not significantly better than similar static constraints in this experiment, they have the potential to achieve a large improvement over their static counterparts given a much larger domain (see section 10.2.1 on page 66).

Nevertheless, as shown by the relative ineffectiveness of the physical knowledge constraints, such constraints (whether static or dynamic) must be engineered in a way that they are very specific semantically: The less specific they are, the lower the probability that they will affect re-scoring (see section 10.2.2 on page 68).

11.1.2. Hand-crafted constraints vs. data-driven methods

Many lattice (and n-best list) re-scoring methods employ data-driven methods, such as by re-scoring with more-sophisticated language models [cf. 41] or by minimising expected word error based on posterior probabilities [cf. 63]. Therefore, such methods are not available for re-scoring in domains for which there is little training data (such as that of ATC in this experiment).

However, the results of this experiment show that large amounts of data are not needed to improve classification through re-scoring: The constraints in this experiment were hand-crafted based on linguistic and semiotic theory as well as general world knowledge. This experiment is not the first attempt at improving data-driven classification through symbolic methods [cf. 6, 7, 33]; However, in encoding dynamic knowledge as explained in the previous section, it uses a novel approach to utilise knowledge which cannot be easily incorporated as a set of (local) features into a probabilistic model.

11.2. Probablistic classification

As Moore [47] expressed, data-driven methods for improving ASR accuracy — and by extension the accuracy of any probabilistic classifier — is limited by the very premise on which they are based: The availability of (typically very large amounts of) training data. However, this experiment has shown that it is possible to incorporate contextual knowledge into a classification system to improve accuracy without modifying the weighted feature function-based classification $f_\Phi(x, y)$ represented by the probabilistic classifier itself (see chapter 6 on page 38).

11.2.1. Classification in a multi-modal context

Incorporating multi-modal knowledge in ASR is not a new topic [cf. 26, 58, 66, 76, 78]. However, this experiment has shown that it is possible to utilise this knowledge for improving off-the-shelf ASR systems like *Millennium* [73]: The probability distribution used for the weighted feature function-based classification of *Millennium* $f_\Phi(x, y)$ was left unmodified and the results were simply re-scored as a linear combination of the original score $f_\Phi(x, y)$ and the constraint-based knowledge score $f_C(x, y)$ for a hypothesis y (see chapter 6 on page 38).

Therefore, with this method, it is possible to apply contextual knowledge to any ASR system — and by extension to any probabilistic classifier — for improving classification accuracy.

11.2.2. Classification in a dynamic context

As described in section 11.1.1 on the preceding page, several of the constraints used in this experiment encode dynamic contextual knowledge which would be difficult to infer from training data as local features. There are many different methods for incorporating dynamicism into probabilistic classification, such as incorporating multiple context-sensitive probability distributions [cf. 41, 57, 62]. What is interesting about the methods used in this experiment, however, is that it is non-probabilistic and based largely on symbolic reasoning about the current context state. Therefore, constraints based on more-sophisticated reasoning systems could be used in place of the simple constraints applied in this experiment [cf. 55, 72].

11.3. Unresolved issues

Although the experiment was largely successful in re-scoring using static and dynamic knowledge, there were a number of issues which need to be addressed: Firstly, there was a lack of in-domain training data for the ASR acoustic and language models which negatively impacted classification accuracy; Secondly, the KSs utilised in this experiment for re-scoring could not be properly taken advantage of due to the simplicity of the reasoning system used, namely the application of simple declarative logical constraints.

11.3.1. Modelling

The vast majority of classification errors were of numbers for either grammar both before and after re-scoring, as described in section 10.3 on page 69. Although many of these errors were corrected after re-scoring with the callsign constraints (see section 9.2.1 on page 59), the majority of the the flight information errors were not corrected through re-scoring and so the word error count for numbers was still relatively high. This indicates a great discrepancy between the ASR speech and language models and the true probability distribution in this area (see sections 5.3 on page 32 and 5.4 on page 34). It seems that the very small amount of training data used, which was largely derived from general sources like new broadcasts and lecture recordings, was not suitable for classification in the domain of ATC.

It seems that at least a certain degree of quality in-domain training data is required regardless of the methods used to minimise problems related to a small amount of in-domain training data. Therefore, the accuracy of the ATC ASR system described in this thesis could have been much better given a proper set of training data for ATC.

11.3.2. Knowledge-based reasoning

As described in section 10.2.3 on page 69, the accuracy improvements seen after re-scoring using physical situational knowledge was relatively small, even though the improvement seen from re-scoring using referential knowledge was relatively large (see section 10.2.1 on page 66). The physical knowledge encoded by such flight information constraints was relatively simple, and so it seems that a more-sophisticated reasoning system may be required in order to achieve accuracy improvements based on physical knowledge.

Alternatively, other forms of reasoning may be employed, such as by predicting future user behaviour: For example, Schäfer [58] as well as Young, Ward and Hauptmann [77] incorporated prediction of future commands the user (e.g. in this case, the air traffic controller) may give.

In conclusion, it seems that simple declarative constraints in themselves are not sophisticated enough to encode enough dynamic situational knowledge to be effective in re-scoring.

11.4. Conclusion

In conclusion, this experiment has shown that it is possible to improve classification accuracy through lattice re-scoring by using both static and dynamic KSs in a given context, by encoding these KSs as declarative constraints in constraint programming $f_C(x, y)$ (see chapter 6 on page 38). Despite the insufficiency of the training data supplied for the ASR system for to the domain at hand, significant improvements in classification accuracy were achieved through re-scoring. This allows the usage of a much wider-coverage grammar without the increases in classification error which are usually associated with such a higher-perplexity grammar.

The dynamic KSs did not achieve a significant improvement over their static counterparts given the dataset used in this experiment (namely the simulated ATC scenario used). However, it has been demonstrated that utilising such dynamic contextual knowledge has the potential to achieve great improvements in accuracy given a more complex domain and and more sophisticated capabilities of the system to reason about the context.

12. Conclusion

In this thesis, it has been demonstrated that the accuracy of probabilistic classification (in this case, ASR) can be greatly improved through the utilisation of knowledge about the context in which the classifier is being used (in this case, in a simulated ATC scenario). The specific KSs used (in this case, referential and physical knowledge) may be not only static in nature but also dynamic, being dependent on the state of the context at the time of the input being classified. Such knowledge may also be multi-modal, in that they may utilise not only the input to be classified itself but also various other sources of information: For example, in this experiment, a model of an airspace created by an ATC simulator was used to improve the accuracy of classifying ATC commands which were situated in the context of the ATC simulation.

Furthermore, this knowledge can be utilised for improving accuracy without encoding it as features into the probabilistic model being used for classification: In this experiment, the accuracy of a previously-trained ASR system was improved significantly through lattice re-scoring, first calculating the original probabilistic score $f_\Phi(x, y)$ for each hypothesis y and then re-scoring it by adding a knowledge score $f_C(x, y)$ defined as a dynamic weighted CSP encoding knowledge from various context-dependent KSs.

The weighted constraint penalty score function $f_C(x, y)$ used in this experiment was based on symbolic reasoning and so no additional training data was needed: This means that such methods can be used for any domain and for any context about which knowledge-based reasoning is possible, regardless of the amount of in-domain training data available. Therefore, this method is ideal for domains for which there is little available data (such as in ATC): Probabilistic classification would already be less accurate for such domains due to data sparsity issues, and so using such methods would have much more potential for improving accuracy in such domains than in those for which there is a large amount of in-domain training data. Although the accuracy of classification could be improved further given better inference (as shown by the errors in classifying numbers), contextual knowledge-based re-scoring is nevertheless an effective method for improving accuracy which can be employed in a variety of situations, including those for which it may be difficult to infer an accurate probabilistic model beforehand.

Therefore, it has been shown that such a method can encode complex, high-level knowledge which may be difficult to encode as local features in a linear model, which is usually used in classification for tractability reasons. In conclusion, dynamic contextual knowledge-based re-scoring is a platform which provides the ability to take advantage of many more KSs than would be possible by using only standard methods for statistical inference; Such abilities hopefully bring classification technology one step closer to the 'living system-inspired' systems described by Moore [47].

13. Future work

In order to develop the system described in this thesis from a proof of concept to a fully-functional (prototype) system, three areas must be addressed: Firstly, a proper corpus of situational ATC recordings must be made and other efforts must be made to improve acoustic and language modelling; Secondly, a more-sophisticated reasoning system must be developed to take advantage of the dynamic knowledge sources at hand; Thirdly, an efficient real-time implementation of the combined weighted feature function- and constraint-based classification score function $f_{\Phi,C}(x, y) \equiv f_{\Phi}(x, y) + f_{C}(x, y)$ must be developed.

13.1. Modelling

As described in section 11.3.1 on page 73, the acoustic and language models were not a good fit for the data seen in the domain of ATC. Therefore, a production-quality system would require a quality set of training data of substantial size for the domain at hand, more sophisticated methods for domain adaptation or both.

13.1.1. ATC corpus

Firstly, a corpus of recordings in ATC should be established, in which each recording constitutes an ATC command or list of commands which is *situated* in context [cf. 35, pp. 311–316]. In other words, each recording must be annotated not only with its correct label classification (e.g. *Adria two three reduce speed two two zero knots* ≅ `adria two three reduce speed two two zero knots`) but also with the state of the ATC scenario in which it was executed: This includes information not only about the aircraft to which the command was issued (e.g. *Adria two three* ≅ `ADR23` in the previous example), but rather as much information about the entire contextual state as can be supplied; The more situational data which can be supplied, the more sophisticated the knowledge-based constraints that can be derived therefrom.

Just as with a normal ASR training dataset, however, the corpus should include as many contexts as possible: Specifically to the domain of ATC, this dataset should include recordings which e.g. feature significant noise from various sources, as described in section 3.1.3 on page 18. However, such a corpus should be created using only trained ATC personnel: When collecting data using the methods described in chapter 4 on page 27 with laypersons, a large amount of recording data had to be removed because it was deemed of insufficient quality. Moreover, instructing laypersons in proper ICAO phraseology and how to use the AMAN software proved to be strenuous and time-consuming

even given the simplified scenario used for this experiment; Therefore, the training- to recording-time ratio was rather poor when using non-ATC personnel as participants.

13.1.2. Modelling techniques

Either as a replacement for a dedicated ATC corpus or (ideally) as an augment to it, more sophisticated techniques should be used for inferring probability distributions from the training data available, especially in regards to numbers, the biggest problem area for classification in the experiment. For example, it may be possible to train multiple state-dependent sub-models for recognising specific parts of an utterance (especially for numbers) [cf. 75].

