PERSONAL ASSISTANT FOR HEALTHY LIFESTYLE

DR 4.4: Natural Multimodal Interaction Final Prototype

Bernd Kiefer, Ivana Kruijff-Korbayová, Anna Welker, Rifca Peters[‡], Sarah McLeod[†]

DFKI GmbH, TU Delft[‡], Universität des Saarlandes[†]

⟨{kiefer, ivana.kruijff, anna.welker}@dfki.de, R.M.Peters@tudelft.nl, smcleod@coli.uni-saarland.de⟩

Project, project Id:	EU H2020 PAL / PHC-643783
Project start date:	March 1 2015 (48 months)
Due date of deliverable:	M48
Actual submission date:	March 30, 2019
Lead partner:	DFKI
Revision:	final
Dissemination level:	PU

The present report describes the work carried out in the fourth project year regarding *Natural Multimodal Interaction*. It summarises the Deliverable D4.4: "Natural multimodal interaction final prototype".
Most efforts in year 4 are targeted towards the final integrated system for the experiments over an extended time range. New activities and modules have been added, many already existing ones been extended, most of these additions or extensions also affecting in some way or other the multimodal interaction. The dialogue policies and the linguistic resources have been adapted accordingly.

In addition, some new functionalities were at the core of human-system interaction, namely a new module for first time use of the system, and an integrated Off-Activity-Talk prototype, consisting of robot self disclosure and a *social talk* part.

The Episodic Memory has been extended, and interaction modules for the following activities have been added: Standalone Break & Sort, new Dance activity, the so-called *Tip of the Day* and the Task Suggestions, and for the *Explainable AI* module.

Furthermore, support for an experiment with different interaction styles, using modulated gestures, has been added, and the VOnDA compiler and run-time system has been heavily improved. Version 2.0 of the framework has been released on GitHub. For less experienced users, a graphical editor and compiler for hierarchical state machines has been implemented and is now ready for use.

Executive Summary This document describes the research and implementation work on multimodal interaction in work package 4 (WP4) for the PAL system in the last year. The overall objective of WP4 is to support the goals set for a patient using the PAL system by developing the means to conduct verbal communication, and to analyse textual data and extract relevant information. The components implemented in this work package must support this communication in a way to foster sustainable long-term interactions between a robot (or its avatar) and a human. This requires user-adaptive communication, coupling of verbal and non-verbal communication and grounding communication in long-term memory. WP4 provides the human interaction layer to the numerous sub-activities in the background, and is responsible for the interactive behaviour of the virtual agent.

During Year 4, WP4 developed new interaction functionality and provided language and dialogue support for numerous extensions of the PAL agent that were requested for the upcoming experiments. Almost all additions to the MyPAL app in some way or other required extensions in the WP4, either by extending the natural language resources, or by adding completely new dialogue strategies.

This comprises verbal feedback for new activities like the standalone *break & sort* game, the new *dance* activity, but also for the new solution for *targeted task suggestions*. An *Explainable AI* module has been added, which provides explanations why certain suggestions are given to the user, or what exactly the correct answer to a quiz question will help the user with. In addition, the module also provides tips to active goals, which can be actively retrieved by the user.

Episodic Memory: The episodic memory module (EMM) has been extended to cover more episodes, e.g. about sports activities or information about physical conditions that was entered by the user into the timeline, making it more versatile and than its predecessor.

Off-Activity-Talk: We added Off-Activity-Talk (OAT) into the system in two ways, firstly, integrating the ontology-backed prototype that was developed in year 3, and an improved version of the self-disclosure that was developed in year 2. This way, we tried to get more interaction data from the children by providing proactive behaviour of the robot and expecting some natural language answers in return. Interpretation is done using the robust NLU prototype from year 3, to which we added more training data, and adding another interpretation layer with support for enhanced SRGS (Speech Recognition Grammar Specification)[1] grammars. The OAT module also exploits results from the *Sentiment Analysis* to assess the emotions connected with an utterance.

Behaviour Styles: To study the effect of variation in non-verbal behaviour on system usage, we implemented stylised behaviours that are ap-

[1] `https://www.w3.org/TR/speech-grammar/`

plied systematically during the different activities. For this purpose, the activities are linked with a teaching style, and the corresponding style, which may vary from child to child, is then applied to the robot's movements.