13.2. Knowledge-based reasoning

The simple declarative constraints used in this experiment did not encode enough dynamic situational knowledge to be significantly effective in improving accuracy, as described in section 11.3.2 on page 73. Therefore, in order to take advantage of the dynamic situational knowledge available in the domain of ATC, a reasoning system should be developed for reasoning about states of an ATC scenario. Specifically, in addition to general logical/semantic reasoning [cf. 55], the system should incorporate reasoning about the physical environment [cf. 72] which constitutes the airspace under control during ATC.

13.3. (Re-)scoring implementation

The lattice re-scoring method used in this experiment (described in section 5.5 on page 34) is not meant for real-time classification: The implementation of the algorithm described in appendix D on page 95 could be improved greatly. Moreover, re-scoring every hypothesis in a word lattice $\mathcal{T}^{\text{word}}$ is neither computationally feasible nor required: Rather, these knowledge-based constraints can be applied during the phone lattice search involved in word lattice projection (see section 5.5.3 on page 35). Not all of these constraints are local, and therefore some can only be applied after search has been completed; Nevertheless, by incorporating this contextual knowledge into the original word lattices themselves, the best-scoring classification $\hat{h} \equiv \arg\min_{y \in \mathcal{Y}} f_\Phi(x, y) + f_C(x, y)$ for an input x including the knowledge-based score $f_C(x, y)$ can be computed with relatively little extra computation.

13.4. Conclusion

Although the system designed for this experiment is not a robust, efficient real-time system, the principles upon which it was designed can be easily used to create a fully-functional system with a better set of training data, more sophisticated, production-quality knowledge reasoning, and some modification of the best-hypothesis search used by the ASR module used.

It is unknown how such a system will compare in accuracy and performance to more complex, sophisticated systems which incorporate multiple layers of probabilistic predictions (such as that of Schäfer [58] and Young [76]). However, such a system as proposed would be highly modular and would hopefully lose fewer of its advantages in situations which cannot be modelled beforehand due to their non-existence in the training data at hand.

Part V.

Appendices

A. Acknowledgements

I would like to acknowledge the many people whose assistance, expertise and advice this project was critical to the success of this project.

Firstly, I would like to thank Prof. Dr Dietrich Klakow for offering the research topic itself and the formal supervision of the MSc thesis, putting his trust in me to deliver as part of such a high-profile project. In this regard, I would also like to thank Prof. Dr Hartmut Helmke, Jürgen Rataj, Heiko Ehr, Oliver Ohneiser and Meilin Schaper for providing the 4D-CARMA and RadarVision software and miscellaneous support related to them. I would also like to thank the Institute of Flight Guidance of the German Aerospace Center (DLR) in general for the research partnership between the Institute and the Department of Spoken Language Systems at Saarland University, giving me the opportunity to do this research.

I am obliged to Friedrich Faubel for his expertise and assistance with speech recognition in general and with word lattices in specific; Also deserving of mention is his patience and general day-to-day research guidance and mentoring, which helped me immensely in brainstorming and defining my research goals. Likewise, the advice of Dr Grzegorz Chrupała proved invaluable in this regard.

I am also grateful for my family (especially my mother) for their support, as well as for my coursemates, colleages and friends who helped make Saarbrücken a home for me. Finally, I would like to dedicate this work to my grandmother and the memory of my grandfather for being the strongest people I have ever known.

B. Corpus details

Table B.1.: The number of utterances each individual speaker recorded in the corpus sorted by native language and gender.

	Name	Native Language	Gender	Count
	xyl	German	Male	94
	phi	German	Male	65
	sen	German	Male	61
	fri	German	Male	49
	kla	German	Male	40
	ben	German	Male	35
	mir	German	Female	86
	mja	German	Female	53
	tsh	English	Male	106
	sup	English	Male	88
	ric	English	Male	86
	nik	Greek	Male	72
	kat	Greek	Female	46
	ice	Malayalam	Male	94
	mna	Romanian	Female	61
	bor	Russian	Male	71
TOTAL	16	6	2	1107
MEAN	—	—	—	69.1875

Table B.2.: The total number of utterances recorded by speakers for each native language and gender.

TOTALS

	Gender		TOTAL	MEAN
Native Language	Male	Female		
German	344	129	473	236.5
English	280	0	280	140
Greek	72	46	118	59
Malayalam	94	0	94	47
Romanian	0	61	61	30.5
Russian	71	0	71	35.5
TOTAL	871	236	1107	553.5
MEAN	$145\frac{2}{3}$	$39\frac{1}{3}$	184.5	92.25

Table B.3.: Corpus recording session lengths, in simulation time, i.e. the time from the start of the simulation until the end of recording, rather than summing the lengths of each individual recording.

Name	Length	Count	Count/min
mir	0:54:07	86	1.59
xyl	0:53:39	94	1.75
ice	0:51:56	94	1.81
tsh	0:49:37	106	2.15
ric	0:46:59	86	1.83
sup	0:46:37	88	1.89
nik	0:45:21	72	1.59
bor	0:43:27	71	1.63
mja	0:38:51	53	1.36
sen	0:34:52	61	1.75
phi	0:33:04	65	1.97
mna	0:31:39	61	1.93
ben	0:30:20	35	1,16
kla	0:27:30	40	1.47
kat	0:26:26	46	1.74
fri	0:22:47	49	2.15
TOTAL	10:37:12	1107	—
MEAN	0:39:49.5	69.1875	1.75

Table B.4.: The callsigns for each aircraft which was present in the experiment ATC simulation with its corresponding airline designator and flight number in natural language; As explained in section 4.2.7 on page 30, the participants were instructed to pronounce `3` ≅ *three* as *tree* /ˈtɹiː/ and `9` ≅ *nine* as *niner* /ˈnaɪ.nə(ɹ)/.

	Displayed	Airline	Flight number
	`ADR23`	*Adria*	*two three*
	`ADR8687`	*Adria*	*eight six eight seven*
	`AF295`	*Air France*	*two nine five*
	`AF418`	*Air France*	*four one eight*
	`AF4867`	*Air France*	*four eight six seven*
	`BMA3685`	*Midland*	*three six eight five*
	`BMA419`	*Midland*	*four one nine*
	`BWA364`	*Speedbird*	*three six four*
	`BWA480`	*Speedbird*	*four eight zero*
	`BWA581`	*Speedbird*	*five eight one*
	`DLH123`	*Lufthansa*	*one two three*
	`DLH1342`	*Lufthansa*	*one three four two*
	`DLH321`	*Lufthansa*	*three two one*
	`DLH439`	*Lufthansa*	*four three nine*
	`DLH4689`	*Lufthansa*	*four six eight nine*
	`DLH495`	*Lufthansa*	*four nine five*
	`DLH645`	*Lufthansa*	*six four five*
	`DLH780`	*Lufthansa*	*seven eight zero*
	`DLH869`	*Lufthansa*	*eight six nine*
	`DLH8904`	*Lufthansa*	*eight nine zero four*
	`FIN5678`	*Finnair*	*five six seven eight*
	`GAO249`	*Golden*	*two four nine*
	`GAO581`	*Golden*	*five eight one*
	`IBE628`	*Iberia*	*six two eight*
	`IBE985`	*Iberia*	*nine eight five*
	`KLM498`	*KLM*	*four nine eight*
	`KLM605`	*KLM*	*six zero five*
	`MAH410`	*Malev*	*four one zero*
	`MAH5489`	*Malev*	*five four eight nine*
	`QFA6`	*Qantas*	*six*
	`QFA764`	*Qantas*	*seven six four*
TOTAL	31	11	31

C. Language model details

In the experiments, two different grammars were used: One is relatively restricted and covers largely only the specific ATC simulation used to make the ATC corpus in the experiments (see section 4 on page 27 and appendix B on page 81), while the other is relatively unrestricted and covers a much greater number of possible aircraft callsigns.

The difference between the two grammars is in the coverage of callsigns: The restricted grammar covers only 35 unique callsigns: The 31 in the simulation (see table B.4 on the previous page) plus the callsigns in table C.2 on page 88), while the unrestricted grammar covers 244,220 unique callsigns, with 11 airline designators · (10 one-digit flight numbers + 100 two-digit flight numbers + 1,000 three-digit numbers + 10,000 four-digit flight numbers) · 2 for one variant with `__pause__` (e.g. `<callsign> <airline> adria </airline> __pause__ <flightnumber> two three </flightnumber> </callsign>`) and one without (e.g. `<callsign> <airline> adria </airline> <flightnumber> two three </flightnumber> </callsign>`).

C.1. Grammar perplexity

In order to compare the two grammars used in this experiment, the perplexity of the grammars was defined using Shannon entropy $PP(S) \equiv b^{H_b(S)}$ [43, p. 78, 60], instead of using cross-entropy, as is often done [cf. 32, pp. 150–151]. Furthermore, the probability $P(s)$ of a sentence able to be produced by the grammar $s \in S$ is defined as the MLE of the sentence $P(s) \equiv MLE(s)$, where $MLE(s) \equiv N(s)/N = 1/|S|$ and $N(s) = 1$ [43, pp. 197 sqq.]. By defining the probability of a sentence as the MLE, the perplexity of a grammar $PP(S)$ is equal to the size of the set of all possible sentences able to be produced by the grammar:

$PP(S) = |S|$ *where* $P(s) \equiv MLE(s)$, $N(s) = 1$.

$$
\begin{aligned}
PP(S) &\equiv b^{H_b(S)} && \text{[43, p. 78]} \\
H_b(S) &\equiv -\sum_{s \in S} P(s) \log_b P(s) && \text{[43, p. 61]} \\
P(s) &\equiv MLE(s) \\
MLE(s) &\equiv N(s)/N && \text{[43, pp. 197 sqq.]} \\
MLE(s) &= 1/|S| && \text{where } N(s) = 1 \\
H_b(S) &= -\sum_{s \in S} \frac{1}{|S|} \log_b \frac{1}{|S|} \\
&= -\log_b \frac{1}{|S|} \\
&= \log_b |S| \\
PP(S) &= b^{\log_b |S|} \\
\log_b^{-1} x &= b^x \\
PP(S) &= |S|
\end{aligned}
\tag{C.1.1}
$$

□

C.2. Vocabulary

Both grammars have the same vocabulary, each word thereof being mapped to a unique integer which is used to represent the words in the lattices.