Dialogue Framework: The VOnDA framework was heavily improved, adding new features in the language as well as the functionality part, fixing many problems, and making compilation between 10 and 20 times faster than before. The visual debugger has also been improved, and a graphical editor for hierarchical state machines, together with a compiler into VOnDA code, has been implemented. We have released the version 2.0 of VOnDA on GitHub on October 1st, 2018.

Automatic Speech Recognition: We have also integrated an ASR (automatic speech recognition) module that can use either the Nuance Cloud infrastructure, or a local Kaldi server for recognition. This integration will be shown at the final review, small experiments with child speech (for English) are under way.

The results of Year 4 are also presented in an accepted conference paper, and a master thesis.

The role of *Multimodal Natural Interaction* in PAL

WP4 focuses on the multimodal interaction around mHealth-Apps and additional conversational functionality in support of the high-level targets set in WP2 and actions selected in WP3. The challenge is to produce natural, flexible, personalised interactions that are sustainable in the long term as well as being useful in collecting important health data about the user. To achieve this, we are incorporating findings from the literature about what aspects are important for long term engagement.

The processing challenges for this work package are the robustness of input interpretation, flexibility *and* personal adaptation of the generated output, handling different situational contexts for both the physical and graphical embodiment of PAL, and allowing for interactions with one child alone or in the presence of an audience of multiple children. Additional challenges are posed by the need for extendable thematic and linguistic coverage.

The core functional component developed in WP4 is a multimodal dialogue system with a repertoire of multimodal dialogue acts (combining verbal and non-verbal means) modulated by affect. Generation as well as interpretation will use parameterised dialogue acts as an interface schema to other modules to abstract away from specific aspects of, e.g., natural language or emotion expression. Based on the high-level targets from WP2, action selection from WP3, the dialogue state (including the latest child's input and interaction history) as well as a long-term memory, the multimodal output generation module decides which act to activate ("what to express") and how to realise it multimodally in the given context ("how to express it").

In order to avoid repetitiveness in the long term, it is important to have flexible dialogue strategies and a rich repertoire of verbal and non-verbal expressions to allow for variation. The multimodal input processing module interprets verbal and non-verbal input. Interpretation is guided by information from the user model and the strategic planning (WP2 & 3) and provides information back to them. First, verbal input is needed for the dialog interaction itself as the dialogue's flow takes input from the child to progress. Second, an interpretation of the child's affective state is needed for engagement analysis used in WP2 to adapt the high-level goal self-management goals. Third, feedback is needed for WP3 as basis for adapting a child's preference model, and the long-term memory.

1 Tasks, objectives, results

The overall objective of WP4 is to develop the technologies for personalised multimodal natural interaction serving to actively foster long-term engagement with the robot and its avatar. Voluntary long-term use is required as a prerequisite for other system objectives. This encompasses natural language interpretation and multimodal generation, as well as dialogue management.

1.1 Planned work

The objective of WP4 in year 4 was to support the extended functionality of the final system, and to improve its interactivity with respect to multimodal interaction, for targeted (i.e., goal-oriented) as well as non-targeted interactions, serving the purpose of increasing the adherence of the child to the system. To achieve this, the following steps were planned:

- Develop and improve language and dialogue support for new and existing activities, such as games, but also for explanations, tips, more feedback, etc.
- Integrate an off-activity talk module, together with natural language understanding, to elicit more child utterances, and to improve the connection of the child to the virtual agent
- Extend the Episodic Memory, exploiting the long-term memory better by covering more episodes, especially with information from the child
- Improve the dialogue management framework VOnDA to make it more attractive to new users
- Integrate an English ASR (automatic speech recognition) module as a proof of concept and link it to the OAT / interpretation module

All planned activities were successfully pursued and either resulted in an implementation or a research prototype for the topic in question. The following section will describe the work on these items and the results that have been achieved.

2 Actual work performed

2.1 Extension of dialogue functionality

In the final phase, several extensions to the system functionality required adaptation and enhancement of the dialogue functionality.