C.2.1. Concept labels

The vocabulary Γ used by the two grammars contains symbols which not only correspond to natural-language words such as `descend` ≅ *descend* and `air_france` ≅ *Air France* (see section 4.2.4 on page 29) but also to a set of symbols representing semantic concepts according to template-filler semantics [cf. 71]. These symbols facilitate machine reading of the ASR output during natural language understanding (see section E.1 on page 100 for more details): For each concept α (e.g. AIRLINE, FLIGHTNUMBER and ALTITUDE), there is a start tag $start\_tag(\alpha) \in \Gamma$ and end tag $end\_tag(\alpha) \in \Gamma$ — The semantic content of a concept α is entailed by the terminal categories (i.e. labels) between these tags:*

$$
\forall \forall \underset{\gamma_i \gamma_k i<j<k}{\forall} \gamma_j \rightarrow \alpha \qquad \text{where } \gamma_i = \text{start\_tag}(\alpha), \gamma_k = \text{end\_tag}(\alpha) \tag{C.2.1}
$$

*In this implementation, the grammar vocabulary Γ, which is the set of symbols output by the ASR system, is a sub-set of the set of ASR input symbols $\Gamma \subset \Sigma$: The relative complement of the set of output symbols in the set of input symbols $\Sigma \setminus \Gamma$ is the set of ASR phone symbols.

In this implementation, these symbols are in the form of XML tags, the name of which corresponds to the concept name: For example, a machine-readable label sequence denoting the callsign *Adria two three* is `<callsign> <airline> adria </airline> <flightnumber> two three </flightnumber> </callsign>`. Due to the fact that such concept tags may be nested, concepts may form a concept tree, where any concept is entailed by its sub-concepts: For instance, in the previous example, an airline concept AIRLINE and the symbol `adria` which entails it in conjunction with a flight number concept FLIGHTNUMBER and the symbol sequence `two three` which entails it transitively entail a callsign concept CALLSIGN:

$$\text{CALLSIGN} \leftrightarrow \text{AIRLINE} \wedge \text{FLIGHTNUMBER} \tag{C.2.2}$$

These machine-readable concept tags are used for parsing concepts and the semantic information they denote during semantic interpretation while re-scoring a hypothesis based on contextual information (see appendix E on page 100 for details).

Table C.1.: The vocabulary used by the ASR grammars, which maps each word string to a unique integer label.

Word	Integer
eps	0
<s>	1
</s>	2
%	3
adria	4
air_france	5
<airline>	6
</airline>	7
airspeed	8
altitude	9
<altitude>	10
</altitude>	11
approach	12
by	13
<callsign>	14
</callsign>	15
cleared	16
</command>	17
<command_type="cleared">	18
<command_type="contact">	19
<command_type="descend">	20
<command_type="reduce">	21
<command_type="turn">	22
<command_type="turncleared">	23

Table C.1.: Grammar vocabulary (continued)

Word	Integer
contact	24
descend	25
<direction>	26
</direction>	27
eight	28
feet	29
finn_air	30
five	31
flight	32
<flightlevel>	33
</flightlevel>	34
<flightnumber>	35
</flightnumber>	36
four	37
golden	38
heading	39
<heading>	40
</heading>	41
hundred	42
iberia	43
ils	44
klm	45
knots	46
left	47
level	48
lufthansa	49
malev	50
midland	51
nine	52
ninety	53
on	54
one	55
__pause__	56
qnh	57
<qnh>	58
</qnh>	59
qantas	60
reduce	61
right	62
runway	63

Table C.1.: Grammar vocabulary (continued)

Word	Integer
<runway>	64
</runway>	65
seven	66
six	67
speed	68
<speed>	69
</speed>	70
speedbird	71
thousand	72
three	73
tower	74
turn	75
two	76
zero	77
TOTAL	78

C.3. Restricted grammar

Table C.2.: The aircraft callsigns which are covered by the restricted grammar but are not in the ATC simulation (see table B.4 on page 83 above).

	Displayed	Airline	Flight number
	DLH765	*Lufthansa*	*seven six five*
	DLH9879	*Lufthansa*	*nine eight seven nine*
	FIN4321	*Finnair*	*four three two one*
	GAO6987	*Golden*	*six nine eight seven*
TOTAL	4	3	4
∪ simulation callsigns	35	11	35

Listing C.1: The restricted grammar used for generating lattices, in Nuance grammar format as used by the ASR system [48].

```
1 .SENTENCE [
2   (CALLSIGN ?PAUSE COMMAND)
3 ]
4
5 PAUSE [
6   __pause__
7 ]
8
```

```
CALLSIGN [
  ( <callsign> [
    ( <airline> qantas </airline> <flightnumber> six </flightnumber> )
    ( <airline> qantas </airline> <flightnumber> seven six four
      </flightnumber> )
    ( <airline> lufthansa </airline> <flightnumber> one two three
      </flightnumber> )
    ( <airline> lufthansa </airline> <flightnumber> four nine five
      </flightnumber> )
    ( <airline> lufthansa </airline> <flightnumber> three two one
      </flightnumber> )
    ( <airline> lufthansa </airline> <flightnumber> eight six nine
      </flightnumber> )
    ( <airline> lufthansa </airline> <flightnumber> four six eight nine
      </flightnumber> )
    ( <airline> lufthansa </airline> <flightnumber> one three four two
      </flightnumber> )
    ( <airline> lufthansa </airline> <flightnumber> four three nine
      </flightnumber> )
    ( <airline> lufthansa </airline> <flightnumber> nine eight seven
      nine </flightnumber> )
    ( <airline> lufthansa </airline> <flightnumber> six four five
      </flightnumber> )
    ( <airline> lufthansa </airline> <flightnumber> seven eight zero
      </flightnumber> )
    ( <airline> lufthansa </airline> <flightnumber> seven six five
      </flightnumber> )
    ( <airline> lufthansa </airline> <flightnumber> eight nine zero four
      </flightnumber> )
    ( <airline> midland </airline> <flightnumber> four one nine
      </flightnumber> )
    ( <airline> midland </airline> <flightnumber> three six eight five
      </flightnumber> )
    ( <airline> speedbird </airline> <flightnumber> five eight one
      </flightnumber> )
    ( <airline> speedbird </airline> <flightnumber> three six four
      </flightnumber> )
    ( <airline> speedbird </airline> <flightnumber> four eight zero
      </flightnumber> )
    ( <airline> air_france </airline> <flightnumber> four one eight
      </flightnumber> )
    ( <airline> air_france </airline> <flightnumber> two nine five
      </flightnumber> )
    ( <airline> air_france </airline> <flightnumber> four eight six
      seven </flightnumber> )
    ( <airline> iberia </airline> <flightnumber> nine eight five
      </flightnumber> )
    ( <airline> iberia </airline> <flightnumber> six two eight
      </flightnumber> )
    ( <airline> klm </airline> <flightnumber> four nine eight
      </flightnumber> )
    ( <airline> klm </airline> <flightnumber> six zero five
      </flightnumber> )
    ( <airline> malev </airline> <flightnumber> four one zero
```

```
          </flightnumber> )
38      ( <airline> malev </airline> <flightnumber> five four eight nine
          </flightnumber> )
39      ( <airline> adria </airline> <flightnumber> two three
          </flightnumber> )
40      ( <airline> adria </airline> <flightnumber> eight six eight seven
          </flightnumber> )
41      ( <airline> finn_air </airline> <flightnumber> five six seven eight
          </flightnumber> )
42      ( <airline> finn_air </airline> <flightnumber> four three two one
          </flightnumber> )
43      ( <airline> golden </airline> <flightnumber> five eight one
          </flightnumber> )
44      ( <airline> golden </airline> <flightnumber> six nine eight seven
          </flightnumber> )
45      ( <airline> golden </airline> <flightnumber> two four nine
          </flightnumber> )
46    ] </callsign> )
47  ]
48
49  NUMBER [
50    (one)
51    (two)
52    (three)
53    (four)
54    (five)
55    (six)
56    (seven)
57    (eight)
58    (nine)
59    (zero)
60  ]
61
62  COMMAND [
63    (<command_type="reduce"> REDUCE </command>)
64    (<command_type="descend"> DESCEND </command>)
65    (<command_type="turn"> TURN </command>)
66    (<command_type="cleared"> CLEARED </command>)
67    (<command_type="contact"> CONTACT </command>)
68  ]
69
70  REDUCE [
71    (reduce [speed airspeed] SPEED knots)
72  ]
73
74  SPEED [
75    ( <speed>
76      [
77        ( one [two three four five six seven eight nine] zero )
78        ( two NUMBER zero )
79        ( three [zero one two three four five] zero )
80        ( [one two three] hundred )
81      ]
82    </speed> )
```

```
]

DESCEND [
  ( descend altitude ALTITUDE feet ?BYQNH )
  ( descend flight level FLIGHT_LEVEL )
]

BYQNH [
  (by qnh QNH)
]

QNH [
  ( <qnh> one zero one NUMBER </qnh> )
]

ALTITUDE [
  ( <altitude> [two three four five] thousand </altitude> )
]

FLIGHT_LEVEL [
  ( <flightlevel>
    [
      ( [one two] NUMBER zero )
      ( three [one two zero] zero )
      ( [six seven eight nine] zero )
      ( [one two three] hundred )
    ]
  </flightlevel>)
]

TURN [
  (turn DIRECTION heading HEADING)
]

DIRECTION [
  ( <direction> left </direction> )
  ( <direction> right </direction> )
]

HEADING [
  (<heading> NUMBER NUMBER zero </heading>)
]

CLEARED [
  (cleared ?on ils approach ?runway RUNWAY)
]

RUNWAY [
  (<runway> NUMBER NUMBER [left right] </runway>)
]

CONTACT [
  (contact tower one ninety nine)
]
```