The biggest part in this respect are the new modules for supporting the children in achieving their tasks: The *Explainable AI*, targeted *Task suggestions* and the so-called *Tip of the day*. These modules guide the child towards better behaviour and improved knowledge for their active tasks. They do so by proactively suggesting tasks which can be achieved shortly and give an immediate reward, or supporting the learning by giving reasons for these suggestions or giving explanations for elements of the games, such as quiz questions.

The hospital flavour of the application has been improved to provide selection dialogues for the standalone Quiz and Break & Sort Games that can be played with the robot, and also for the Dance activity, where the child can play the dance moves he/she has created on his/her tablet, which will then be executed by the real NAO. In addition, the introductory part has been improved, providing more guidance to first-time users than in the last period.

The number of basic semantic structures (dialogue acts) that are used for natural language generation have again increased, from around 460 to 536 for English in the current prototype, not counting all variations in the arguments. The number of supported dialogue acts, and the number of different utterances that are produced for the three supported languages, again not exploiting the full set of possible arguments, is shown in table 1.

	Dutch	Italian	English
Covered Dialogue Acts	494	533	536
Generated Utterances	35735	62134	66090

Table 1: Number of utterances generated for the 557 currently supported dialogue acts (shallow semantic representations)

The increased number of dialogue acts in English is mainly due to the extension of the OAT prototype that was build for the PAL end event, and the Italian generation profits from coverage extensions already created in a previous project.

2.2 Episodic Memory

The *episodic memory* module (EMM) exploits information and interactions of the past to conduct short dialogues in convenient situations. The goal

of this module is to make the agent more believable [1, 11] and improve the social bond between the user and the agent by showing that there are recollections of past events, a property that is highly valued in human communication.

The EMM and the feedback module (which reacts immediately to events in the application, e.g., user input, or game results), directly exploit the dynamic information about user input and agent-user interaction contained in the database, showing the benefit of maintaining a long-term memory.

To make the EMM even more versatile and attractive, new episodes together with the corresponding natural language resources have been added, notably for regular sports activities, physical conditions such as sickness, and exceptional emotional states. Additionally, sparse or missing usage of the timeline or the application itself for a longer period of time are also noted and discussed with the user.

2.3 Off-Activity-Talk

We integrated an enhanced prototype of the off-activity talk (OAT) [6, 8] module into the system, which consists of two sub-modules. The first attempts to elicit information from the user using so-called *self disclosures*. The robot actively starts in phases of prolonged inactivity, prompting the user if she/he is interested in knowing something about the virtual agent. If the user agrees, the agent continues by revealing some facts of itself and then prompts the child for similar experiences. The research for this part was done in year 2, an enhanced version of the experimental system has been integrated in the final system, requiring larger extensions of the ontology and the dialogue strategies.

The goal of this module was to elicit some more verbose utterances from the children during the experiment, which in turn would allow to build a data-driven extension for longer dialogues. Unfortunately, only in 40 of 452 Italian sessions and in 54 of 233 dutch sessions children answered at all, and only 15 respectively 11 contained meaningful answers.

The second sub-module is an improved version of the OAT module of year 3 that uses WordNet information to conduct more flexible dialogues. This part is still very experimental, and we found that it will require a much stronger open-domain interpretation module and even more background knowledge, e.g., from VerbOcean[2], to be able to produce free conversations. Currently, we are trying to also exploit dbpedia, but that requires internet access, while the current prototype could rely fully on local resources. To make this work in the integrated system, we improved the NLU module, adding new training data and using its already available functionality for morphological tagging and lemmatisation. The results of the lemmatisa-

[2] `http://demo.patrickpantel.com/demos/verbocean/`

tion help to identify the right synsets (concepts) in WordNet, since they are labelled with the lemmatised forms.

Some example dialogues can already be demonstrated, and were shown at the PAL end event in the Netherlands, in conjunction with automatic speech recognition.

2.4 Robust Interpretation Module

As already described, the NLU module was improved in several ways. Firstly, we added more training data to be able to process a wider range of utterances. Secondly, we passed the results from lemmatization and morphological tagging on to the interpretation module in the dialogue manager to support the social talk.