C.4. Unrestricted grammar

Listing C.2: The unrestricted grammar used for generating lattices, in Nuance grammar format as used by the ASR system [48].

```
.SENTENCE [
  (CALLSIGN ?PAUSE COMMAND)
]

PAUSE [
  __pause__
]

CALLSIGN [
  ( <callsign> AIRLINE ?__pause__ FLIGHT_NUMBER </callsign> )
]

AIRLINE [
  (<airline> adria </airline>)
  (<airline> air_france </airline>)
  (<airline> finn_air </airline>)
  (<airline> golden </airline>)
  (<airline> iberia </airline>)
  (<airline> klm </airline>)
  (<airline> lufthansa </airline>)
  (<airline> malev </airline>)
  (<airline> midland </airline>)
  (<airline> qantas </airline>)
  (<airline> speedbird </airline>)
]

FLIGHT_NUMBER [
  (<flightnumber> NUMBER </flightnumber>)
  (<flightnumber> NUMBER NUMBER </flightnumber>)
  (<flightnumber> NUMBER NUMBER NUMBER </flightnumber>)
  (<flightnumber> NUMBER NUMBER NUMBER NUMBER </flightnumber>)
]

NUMBER [
  (one)
  (two)
  (three)
  (four)
  (five)
  (six)
  (seven)
  (eight)
  (nine)
  (zero)
]
```

```
46
47  COMMAND [
48    (<command_type="reduce"> REDUCE </command>)
49    (<command_type="descend"> DESCEND </command>)
50    (<command_type="turn"> TURN </command>)
51    (<command_type="turn"> TURN </command> <command_type="cleared">
          CLEARED </command>)
52    (<command_type="cleared"> CLEARED </command>)
53    (<command_type="contact"> CONTACT </command>)
54  ]
55
56  REDUCE [
57    (reduce [speed airspeed] SPEED knots)
58  ]
59
60  SPEED [
61    ( <speed>
62      [
63        ( one [two three four five six seven eight nine] zero )
64        ( two NUMBER zero )
65        ( three [zero one two three four five] zero )
66        ( [one two three] hundred )
67      ]
68    </speed> )
69  ]
70
71  DESCEND [
72    ( descend altitude ALTITUDE feet ?BYQNH )
73    ( descend flight level FLIGHT_LEVEL )
74  ]
75
76  BYQNH [
77    (by qnh QNH)
78  ]
79
80  QNH [
81    ( <qnh> one zero one NUMBER </qnh> )
82  ]
83
84  ALTITUDE [
85    ( <altitude> [two three four five] thousand </altitude> )
86  ]
87
88  FLIGHT_LEVEL [
89    ( <flightlevel>
90      [
91        ( [one two] NUMBER zero )
92        ( three [one two zero] zero )
93        ( [six seven eight nine] zero )
94        ( [one two three] hundred )
95      ]
96    </flightlevel>)
97  ]
98
```

```
TURN [
  (turn DIRECTION heading HEADING)
]

DIRECTION [
  ( <direction> left </direction> )
  ( <direction> right </direction> )
]

HEADING [
  (<heading> NUMBER NUMBER zero </heading>)
]

CLEARED [
  (cleared ?on ils approach ?runway RUNWAY)
]

RUNWAY [
  (<runway> NUMBER NUMBER [left right] </runway>)
]

CONTACT [
  (contact tower one ninety nine)
]
```

D. Lattice search details

The phone lattice WFSTs $\mathcal{T}^{\text{phone}}$ used for scoring are non-deterministic and may contain multiple edges which output a non-word label symbol, e.g. ϵ or concept symbols as described in section C.2.1 on page 85. Furthermore, a phone lattice and its corresponding WFST may contain cycles. Due to these properties, a traditional implementation of Viterbi search [5, pp. 629–631, 69] cannot be used for calculating the score of a word label sequence $y^{\text{word}} \equiv \langle \gamma_1^{\text{word}} \dots \gamma_n^{\text{word}} \rangle$. Therefore, a search algorithm was implemented which finds the score of the best-scoring corresponding output symbol sequence $y \equiv \langle \gamma_1 \dots \gamma_m \rangle$, which may contain such non-word symbols and so the length of the word label sequence n may not equal that of the output symbol sequence m (see section 5.5.3 on page 35 for details).

In order to find the best-scoring sequence of transitions $\delta(q_{i-1}, \sigma_i, \gamma_i, q_i)$ in a phone lattice WFST $\mathcal{T}$ which outputs a symbol sequence y corresponding to the given word label sequence y^{word}, a breadth-first search of the phone lattice is performed, using each output symbol as an intermediate goal in the complete path which is to be found and which represents the entire word label sequence. This is done by iteratively expanding all edges $\langle s_d, \sigma_d, \gamma_d, w_d, e_d \rangle \in \delta$ for which the start state $s_d = q_{i-1}$ and the output symbol $\gamma_d = \gamma_i$ for each label in the sequence $\gamma_1 \dots \gamma_n$. The score of the best-scoring path to each unique state for each iteration (i.e. for each output label and thus for each path segment) is stored in an associative array (see algorithm D.1 on the following page).

D.1. Goal edge search

In each iteration, the goal edge updating function
$update\_goal\_edges(\gamma_y, state, score, scores', visited\_states, \delta)$ in
algorithm D.2 on page 97 updates the score *score* of the best-scoring path to a given state *state* by adding *score* to the transition weight w for each transition with the given end state as a start state, i.e. for all $\langle s, \sigma, \gamma_d, w, e \rangle \in \delta$ where $\gamma_d = \gamma_y$ and $s = state$: For each of these **goal edges**, an updated path score $score' \equiv score + w$ is calculated and put into the associative array *scores'* ($scores'[e] \leftarrow score'$) if the transition end state e is not already in the array; In the case that the array already contains a reference to the end state e, the path score for the given state is re-assigned only if it is less than the one currently in the array (see algorithm D.3 on page 97). Likewise, the best path to the given state is also added an associative array *visited_states* which represents the best path to *all* states visited during the given search iteration — not just the goal states: See section D.2.1 on page 97 for details on how this information is used by an edge expansion search heuristic to avoid exploring a transition path more than once.

Algorithm D.1 Lattice search

function UPDATE($\gamma_y, scores, scores', visited\_states, \delta$) ▷ $scores[Q] \mapsto \mathbb{R}$, $visited\_states[Q] \mapsto \mathbb{R}$, $d \in \delta \equiv \langle s, \sigma, \gamma, w, e \rangle$
  for $state \in scores$ **do**
    $score \leftarrow scores[state]$ ▷ The score of the best-scoring path to *state*
    $update\_goal\_edges(\gamma_y, state, score, scores', visited\_states, \delta)$ ▷ See algorithm D.2 on the next page
    $update\_pre\_goal\_edges(\gamma_y, state, score, scores, scores', visited\_states, \delta)$ ▷ See algorithm D.4 on page 98
  end for
  return $scores'$
end function
procedure SCORE(y, δ) ▷ $y \equiv \langle \gamma_1 \dots \gamma_n \rangle$, $d \in \delta \equiv \langle s, \sigma, \gamma, w, e \rangle$
  for $\langle s, \sigma, \gamma_d, w, e \rangle \in \delta$ **where** $s = q_0 \wedge \gamma_d = \gamma_1$ **do**
    $scores[e] \leftarrow w$ ▷ Put the transition end state and weight into an associative array of initial transition scores
  end for
  if $|y| > 1$ **then**
    for $\gamma_2 \dots \gamma_n \in y$ **do**
      $scores' \leftarrow \{\}$ ▷ Construct an associative array of updated path scores to goal states
      $visited\_states \leftarrow \{\}$ ▷ Construct an associative array of path scores to all visited states
      $scores \leftarrow update(\gamma_i, scores, scores', visited\_states, \delta)$
    end for
  end if
  return $\min_{state} scores[state]$ ▷ Return the minimum final path score
end procedure

D.2. Pre-goal edge search

Although ϵ-transitions are removed in projecting the ASR phone lattice to the word lattice, there may be multiple transitions for a given state in the phone lattice which do not output explicit word symbols (including ϵ) (see section 5.5.3 on page 35). These include transitions which have non-word labels as output symbols which represent concepts, which are used for machine reading during re-scoring (see section C.2.1 on page 85). Therefore, label sequences including these non-word symbols are considered equivalent to those without during scoring (see formula 5.5.3 on page 36 and section C.2.1 on page 85).

In order to find the score of the best-scoring path $f_\Phi(x, y)$ through the phone lattice WFST $\mathcal{T}^{\text{phone}}$ for the word label sequence $y^{\text{word}} \equiv \langle \gamma_1^{\text{word}} \dots \gamma_n^{\text{word}} \rangle$, these non-word transitions must be traversed for each word label γ_i^{word} in order to find the best-scoring path to a state which outputs the next word label $\gamma_{i+1}^{\text{word}}$.

For each of these **pre-goal edges**, the function

Algorithm D.2 Lattice search: Updating goal edges

function UPDATE_GOAL_EDGES($\gamma_y, state, score, scores', visited\_states, \delta$) ▷
$state \in Q$, $score \in \mathbb{R}$, $scores[Q] \mapsto \mathbb{R}$, $visited\_states[Q] \mapsto \mathbb{R}$, $d \in \delta \equiv \langle s, \sigma, \gamma, w, e \rangle$
  for $\langle s, \sigma, \gamma_d, w, e \rangle \in \delta$ **where** $\gamma_d = \gamma_y \wedge s = state$ **do**
    $score' \leftarrow score + w$
    $put(state, score', scores')$ ▷ See algorithm D.3
    $put(state, score', visited\_states)$
  end for
end function

Algorithm D.3 Lattice search: Putting path scores into an associative array

function PUT($state, score, scores$) ▷ $state \in Q$, $score \in \mathbb{R}$, $scores[Q] \mapsto \mathbb{R}$
  if $state \in scores$ **then**
    $current\_score \leftarrow scores[state]$ ▷ The score of the best-scoring path to $state$
    if $score < current\_score$ **then**
      $scores[state] \leftarrow score$
    end if
  else
    $scores[state] \leftarrow score$
  end if
end function

$update\_pre\_goal\_edges(\gamma_i, state, score, scores, scores', visited\_states, \delta)$ in algorithm D.4 on the next page performs a depth-first search for the goal edge by expanding the pre-goal edge, updating the path score $score' \equiv scores[state] + w_d$ just as done with goal edges. However, unlike the goal edge updating function $update\_goal\_edges(\cdot)$ in section D.1 on page 95, instead of putting this information directly in $scores'$, the goal and pre-goal edges for the next state e are recursively expanded, passing the updated path score $score'$ to the recursively-called function
$update(\gamma_i, e_d, score', scores, scores', visited\_states, \delta)$ in algorithm D.1 on the previous page. In this fashion, the algorithm expands an arbitrary number of intermediate pre-goal edges while performing a depth-first search for a transition or transitions outputting the goal label γ_i and puts the goal path results in the path score map $scores'_i$ for the given output label updating iteration for $\gamma_i \in y$.

D.2.1. Pre-goal edge expansion heuristic

Due to the fact that there may be many ϵ-transitions which must be traversed as pre-goal edges in order to find the next goal edge outputting the word label $\gamma_{i+1}^{\text{word}}$, pre-goal edge search can be computationally expensive. However, the linear combination of transition weights $\sum_{i=1}^{n} \delta(q_{i-1}, \sigma_i, \gamma_i, q_i)$ is a strictly increasing monotonic function: With the exception of cases involving weight pushing [74, pp. 249–251], a transition weight is always greater than zero, and so, for every edge traversed, the score increases:

Algorithm D.4 Lattice search: Updating pre-goal edges

$\Gamma^{\overline{\text{word}}} = \Gamma \setminus \Gamma^{\text{word}}$ ▷ The set of non-word labels (e.g. ϵ; see section 5.2 on page 32)
function UPDATE_PRE_GOAL_EDGES($\gamma_y, state, score, scores, scores', visited\_states, \delta$)
▷ $state \in Q, score \in \mathbb{R}$, $scores[Q] \mapsto \mathbb{R}$, $visited\_states[Q] \mapsto \mathbb{R}$, $d \in \delta \equiv \langle s, \sigma, \gamma, w, e \rangle$
  for $\langle s, \sigma, \gamma_d, w, e \rangle \in \delta$ **where** $\gamma_d \in \Gamma^{\overline{\text{word}}} \wedge s = state$ **do**
    $score' \leftarrow score + w$
    if $should\_be\_expanded(state, score', visited\_states) =$ **true then** ▷ See algorithm D.5 on the next page
      $update\_goal\_edges(\gamma_y, scores, scores', visited\_states, \delta)$ ▷ See algorithm D.2 on the preceding page
      $update\_pre\_goal\_edges(\gamma_y, scores, scores', visited\_states, \delta)$
    end if
  end for
end function

$\delta(q_{i-1}, \sigma_i, \gamma_i, q_i) < \delta(q_{i-1}, \sigma_i, \gamma_i, q_i) + \delta(q_i, \sigma_{i+1}, \gamma_{i+1}, q_{i+1})$.