In addition, we integrated a parser for SRGS grammars (Speech Recognition Grammar Specification)[3]. The integrated parser can use the XML as well as the ABNF (augmented Backus-Naur form) grammars, and contains extensions to the original specification, such as the possibility of matching regular expressions, and a more flexible use of the semantic content of non-terminals in a rule. While it is in general not possible to provide very general grammars with such a formalism, it is very useful to implement a core set of utterances that have little variation, and the results are absolutely reliable, in contrast to systems learned from data. Additionally, such grammars can be used for rapid prototyping of limited dialogue systems. In the PAL system, if this parser returns an analysis, it is always preferred over possible analyses coming from the NLU. We also plan to add this parser to the base functionality of VOnDA.

We also conducted research in the area of multi-task learning for dialogue with deep neural networks. The aim is to reduce the amount of necessary training data by jointly learning the objective functions of several overlapping tasks, such as dialogue state tracking and argument identification. Multi-task learning has proven effective in many NLP areas, and the achieved results show some promising effects. Due to the very diverse nature of the available corpora and, as a consequence, limited experiments, there is no definite trend towards the right neural network architecture yet. The results are presented in [9], a paper to be submitted to ACL 2019 is currently prepared.

2.5 ASR Integration

We studied the new EU General Data Protection Regulation and the information provided by the Automatic Speech Recognition (ASR) cloud service providers Nuance and Google concerning data protection. From the available information, we could not deduce that we can safely use them as back

[3] `https://www.w3.org/TR/speech-grammar/`

ends for ASR. This led us to the conclusion of dropping our goal to use ASR for the upcoming prototype, since the work package does not have the resources of setting up a local ASR infrastructure. ASR would be interesting for open-domain talk since for short, foreseeable reactions, other means of input are more efficient. For open-domain application, however, the local services that could be set up are not of sufficient quality, at least not without a considerable amount of additional data collection for the languages in question, and retraining, which is beyond the scope of this project.

Given that the companies providing Cloud ASR services will have to implement the EU General Data Protection Regulation in the future, and that commercial applicants of the PAL technology will have more resources to handle the privacy and ethical issues, the use of this services may be feasible in the future, and in fact very desirable for a more natural communication.

If time permits, we will also investigate the possibility of using state-of-the-art free systems like Kaldi[4], but to set them up is not trivial and may exceed our means.

2.6 VOnDA framework

The VOnDA framework has undergone a major revision, fixing many problems that became obvious during implementation of the PAL dialogue functionality, but also improving the base functionality of the run-time system. In addition, the analysis stage of the compiler is now based on another parser generator (`bison` instead of `antlr`) and a better grammar, leading to more than 10 times faster compile times for the VOnDA source code.

We also added new syntax features, such as type inference for complex types like function types or complex collection types. This helps to support a better and more concise treatment functional programming expressions, and of relational RDF property values, which are presented to the implementer as sets of objects. The functionality of extended assignment operators like `+=` and `-=` has been extended for the use with those values. VOnDA now also features full Koenig binding for variables in `for` loop headers, which was not working in version 1.

Version 2, which was released October 1, 2018, also contains a strongly improved documentation and a small introductory example for first-timers. We also continued the work on a pre-compiler for hierarchical state automata into VOnDA code, for rapid prototyping as well as implementing strongly structured parts of dialogue functionality.

A paper describing VOnDA and its ecosystem has been accepted for the International Workshop on Spoken Dialogue Systems Technology (IWSDS 2019) in April. The paper is included in the non-public version of this document in appendix A.1.1.

[4] `http://kaldi-asr.org/`

Graphical Editor and Automata Pre-compiler To make creating the automata even more convenient, we adapted the graphical editor from VisualSceneMaker[5] for our purpose. Although the new version looks quite similar on the surface, the code has been massively reworked to allow the necessary adaptations. The editor now has a GitHub project of its own (`https://github.com/bkiefer/GraVE`).