Using this knowledge, a simple search heuristic was implemented which prunes away pre-goal edges for which the sum of the current path score $s \equiv \sum_{i=1}^{n} \delta(q_{i-1}, \sigma_i, \gamma_i, q_i)$ and the transition weight $\delta(q_n, \sigma_{n+1}, \gamma_{n+1}, q_{n+1})$ is less than the score of the current best path to the next state q_{n+1}:

$$should\_be\_expanded(p, V) \equiv \begin{cases} 1 & \text{if } \{v \in V : same(v, p) \wedge [\![score(v) \leq score(p)]\!]\} = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

$$score(p) \equiv \sum_{\langle q_i, \gamma_i \rangle \in p}^{|p|} \delta(q_{i-1}, \sigma_i, \gamma_i, q_i) \tag{D.2.1}$$

$$same(p_a, p_b) \equiv [\![last(p_a) = last(p_b)]\!]$$

$$last(p) \equiv q \in e_n \in p$$

$$\text{where } p \equiv \langle e_1 \dots e_n \rangle, e \equiv \langle q, \gamma \rangle$$

Given the current best path *score* to a given *state*, an outgoing pre-goal edge from *state* is expanded during search only if the heuristic function *should_be_expanded*(*state*, *score*, *visited_states*) returns **true** (see algorithm D.5 on the next page) [cf. 55, pp. 105 sqq.].

This heuristic is rather conservative in the edges it prunes away: In reality, there may be pre-goal edges which have an updated path score $s' \equiv s + \delta(q_n, \sigma_{n+1}, \gamma_{n+1}, q_{n+1})$ which is less than the current best path score but, nevertheless, subsequently updated path scores $s'' \equiv s' + \delta(q_{n+1}, \sigma_{n+2}, \gamma_{n+2})$ are not less than the best path score: It would be possible to e.g. prune away all pre-goal edges for which the updated path score s' plus a constant l which represents the least weight of all pre-goal edges (e.g. 30.4643 in the case of the grammars used in this experiment) is equal to or greater than the current

Algorithm D.5 Lattice search: Pre-goal edge expansion heuristic

function SHOULD_BE_EXPANDED($state, score, visited\_states$) ▷
$state \in Q, score \in \mathbb{R}$, $visited\_states[Q] \mapsto \mathbb{R}$
  if $state \in visited\_states$ **then**
    $current\_score \leftarrow visited\_states[state]$ ▷ The score of the best-scoring path to *state*
    return $[\![score < current\_score]\!]$
  else
    return true
  end if
end function

best path score, given that the subsequent path s'' with the lowest score will be at least $s'' \geq s' + l$. However, this entails that l must be tuned specifically for each dataset: If the value is too low, the heuristic would not be effective, but if it is too high, the heuristic would no longer be admissible and so the optimal solution (or any solution at all) may not be found [cf. 55, pp. 107–109].

Therefore, by pruning the edges which result in a new path score equal to or greater than the current best path to the goal label $\gamma_{i+1}^{\text{word}}$ before the edges are expanded, the number of active paths which must be searched is minimised while retaining admissibility.

E. Re-scoring methods

In order to calculate the knowledge score $f_C(x, y)$ for a hypothesis y, a vector of semantic attributes is parsed from the hypothesis using the concept labels in the vocabulary (see section C.2.1 on page 85); This frame-based semantic representation [cf. 71] is evaluated against the predicates encoded by the constraint set C.

E.1. Concept parsing

In order to parse concepts from the label sequence $y \equiv \langle \gamma_1 \dots \gamma_n \rangle$, a version of a simple LR parser [9, pp. 33–55] was implemented which parses the semantic content of individual concepts denoted by the label sequence (e.g. CALLSIGN or ALTITUDE) and returns a vector of semantic attributes $parse\colon \mathcal{Y} \mapsto \mathcal{Z}$, where $z \in \mathcal{Z} \equiv \begin{bmatrix} z_1 & \cdots & z_n \end{bmatrix}$.

For the CFGs described in section 5.3 on page 32 and appendix C on page 84, a non-terminal grammar category N which denotes a concept α features the corresponding concept start tag $start\_tag(\alpha)$ as the first terminal category $\gamma_1 = start\_tag(\alpha)$ in the right-hand side of the corresponding production rule $N \to \gamma_1 \dots \gamma_n$ and the corresponding end tag $end\_tag(\alpha)$ as the last terminal category $\gamma_n = end\_tag(\alpha)$ in the right-hand side (see section 5.2 on page 32). During parsing, a concept tree represented by nested concept tags and the symbols they contain is created by maintaining a stack of concepts $\alpha_1 \dots \alpha_n$, shifting a new concept α_{n+1} onto the stack when the corresponding concept start tag $start\_tag(\alpha_{n+1})$ is parsed, and adding a terminal category (i.e. output symbol) to the concept on the top of the stack when the label is parsed. When a concept end tag $end\_tag(\alpha_n)$ is parsed, the stack is reduced, popping the last concept α_n and adding it to the ordered set of children of the previous concept $children(\alpha_{n-1}) \leftarrow children(\alpha_{n-1}) \cup \alpha_n$. Therefore, each concept α has a parent-child relationship with others in the concept tree, maintaining references to any children it has.

Finally, when a concept end tag $end\_tag(\alpha_n)$ is parsed, the contents of that concept are parsed by a formatter which converts the content labels $\gamma_i \dots \gamma_{n-1}$ (where $\gamma_j = start\_tag(\alpha_n)$ and initially $i = j + 1$) into an internal representation, e.g. parsing the integer value 70 given the labels `<flightlevel> seven zero </flightlevel>` denoting a flight level or parsing the alphanumeric string `ADR23` given `<callsign> <airline> adria </airline> <flightnumber> two three </flightnumber> </callsign>`.
These parsed values are then converted into numeric attribute values and put into an attribute vector $z \equiv \begin{bmatrix} z_1 & \cdots & z_n \end{bmatrix}$, where n is the count of all concept types ($n = |A|$), e.g. α_{fl} for FLIGHTLEVEL concepts and α_{callsign} for CALLSIGN concepts. This attribute vector is used for semantic interpretation in natural language understanding using frame-and-slot semantics.

E.2. Situation modelling

In order to create a situation model representing the ATC simulation at the time the utterance was made, the difference is calculated between the time of the start of the recording session and the timestamp of a given utterance. This time offset τ is then used to obtain the simulation data from the 4D-CARMA server log file (see section 4.2.1 on page 27) which represents the flight information for all aircraft in the simulation airspace at the time of the utterance within an envelope of $\tau - 5 \ldots \tau$ seconds. This situation model is used to evaluate constraints given an utterance to be re-scored.

E.3. Semantic interpretation

After parsing an attribute vector representing the concepts denoted by the words in a given label sequence, this attribute vector is used in semantic interpretation [cf. 19, 65] to create a template representing the semantic information denoted by the utterance [cf. 71]: For example, see figure E.1.

Figure E.1.: A semantic template representing an ATC utterance (specifically, *Adria two three turn left heading two five zero*).

$$\left[\begin{array}{ll} \textsc{Callsign} & \texttt{ADR23} \\ \textsc{Command} & \left[\begin{array}{ll} \textsc{Direction} & \texttt{left} \\ \textsc{Heading} & 250 \end{array}\right]_{turn} \end{array}\right]_{s}$$

This semantic template is used in addition to the situation model representing the ATC simulation state at the time of the utterance in order to evaluate constraints in the context in which the utterance the template represents was made.

F. Baseline results

Table F.1.: Baseline WER and SER results by speaker for classification using the restricted grammar.

Group	WER	SD	SER
ben	0.00	0.00	0.00
bor	0.00	0.00	0.00
fri	0.64	0.64	6.12
ice	0.00	0.00	0.00
kat	0.23	0.28	2.17
kla	0.26	0.25	2.50
mir	0.00	0.00	0.00
mja	1.98	1.95	18.87
mna	0.86	0.90	8.20
nik	0.00	0.00	0.00
phi	0.16	0.16	1.54
ric	1.34	1.19	12.79
sen	0.00	0.00	0.00
sup	0.00	0.00	0.00
tsh	0.00	0.00	0.00
xyl	0.00	0.00	0.00
MEAN	0.30	0.33	2.89

Table F.2.: Baseline MRR results by speaker for classification using the restricted grammar.

Group	MRR	SD
ben	1.000000	0.000000
bor	1.000000	0.000000
fri	0.969388	0.014369
ice	1.000000	0.000000
kat	0.989130	0.005317
kla	0.987500	0.006094
mir	1.000000	0.000000
mja	0.896226	0.050552
mna	0.954918	0.023582
nik	1.000000	0.000000
phi	0.992308	0.003787
ric	0.906977	0.066928
sen	1.000000	0.000000
sup	1.000000	0.000000
tsh	1.000000	0.000000
xyl	1.000000	0.000000
MEAN	0.982611	0.011272

Table F.3.: Baseline WER and SER by speaker for classification using the unrestricted grammar.

Group	WER	SD	SER
ben	3.87	2.61	37.14
bor	1.78	2.42	14.08
fri	2.12	1.85	20.41
ice	1.11	1.16	10.64
kat	5.69	19.66	34.78
kla	2.34	3.33	17.50
mir	3.41	2.43	32.56
mja	6.75	11.41	41.51
mna	6.54	5.06	52.46
nik	1.02	1.07	9.72
phi	2.90	2.75	26.15
ric	6.21	6.46	44.19
sen	0.69	0.75	6.56
sup	2.26	2.18	20.45
tsh	1.09	1.55	9.43
xyl	0.89	0.86	8.51
MEAN	2.81	3.98	22.58

Table F.4.: Baseline MRR by speaker for classification using the unrestricted grammar.