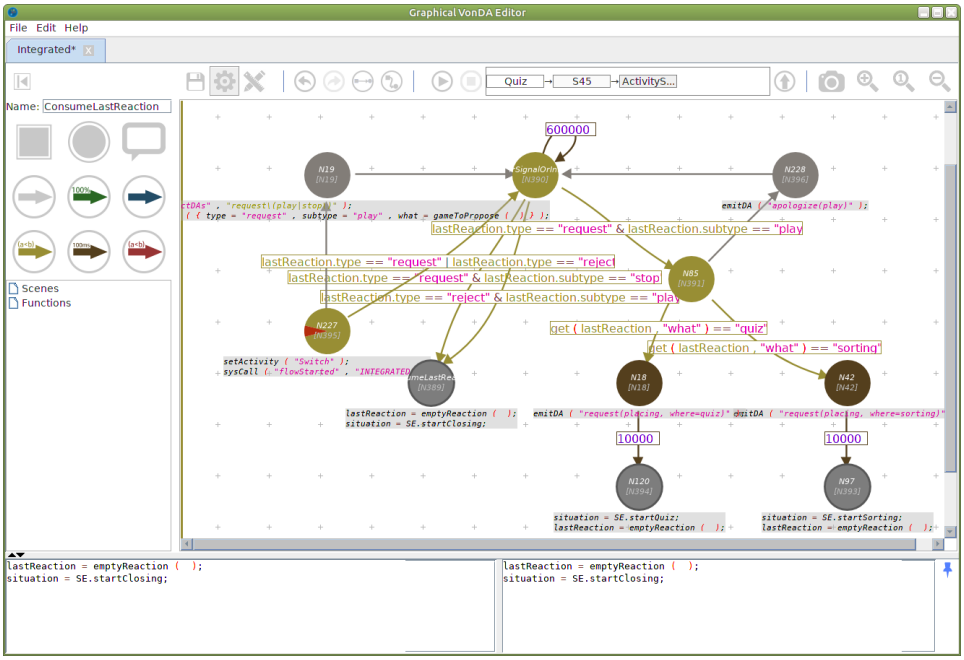


Figure 1: Screenshot of GraVE, the graphical VOnDA editor

2.7 Supporting work on the RDF storage

While not being really at the core of WP4, the decision to use an RDF database as the central data hub of the PAL system was driven by the conviction that this representation helps with the needs that arise from a dialogue system with long-term memory. Insofar has the extension of the database functionality always affected the work in WP4, and led to improvements in HFC whose value has immediate impact on the dialogue framework and will extend the project lifetime.

Many functionalities in the MyPAL app needed such extensions, starting with the goal achievement calculator, the episodic memory manager, the gamification with the reward system and high score table, up to the Dance manager that stores user-specific choreographies in the database. Based on

[5] `http://scenemaker.dfki.de/index.html`

the work for stream reasoning, we implemented an efficient synchronisation mechanism with the tablet app.

For the data access policies, we implemented an access control schema, based on a role system that exploits the membership of a user to a specific RDF class such as *Administrator* or *Professional* to determine if some information stored in the database is accessible to this person, using a set of declarative rules, e.g., that a professional can only look into the data of persons treated by him-/herself.

2.8 Behaviour Styles

In inter-human communication, non-verbal behaviour, especially body language, not only has a communicative meaning but also informs about the relationship and about likes and dislikes [10]. This means non-verbal behaviour is moderated depending on the context. For artificial agents (both robotic and virtual), to engage in meaningful interactions with humans, the importance of social intelligence is widely acknowledged [2, 13].

For the PAL Actor (robot and avatar), providing cognitive support, the expression of appropriate teaching style is crucial. Based on the hypothesis that non-verbal behaviour shapes interactions, we explored the effect of the PAL Actor's (robot and avatar) display of various teaching styles on children's interaction with the PAL system. The teaching styles were selected and appointed to specific activities based on educational psychology [14, 3]. The style expressions were designed based on results of earlier research [12].

For each behaviour that the PAL Actor can display, alternative versions were created expressing either a neutral, direct (instructor), or friendly (tutor/coach) style. Every activity in the MyPAL app was linked with a teaching style, for example, a quiz was considered a collaborative gaming activity and was appointed the friendly style whereas filling the timeline is more task oriented and therefore appointed the direct style.

The purpose of this feature is to explore the effect of variation in stylised non-verbal behaviour on system usage (i.e., frequency, duration, activity) and learning outcome (e.g., goal-progress, quiz responses).

3 Conclusions

The main work in year 4 was dedicated to support the system extensions requested by the experimenters, and to improve the overall interactivity. This was achieved by adding dialogue policies and linguistic resources for new activities, the Explainable AI module and an extended episodic memory.