Group	MRR	SD
ben	0.766463	0.101094
bor	0.897746	0.071116
fri	0.873299	0.066182
ice	0.936170	0.035967
kat	0.768034	0.111364
kla	0.873719	0.082068
mir	0.812081	0.079058
mja	0.695232	0.148300
mna	0.664821	0.118419
nik	0.942130	0.034074
phi	0.845876	0.074785
ric	0.605491	0.207323
sen	0.959563	0.024339
sup	0.881117	0.059157
tsh	0.947889	0.027276
xyl	0.948582	0.028976
MEAN	0.848671	0.087077

G. Re-scoring results

Table G.1.: WER and SER results while re-scoring using constraints for the restricted grammar.

Test	WER	SD	baseline-WER	SER	baseline-SER
baseline	0.30	0.33	0.0	2.89	0.0
possibleCS	0.30	0.33	0.0	2.89	0.0
radarCS	0.30	0.33	0.0	2.89	0.0
goldStdCS	0.30	0.33	0.0	2.89	0.0
spd	0.30	0.33	0.0	2.89	0.0
alt	0.30	0.33	0.0	2.89	0.0
valHdg	0.29	0.32	0.01	2.80	0.09
alt,spd,valHdg	0.29	0.32	0.01	2.80	0.09
alt,radarCS,spd,valHdg	0.29	0.32	0.01	2.80	0.09
alt,goldStdCS,spd,valHdg	0.29	0.32	0.01	2.80	0.09
rwy	0.24	0.26	0.06	2.26	0.63
alt,spd,rwy	0.24	0.26	0.06	2.26	0.63
alt,radarCS,spd,rwy	0.24	0.26	0.06	2.26	0.63
alt,goldStdCS,spd,rwy	0.24	0.26	0.06	2.26	0.63

Table G.2.: MRR results while re-scoring using constraints for the restricted grammar.

Test	MRR	SD	MRR-baseline
baseline	0.982611	0.011272	0.000000
possibleCS	0.982611	0.011272	0.000000
radarCS	0.982611	0.011272	0.000000
goldStdCS	0.982611	0.011272	0.000000
spd	0.982611	0.011272	0.000000
alt	0.982686	0.011168	0.000075
valHdg	0.983062	0.011061	0.000452
alt,spd,valHdg	0.983138	0.010957	0.000527
alt,radarCS,spd,valHdg	0.983138	0.010957	0.000527
alt,goldStdCS,spd,valHdg	0.983138	0.010957	0.000527
rwy	0.985772	0.009791	0.003162
alt,spd,rwy	0.985848	0.009686	0.003237
alt,radarCS,spd,rwy	0.985848	0.009686	0.003237
alt,goldStdCS,spd,rwy	0.985848	0.009686	0.003237

Table G.3.: WER and SER results while re-scoring using constraints for the unrestricted grammar.

Test	WER	SD	baseline-WER	SER	baseline-SER
baseline	2.81	3.98	0.0	22.58	0.0
spd	2.81	3.98	0.0	22.58	0.0
alt	2.81	3.98	0.0	22.58	0.0
valHdg	2.79	3.82	0.02	22.49	0.09
alt,spd,valHdg	2.79	3.82	0.02	22.49	0.09
rwy	2.74	3.91	0.07	22.13	0.45
alt,spd,rwy	2.74	3.91	0.07	22.13	0.45
possibleCS	0.55	0.78	2.26	4.61	17.97
radarCS	0.55	0.78	2.26	4.61	17.97
alt,radarCS,spd,valHdg	0.52	0.74	2.29	4.43	18.15
goldStdCS	0.50	0.61	2.31	4.52	18.06
alt,goldStdCS,spd,valHdg	0.50	0.66	2.31	4.43	18.15
alt,radarCS,spd,rwy	0.45	0.62	2.36	3.88	18.7
alt,goldStdCS,spd,rwy	0.44	0.54	2.37	3.88	18.7

Table G.4.: MRR results while re-scoring using constraints for the unrestricted grammar.

Test	MRR	SD	MRR-baseline
baseline	0.848671	0.087077	0.000000
spd	0.848671	0.087077	0.000000
alt	0.848675	0.087071	0.000004
valHdg	0.849418	0.086676	0.000748
rwy	0.852010	0.085420	0.003339
alt,spd,valHdg	0.849422	0.086671	0.000751
alt,spd,rwy	0.852013	0.085414	0.003343
possibleCS	0.965786	0.026960	0.117115
radarCS	0.965854	0.026851	0.117183
goldStdCS	0.966576	0.026322	0.117906
alt,radarCS,spd,valHdg	0.967179	0.025923	0.118508
alt,goldStdCS,spd,valHdg	0.967179	0.025923	0.118508
alt,radarCS,spd,rwy	0.969889	0.024738	0.121218
alt,goldStdCS,spd,rwy	0.969889	0.024738	0.121218

Bibliography

[1] *Aeronautical Information Manual. Official Guide to Basic Flight Information and ATC Procedures.* U.S. Department of Transportation Federal Aviation Administration. Washington, D.C., USA, 11th Feb. 2010. URL: http://www.faa.gov/air_traffic/publications/ATPubs/AIM/aim.pdf (visited on 23/11/2011).

[2] *All Clear Phraseology Manual.* Eurocontrol. Brussels, Belgium, 4th Apr. 2011. 20 pp. URL: http://www.skybrary.aero/bookshelf/books/115.pdf (visited on 03/11/2011).

[3] Jørgen Bang-Jensen and Gregory Gutin. *Digraphs: Theory, Algorithms and Applications.* 1st edition (online). Springer Monographs in Mathematics. London, England, UK: Springer, Aug. 2007. 754 pp. ISBN: 1-85233-268-9. URL: http://www.cs.rhul.ac.uk/books/dbook/ (visited on 21/11/2011).

[4] Sarah Belia et al. 'Researchers misunderstand confidence intervals and standard error bars'. In: *Psychological Methods* 10.4 (Dec. 2005), pp. 389–396. DOI: 10.1037/1082-989X.10.4.389.

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Information Science and Statistics. New York, NY, USA: Springer, 2006. 304 pp. ISBN: 978-0-387-31073-2.

[6] Ming-Wei Chang, Lev Ratinov and Dan Roth. 'Guiding semi-supervision with constraint-driven learning'. In: *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics.* (Prague, Czech Republic, 23rd–30th June 2007). Ed. by Annie Zaenen and Antal van den Bosch. Stroudsberg, PA, USA: Association for Computational Linguistics, June 2007, pp. 280–287. URL: http://aclweb.org/anthology-new/P/P07/P07-1036.pdf (visited on 02/11/2011).

[7] Ming-Wei Chang, Lev Ratinov and Dan Roth. 'Constraints as prior knowledge'. In: *ICML Workshop on Prior Knowledge for Text and Language Processing.* (Hesinki, Finland, 5th–9th July 2008). The International Machine Learning Society. 2008.

[8] Ming-Wei Chang, Nicholas Rizzolo and Dan Roth. 'Integer Linear Programming in NLP — Constrained Conditional Models'. In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics.* (Los Angeles, CA, USA, 2nd–4th June 2010). Ed. by Ron Kaplan et al. Stroudsberg, PA, USA: Association for Computational Linguistics, June 2010. Chap. Tutorial Abstracts, pp. 9–14. URL: http://aclweb.org/anthology-new/N/N10/N10-4005.pdf (visited on 22/10/2011).

[9] Nigel P. Chapman. *LR Parsing: Theory and Practice.* Cambridge, England, UK: Cambridge University Press, 17th Dec. 1987. 240 pp. ISBN: 978-0-521-30413-9.

[10] Stanley Chen, Douglas Beeferman and Ronald Rosenfeld. 'Evaluation Metrics for Language Models'. In: *Proceedings of the 1998 DARPA Broadcast News Transcription and Understanding Workshop*. (Lansdowne Conference Resort, Lansdowne, VA, USA, 8th–11th Feb. 1998). Defense Advanced Research Projects Agency. Gaithersburg, MD, USA: NIST Information Access Division, Feb. 1998. Chap. Language Modeling. URL: http://www.itl.nist.gov/iad/mig//publications/proceedings/darpa98/ (visited on 17/11/2011).

[11] R. Córdoba et al. 'Air traffic control speech recognition system cross-task & speaker adaptation'. In: *IEEE Aerospace and Electronic Systems Magazine* 21.9 (Sept. 2006), pp. 12–17. DOI: 10.1109/MAES.2006.1705165.

[12] K. H. Davis, R. Biddulph and S. Balashek. 'Automatic Recognition of Spoken Digits'. In: *Journal of the Acoustical Society of America* 24.6 (Nov. 1952), pp. 637–642. DOI: 10.1121/1.1906946.

[13] Rina Dechter. 'Bucket elimination: a unifying framework for reasoning'. In: *Artificial Intelligence* 113 (1–2 Sept. 1999), pp. 41–85. DOI: 10.1016/S0004-3702(99)00059-4.

[14] Rina Dechter and Avi Dechter. 'Belief Maintenance in Dynamic Constraint Networks'. In: *Proceedings of the 7th National Conference on Artificial Intelligence*. (St Paul, MN, USA, 21st–26th Aug. 1988). Association for the Advancement of Artificial Intelligence. Menlo Park, CA, USA: AAAI Press, Aug. 1988, pp. 37–42. ISBN: 0-262-51055-3.

[15] Sir Arthur Conan Doyle. 'The Adventure of the Beryl Coronet'. In: *The Adventures of Sherlock Holmes*. Bath, England, UK: George Newnes, 1892.

[16] *Dragon User Guide, Version 11*. Nuance Communications, Inc. 1005 Hamilton Ave., Menlo Park, CA, 94025, USA, 2010. 121 pp. URL: http://www.nuance.com/ucmprod/groups/dragon/documents/webasset/nd_004969.pdf (visited on 02/11/2011).

[17] Denys Duchier. 'Constraint Programming for Natural Language Processing'. In: *Proceedings of the 12th European Summer School in Logic, Language and Information*. (Birmingham, England, UK, 6th–18th Aug. 2000). Ed. by Achim Jung and Eike Ritter. European Association for Logic, Language and Information. Villers les Nancy Cedex, France, Aug. 2000. URL: http://www.ps.uni-sb.de/~duchier/esslli-2000/ (visited on 20/11/2011).

[18] Christopher Dyer, Smaranda Muresan and Philip Resnik. 'Generalizing Word Lattice Translation'. In: *46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-08: HLT). Proceedings of the Conference*. (Columbus, OH, USA, 15th–20th June 2008). Ed. by Johanna D. Moore et al. Stroudsberg, PA, USA: Association for Computational Linguistics, June 2008, pp. 1012–1020. URL: http://aclweb.org/anthology-new/P/P08/P08-1115.pdf (visited on 22/10/2011).

[19] Wieland Eckert and Heinrich Niemann. 'Semantic analysis in a robust spoken dialog system'. In: *Third International Conference on Spoken Language Processing (ICSLP 94)*. (Yokohama, Japan, 18th–22nd Sept. 1994). International Speech Communication Association. ISCA Archive, 1994, pp. 107–110. URL: http://www.isca-speech.org/archive/icslp_1994/i94_0107.html (visited on 21/10/2011).

[20] Lee D. Erman et al. 'The HEARSAY-II speech understanding system: Integrating knowledge to resolve uncertainty'. In: *ACM Computing Surveys* 12 (June 1980), pp. 213–253. DOI: 10.1145/356810.356816.

[21] Pamela K. Fink and Alan W. Biermann. 'The correction of ill-formed input using history-based expectation with applications to speech understanding'. In: *Computational Linguistics* 12.1 (Jan. 1986), pp. 13–36.

[22] Jenny Rose Finkel and Christopher D. Manning. 'Enforcing transitivity in coreference resolution'. In: *46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-08: HLT). Proceedings of the Conference*. (Columbus, OH, USA, 15th–20th June 2008). Ed. by Johanna D. Moore et al. Stroudsberg, PA, USA: Association for Computational Linguistics, June 2008. Chap. Short Papers, pp. 45–48. URL: http://aclweb.org/anthology-new/P/P08/P08-2012.pdf (visited on 22/10/2011).

[23] Jonathan Fiscus et al. *1997 English broadcast news speech (HUB4)*. Philadelphia, PA, USA: Linguistic Data Consortium, 1998. URL: http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC98S71 (visited on 02/11/2011).

[24] Eugene C. Freuder and Alan K. Mackworth. 'Constraint Satisfaction: An Emerging Paradigm'. In: *Handbook of Constraint Programming*. Ed. by Francesca Rossi, Peter van Beek and Toby Walsh. Foundations of Artificial Intelligence. New York, NY, USA: Elsevier, Aug. 2006. Chap. 2, pp. 13–27. ISBN: 978-0-444-52726-4.

[25] Christian Fügen et al. 'Open Domain Speech Translation: From Seminars and Speeches to Lectures'. In: *Proceedings of the TC-Star Workshop on Speech-to-Speech Translation*. (Barcelona, Spain, 2006). Paris, France: Evaluations and Language Resources Distribution Agency, 2006.

[26] Sadaoki Furui. 'Toward Robust Speech Recognition and Understanding'. In: *Text, Speech and Dialogue*. Ed. by Václav Matoušek and Pavel Mautner. Vol. 2807. Lecture Notes in Computer Science. Springer, 2003, pp. 2–11. DOI: 10.1007/978-3-540-39398-6_2.

[27] Hartmut Helmke. 'Time-based arrival management: New controller support tools are necessary to ensure the best-possible use of available airspace'. In: *Air Traffic Technology International* (2011), pp. 40–43.

[28] Hartmut Helmke et al. 'Time-Based Arrival Management for Dual Threshold Operation and Continous Descent Approaches'. In: *Proceedings of the Eighth USA/Europe Air Traffic Management Research and Development Seminar (ATM2009)*. (Napa, CA, USA, 29th June–2nd July 2009). 2009. URL: http://www.atmseminarus.org/seminarContent/seminar8/papers/p_014_CDA.pdf (visited on 28/11/2011).

[29] John N. Hooker. 'Operations Research Methods in Constraint Programming'. In: *Handbook of Constraint Programming*. Ed. by Francesca Rossi, Peter van Beek and Toby Walsh. Foundations of Artificial Intelligence. New York, NY, USA: Elsevier, Aug. 2006. Chap. 15, pp. 525–586. ISBN: 978-0-444-52726-4.

[30] Xuedong Huang, Yasuo Ariki and Mervyn Jack. *Hidden Markov Models for Speech Recognition*. Edinburgh Information Technology 7. Edinburgh, Scotland, UK: Edinburgh University Press, 1991. ISBN: 0-7486-0162-7.

[31] *Interspeech'2005–EUROSPEECH, 9th European Conference on Speech Communication and Technology*. (Lisbon, Portugal, 4th–8th Sept. 2005). International Speech Communication Association. ISCA Archive, 2005. 3388 pp. URL: `http://www.isca-speech.org/archive/interspeech_2005/` (visited on 21/10/2011).

[32] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. 2nd edition. Englewood Cliffs, NJ, USA: Prentice-Hall, 9th Feb. 2008. 1024 pp. ISBN: 978-0-13-504196-3.

[33] Judith L. Klavans and Philip Resnik, eds. *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*. (New Mexico State University, Las Cruces, NM, USA, 1st July 1994). Stroudsberg, PA, USA: Association for Computational Linguistics, July 1994. 252 pp. URL: `http://aclweb.org/anthology-new/W/W94/` (visited on 19/11/2011).

[34] Kazunori Komatani et al. 'Contextual constraints based on dialogue models in database search task for spoken dialogue systems'. In: *Interspeech'2005–EUROSPEECH, 9th European Conference on Speech Communication and Technology*. (Lisbon, Portugal, 4th–8th Sept. 2005). International Speech Communication Association. ISCA Archive, 2005, pp. 877–880. URL: `http://www.isca-speech.org/archive/interspeech_2005/i05_0877.html` (visited on 21/10/2011).

[35] Geert-Jan M. Kruijff et al. 'Situated Dialogue Processing for Human-Robot Interaction'. In: *Cognitive Systems*. Ed. by Henrik Iskov Christensen et al. Vol. 8. Cognitive Systems Monographs. Heidelberg, Germany: Springer, July 2010. URL: `http://folk.uio.no/plison/pdfs/cl/my-cosy-book.pdf` (visited on 20/05/2011).

[36] Simon Lacoste-Julien et al. 'Word alignment via quadratic assignment'. In: *Proceedings of the Human Language Technology Conference of the NAACL. Main Conference*. (New York, NY, USA, 4th–9th June 2006). Ed. by Robert C. Moore et al. Stroudsberg, PA, USA: Association for Computational Linguistics, June 2006, pp. 112–119. URL: `http://aclweb.org/anthology-new/N/N06/N06-1015.pdf` (visited on 22/10/2011).

[37] Lori F. Lamel et al. 'The translanguage English database (TED)'. In: *Third International Conference on Spoken Language Processing (ICSLP 94)*. (Yokohama, Japan, 18th–22nd Sept. 1994). International Speech Communication Association. ISCA Archive, 1994, pp. 1795–1798. URL: `http://www.isca-speech.org/archive/icslp_1994/i94_1795.html` (visited on 02/11/2011).

[38] Kevin Lenzo. *The CMU Pronouncing Dictionary*. School of Computer Science, Carnegie Mellon University. URL: http://www.speech.cs.cmu.edu/cgi-bin/cmudict/ (visited on 02/11/2011).

[39] Vladimir Iosifovich Levenshtein. 'Binary codes capable of correcting deletions, insertions, and reversals'. In: *Soviet Physics — Doklady* 10.8 (Feb. 1966), pp. 707–710.

[40] Jinyu Li, Yu Tsao and Chin-Hui Lee. 'A study on knowledge source integration for candidate rescoring in automatic speech recognition'. In: *Proceedings of the 2005 IEEE International Conference on Acoustics, Speech, and Signal Processing*. (18th–23rd Mar. 2005). Vol. 1. Piscataway, NJ, USA: IEEE Signal Processing Society, 2005, I–837–I–840. DOI: 10.1109/ICASSP.2005.1415244.

[41] Andrej Ljolje, Fernando Pereira and Michael Riley. 'Efficient general lattice generation and rescoring'. In: *Proceedings, Sixth European Conference on Speech Communication and Technology (EUROSPEECH'99)*. (Budapest, Hungary, 5th–9th Sept. 1999). European Speech Communication Association. ISCA Archive, 1999, pp. 1251–1254. URL: http://www.isca-speech.org/archive/eurospeech_1999/e99_1251.html (visited on 21/10/2011).

[42] Bronisław K. Malinowski. 'The Problem of Meaning in Primitive Languages'. In: *The Meaning of Meaning*. Ed. by Charles K. Ogden and Ian A. Richards. London, England, UK: Routledge, 1923, pp. 146–152.

[43] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. 6th edition. Cambridge, MA, USA: MIT Press, 18th June 1999. 620 pp. ISBN: 0-26-213360-1.

[44] André F. T. Martins, Noah A. Smith and Eric P. Xing. 'Concise integer linear programming formulations for dependency parsing'. In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. (Suntec, Singapore, 2nd–7th Aug. 2009). Ed. by Keh-Yih Su et al. Stroudsberg, PA, USA: Association for Computational Linguistics, Aug. 2009, pp. 342–350. URL: http://aclweb.org/anthology-new/P/P09/P09-1039.pdf (visited on 22/10/2011).

[45] Mehryar Mohri, Fernando Pereira and Michael Riley. 'Weighted finite-state transducers in speech recognition'. In: *Computer Speech and Language* 16.1 (Jan. 2002), pp. 69–88. DOI: 10.1006/csla.2001.0184.

[46] Johanna D. Moore et al., eds. *46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-08: HLT). Proceedings of the Conference*. (Columbus, OH, USA, 15th–20th June 2008). Stroudsberg, PA, USA: Association for Computational Linguistics, June 2008. URL: http://aclweb.org/anthology-new/P/P08/ (visited on 22/10/2011).

[47] Roger K. Moore. 'PRESENCE: A human-inspired architecture for speech-based human-machine interaction'. In: *IEEE Transactions on Computers* 56.9 (Aug. 2007), pp. 1176–1188. DOI: 10.1109/TC.2007.1080.

[48] *Nuance Speech Recognition System 8.5. Grammar Developer's Guide*. Nuance Communications, Inc. 1005 Hamilton Ave., Menlo Park, CA, 94025, USA, Dec. 2003. 248 pp.

[49] M. Ostendorf et al. 'Integration of diverse recognition methodologies through reevaluation of N-best sentence hypotheses'. In: *Speech and Natural Language: Proceedings of a Workshop*. (Pacific Grove, CA, USA, 19th–22nd Feb. 1991). Stroudsburg, PA, USA: Association for Computational Linguistics, Feb. 1991, pp. 83–87. URL: http://aclweb.org/anthology-new/H/H91/H91-1013.pdf (visited on 15/11/2011).

[50] Vasin Punyakanok, Dan Roth and Wen-Tau Yih. 'The importance of syntactic parsing and inference in semantic role labeling'. In: *Computational Linguistics* 34.2 (June 2008), pp. 257–287. DOI: 10.1162/coli.2008.34.2.257.