The prototypical module of off-activity talk has been integrated into the systems, together with a module for self-disclosure, which makes use of the prototype for robust natural language understanding and a new natural language parser based on SRGS/VoiceXML grammars.

As a proof of concept, automatic speech recognition for English, being able to use either cloud or local services, has been integrated with the robust recognition and off-activity-talk.

The multi-party interaction that is part of this milestone manifests in the multi-player modes of the *memory* and *break & sort* games of the MyPAL app, and was already implemented in year 3.

Overall, the WP-tasks provided the following major outcomes:

- Extended dialogue and language support for new or extended activities in the PAL system
- A module for Off-Activity Talk with self disclosure
- A parser using SRGS/VoiceXML grammars for natural language analysis
- A second, strongly improved release of the VOnDA dialogue framework
- A graphical editor and compiler for hierarchical finite state machines into VOnDA modules

With these outcomes, work package 4 achieved milestone 4.4, *Support for multi-party interaction, further improved adaptivity.*

4 Future Extensions for Personalized Communication

The current version of the multimodal verbal communication already supports a fair amount of personalised interactions with the user based on personal data and information gathered otherwise, across almost all activities.

Previous studies ([5]) have shown that off-activity talk has a positive effect on system usage, both in duration and adherence to the goals. Integrating better such-like functionality in an extended version of the system will therefore most likely improve its acceptance.

The current knowledge-based approach for social interaction will require its own data collection experiment to explore if such an approach is feasible

for a clinical trial version of the PAL system. A Wizard-of-Oz experiment that would result in 5 hours of spoken dialogue (e.g. 20 children for 15 minutes each) would suffice to get enough seed data for extending the current system to a reasonable coverage. When collecting such a corpus with children, it is vital that all participants, including the parents, are aware that the conversation will be pre-processed also by human annotators to avoid privacy violations.

An alternative approach would be to train a persona-based chatbot[6] that uses user utterances to compute a persona representation to retrieve answers resp. avatar utterances from a preconstructed set. The task is then to find or create a dialogue corpus for the target languages that is sufficiently large and covers the most popular topics. A possible source to generate such corpora can be, e.g., crawled from social media or extracted from subtitle corpora (`www.subtitles.org`, [7]).

While all these procedures require a certain amount of human preparation and processing, reasonable results can be expected by investing 6 person months for data collection and preparation with medium to low skilled personnel, and 3 to 6 person months of a NLP and/or machine learning expert.

During the trial period, more real application data could be collected using a privacy-presevering procedure by providing the participants a portal to edit the collected data and explicitely authorize what will be visible to the experimenters.

[6] `https://medium.com/huggingface/how-to-build-a-state-of-the-art-conversational-ai-with-transfer-learning-2d818ac26313`

References

[1] Cyril Brom and J Lukavsky. Towards virtual characters with a full episodic memory ii: The episodic memory strikes back. In *Proc. empathic agents, AAMAS workshop*, pages 1–9, 2009.

[2] Terrence Fong, Illah Nourbakhsh, and Kerstin Dautenhahn. A survey of socially interactive robots. *Robotics and autonomous systems*, 42(3-4):143–166, 2003.

[3] Anthony F Grasha. A matter of style: The teacher as expert, formal authority, personal model, facilitator, and delegator. *College teaching*, 42(4):142–149, 1994.

[4] Bernd Kiefer, Anna Welker, and Christophe Biwer. VOnDA: A framework for ontology-based dialogue management. In *International Workshop on Spoken Dialogue Systems Technology (IWSDS)*. Springer, April 2019.

[5] Ivana Kruijff-Korbayová, Elettra Oleari, Ilaria Baroni, Bernd Kiefer, Mattia Coti Zelati, Clara Pozzi, and Alberto Sanna. Effects of Off-Activity Talk in Human-Robot Interaction with Diabetic Children. In *Ro-Man 2014: The 23rd IEEE International Symposium on Robot and Human Interactive Communication. IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), August 25-29, Edinburgh*, pages 649–654. IEEE, August 2014.