[51] Vasin Punyakanok et al. 'Inference & Learning with Linear Constraints'. In: *Minisymposium on Structural Inference & Learning with Constraints*. (Urbana, IL, USA, 10th–11th June 2004). Cognitive Computation Group, University of Illinois at Urbana-Champaign. 2004. URL: http://cogcomp.cs.illinois.edu/sites/LX/ (visited on 02/11/2011).

[52] *Python Programming Language — Official Website*. Python Software Foundation. URL: http://http://www.python.org/ (visited on 18/11/2011).

[53] Dragomir R. Radev et al. 'Evaluating web-based question answering systems'. In: *Third International Conference on Language Resources (LREC 2002)*. (Las Palmas, Canary Islands, Spain, 29th–31st May 2002). European Language Resources Association. Paris, France, 2002. Chap. D1: Demos Session. URL: http://www.lrec-conf.org/proceedings/lrec2002/sumarios/301.htm (visited on 03/11/2011).

[54] Francesca Rossi, Peter van Beek and Toby Walsh, eds. *Handbook of Constraint Programming*. Foundations of Artificial Intelligence. New York, NY, USA: Elsevier, Aug. 2006. 978 pp. ISBN: 978-0-444-52726-4.

[55] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 2nd edition. Englewood Cliffs, NJ, USA: Prentice-Hall, 20th Dec. 2002. 1081 pp. ISBN: 0-13-790395-2.

[56] David Rybach et al. 'The RWTH Aachen University open source speech recognition system'. In: *Proceedings of Interspeech 2009, 10th Annual Conference of the International Speech Communication Association*. (Brighton, England, UK, 6th–10th Sept. 2009). International Speech Communication Association. ISCA Archive, 2009, pp. 2111–2114. URL: http://www.isca-speech.org/archive/interspeech_2009/i09_2111.html (visited on 22/10/2011).

[57] George Saon, Daniel Povey and Geoffrey Zweig. 'Anatomy of an extremely fast LVCSR decoder'. In: *Interspeech'2005–EUROSPEECH, 9th European Conference on Speech Communication and Technology*. (Lisbon, Portugal, 4th–8th Sept. 2005). International Speech Communication Association. ISCA Archive, 2005, pp. 549–

552. URL: http://www.isca-speech.org/archive/interspeech_2005/i05_0549.html (visited on 21/10/2011).

[58] Dirk Schäfer. 'Context-sensitive speech recognition in the air traffic control simulation'. a.k.a. EEC Note No. 02/2001. Brétigny-sur-Orge, France and Munich, Germany: Eurocontrol Experimental Centre and Universität der Bundeswehr München, Feb. 2001. 193 pp. URL: http://www.eurocontrol.int/eec/public/standard_page/DOC_Report_2001_004.html (visited on 03/11/2011).

[59] Ernst Günter Schukat-Talamazzini. *Automatische Spracherkennung. Statistische Verfahren der Musteranalyse*. German. Künstliche Intelligenz. Braunschweig, Germany and Wiesbaden, Germany: Vieweg, 30th Mar. 1995. 403 pp. ISBN: 3-528-05492-1. URL: http://www.minet.uni-jena.de/fakultaet/schukat/asebuch.html (visited on 22/10/2011).

[60] Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication*. Urbana, IL, USA and Chicago, IL, USA: University of Illinois Press, 1949. 144 pp.

[61] Sabato Marco Siniscalchi, Jinyu Li and Chin-Hui Lee. 'A study on lattice rescoring with knowledge scores for automatic speech recognition'. In: *INTERSPEECH 2006 — ICSLP, Ninth International Conference on Spoken Language Processing*. (Pittsburgh, PA, USA, 17th–21st Sept. 2006). International Speech Communication Association. ISCA Archive, Sept. 2006. URL: http://www.isca-speech.org/archive/archive_papers/interspeech_2006/i06_1319.pdf (visited on 19/11/2011).

[62] Roger Argiles Solsona et al. 'Adaptive language models for spoken dialogue systems'. In: *Proceedings of the 2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*. (13th–17th May 2002). Orlando, FL, USA: IEEE Signal Processing Society, 2002. DOI: 10.1109/ICASSP.2002.5745340.

[63] Andreas Stolcke, Yochai König and Mitchel Weintraub. 'Explicit word error minimization in n-best list rescoring'. In: *EUROSPEECH '97 — 5th European Conference on Speech Communication and Technology*. (Rhodes, Greece, 22nd–25th Sept. 1997). Ed. by G. Kokkinakis, N. Fakotakis and E. Dermatas. European Speech Communication Association. ISCA Archive, 1997, pp. 163–166. URL: http://www.isca-speech.org/archive/eurospeech_1997/e97_0163.html (visited on 18/11/2011).

[64] Helmer Strik, Catia Cucchiarini and Judith M. Kessens. 'Comparing the performance of two CSRs: Hnicow to determine the significance level of the differences'. In: *EUROSPEECH 2001 Scandinavia, 7th European Conference on Speech Communication and Technology, 2nd INTERSPEECH Event*. (Aalborg, Denmark, 3rd–7th Sept. 2001). Ed. by Paul Dalsgaard et al. International Speech Communication Association. ISCA Archive, 2001, pp. 2091–2094. URL: http://www.isca-speech.org/archive/eurospeech_2001/e01_2091.html (visited on 22/10/2011).

[65] Lena Strömbäck and Arne Jönsson. 'Robust interpretation for spoken dialogue systems'. In: *The 5th International Conference on Spoken Language Processing, Incorporating the 7th Australian International Speech Science and Technology Conference*. (Sydney Convention Centre, Sydney, Australia, 30th Nov.–4th Dec. 1998). Ed. by Robert H. Mannell and Jordi Robert-Ribes. Australasian Speech Science and Technology Association, Inc. ISCA Archive, 1998, pp. 491–494. URL: http://www.isca-speech.org/archive/icslp_1998/i98_0478.html (visited on 21/10/2011).

[66] Satoshi Tamura, Koji Iwano and Sadaoki Furui. 'Multi-Modal Speech Recognition Using Optical-Flow Analysis for Lip Images'. In: *The Journal of VLSI Signal Processing* 36 (2 2004), pp. 117–124. DOI: 10.1023/B:VLSI.0000015091.47302.07.

[67] *Third International Conference on Spoken Language Processing (ICSLP 94)*. (Yokohama, Japan, 18th–22nd Sept. 1994). International Speech Communication Association. ISCA Archive, 1994. 2234 pp. URL: http://www.isca-speech.org/archive/icslp_1994/ (visited on 21/10/2011).

[68] Gérard Verfaillie and Thomas Schiex. 'Solution Reuse in Dynamic Constraint Satisfaction Problems'. In: *Proceedings of the 11th National Conference on Artificial Intelligence*. (Seattle, WA, USA, 1st–4th Aug. 1994). Association for the Advancement of Artificial Intelligence. Menlo Park, CA, USA: AAAI Press, Aug. 1994, pp. 307–312. ISBN: 978-0-262-51078-3.

[69] Andrew Viterbi. 'Error bounds for convolutional codes and an asymptotically optimum decoding algorithm'. In: *IEEE Transactions on Information Theory* 13.2 (6th Apr. 1967), pp. 260–269. DOI: 10.1109/TIT.1967.1054010.

[70] Willie Walker et al. *Sphinx-4: A flexible open source framework for speech recognition*. Tech. rep. SMLI TR-2004-139. 16 Network Circle, Menlo Park, CA, 94025, USA: Sun Microsystems Laboratories, Nov. 2004. 15 pp. URL: http://research.sun.com/techrep/2004/smli_tr-2004-139.pdf (visited on 03/11/2011).

[71] Yeyi Wang, Li Deng and Alex Acero. 'Semantic Frame Based Spoken Language Understanding'. In: *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*. Ed. by Gokhan Tur and Renato de Mori. Chichester, England, UK: John Wiley & Sons, May 2011. Chap. 3. ISBN: 978-0-470-68824-3. DOI: 10.1002/9781119992691.

[72] Daniel S. Weld and Johan de Kleer, eds. *Readings in qualitative reasoning about physical systems*. Representation and Reasoning. San Francisco, CA, USA: Morgan Kaufmann, 1990. ISBN: 1-55860-095-7.

[73] Matthias Wölfel and John M. McDonough. *Distant Speech Recognition*. Chichester, England, UK: John Wiley & Sons, 27th Apr. 2009. 573 pp. ISBN: 978-0-470-51704-8. DOI: 10.1002/9780470714089.

[74] Matthias Wölfel and John M. McDonough. 'Search: Finding the Best Word Hypothesis'. In: *Distant Speech Recognition*. Chichester, England, UK: John Wiley & Sons, 27th Apr. 2009. Chap. 7, pp. 231–282. ISBN: 978-0-470-51704-8. DOI: 10.1002/9780470714089.

[75] Wei Xu and Alex Rudnicky. 'Language modeling for dialog system'. In: *Proceedings, Sixth International Conference on Spoken Language Processing (ICSLP 2000)*. (Beijing, China, 16th–20th Oct. 2000). Vol. 1. 4 vols. International Speech Communication Association. 2000, pp. 118–121. URL: http://www.isca-speech.org/archive/icslp_2000/i00_1118.html (visited on 22/10/2011).

[76] Sheryl R. Young. 'The MINDS system: using context and dialog to enhance speech'. In: *Speech and Natural Language: Proceedings of a Workshop Held at Philadelphia, Pennsylvania, February 21–23, 1989*. (Philadelphia, PA, USA, 21st–23rd Feb. 1989). Association for Computational Linguistics. San Francisco, CA, USA: Morgan Kaufmann, June 1989, pp. 131–136. URL: http://aclweb.org/anthology/H/H89/H89-1017.pdf (visited on 02/11/2011).

[77] Sheryl R. Young, Wayne H. Ward and Alexander G. Hauptmann. 'Layering Predictions: Flexible Use of Dialog Expectation in Speech Recognition'. In: *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (ICJAI-89)*. (Detroit, MI, USA, 20th–25th Aug. 1989). Ed. by Natesa Sastri Sridharan. Vol. 2. International Joint Conferences on Artificial Intelligence. San Francisco, CA, USA: Morgan Kaufmann, 1989, pp. 1543–1549. URL: http://ijcai.org/Past%20Proceedings/IJCAI-89-VOL-2/PDF/110.pdf (visited on 22/10/2011).

[78] Sheryl R. Young et al. 'High level knowledge sources in usable speech recognition systems'. In: *Communications of the ACM* 32.2 (Feb. 1989), pp. 183–194. DOI: 10.1145/63342.63344.

[79] Steve J. Young. *The HTK Hidden Markov Model toolkit: Design and philosophy*. Tech. rep. CUED/F-INFENG/TR.152. Cambridge, England, UK: Cambridge University Engineering Department, 6th Sept. 1994.