[6] Ivana Kruijff-Korbayová, Elettra Oleari, Ilaria Baroni, Bernd Kiefer, Mattia Coti Zelati, Clara Pozzi, and Alberto Sanna. Effects of off-activity talk in human-robot interaction with diabetic children. In *Ro-Man 2014: The 23rd IEEE International Symposium on Robot and Human Interactive Communication. IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), August 25-29, Edinburgh*, pages 649–654. IEEE, 8 2014.

[7] Pierre Lison and Raveesh Meena. Automatic turn segmentation for movie & tv subtitles. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 245–252. IEEE, 2016.

[8] Rosemarijn Looije, Mark A Neerincx, and Fokie Cnossen. Persuasive robotic assistant for health self-management of older adults: Design and evaluation of social behaviors. *International Journal of Human-Computer Studies*, 68(6):386–397, 2010.

[9] Sarah McLeod. Multi-task learning for goal-oriented spoken dialogue systems. Master's thesis, Saarland University, 2019.

[10] Albert Mehrabian. Nonverbal communication. In *Nebraska symposium on motivation*. University of Nebraska Press, 1971.

[11] Andrew M Nuxoll and John E Laird. Extending cognitive architecture with episodic memory. In *AAAI*, pages 1560–1564, 2007.

[12] Rifca Peters, Joost Broekens, and Mark A Neerincx. Robots educate in style: The effect of context and non-verbal behaviour on children's perceptions of warmth and competence. In *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 449–455. IEEE, 2017.

[13] Alessandro Vinciarelli, Maja Pantic, Dirk Heylen, Catherine Pelachaud, Isabella Poggi, Francesca D'Errico, and Marc Schroeder. Bridging the gap between social animal and unsocial machine: A survey of social signal processing. *IEEE Transactions on Affective Computing*, 3(1):69–87, 2012.

[14] Theo Wubbels, Marie-Christine Opdenakker, and Perry Den Brok. Let's make things better. In *Interpersonal relationships in education*, pages 225–249. Springer, 2012.

A Annexes

A.1 Accepted Papers

A.1.1 B. Kiefer, A. Welker, C. Biwer (2019), "VOnDA: A Framework for Ontology-Based Dialogue Management"

Abstract We present VOnDA, a framework to implement the dialogue management functionality in dialogue systems. Although domain-independent, VOnDA is tailored towards dialogue systems with a focus on social communication, which implies the need of a long-term memory and high user adaptivity. For these systems, which are used in health environments or elderly care, margin of error is very low and control over the dialogue process is of topmost importance. The same holds for commercial applications, where customer trust is at risk. VOnDA's specification and memory layer relies upon (extended) RDF/OWL, which provides a universal and uniform representation, and facilitates interoperability with external data sources, e.g., from physical sensors.

Relation to WP This is a strongly modified resubmission of the paper submitted to ACL 2018¸ which was not accepted for the demonstration section.

Making the results of the PAL project open source was one of the first decisions taken at the beginning of the project. We are taking one step towards this goal by providing a first open-source release of VOnDA and make it known in the computational linguistics community.

Availability Accepted for IWSDS 2019. Included only in the non-public version of this deliverable (Annex B.1).

A.2 Master Theses

A.2.1 Sarah McLeod (2019), "Multi-Task Learning for Goal-Oriented Spoken Dialogue Systems"

Abstract Dialogue Systems are an active area of research for Natural Language Processing (NLP). Advancements in machine learning and the increasing availability of corpora have facilitated advancements in language technologies that make it easier for humans to interact with systems through spoken languages. Conversational agents such as Siri and Alexa are quickly finding their way into every home and onto every cell phone.

The focus of this thesis is modern goal-oriented spoken dialogue systems, specifically, a methodology to use Multi-Task Learning (MTL) to bootstrap dialogue management is proposed. The typical approach in machine learning

is to focus on one task at a time. Large problems are segmented into smaller subtasks and each subtask is learned independently. Rather than learning a single task with a single loss function, MTL uses multiple loss functions to learn more than one task simultaneously to profit from generalizations in the data

Relation to WP This research was conducted at the MLT lab of DFKI, as an attempt to learn dialogue systems from a relatively moderate sized set of data. This effort fits into task 4.3: Techniques for extending linguistic and background knowledge, which with modern methods is implemented using machine learning methods in general, and deep neural networks in particular

This thesis is still under review. For that reason, it is not yet publicly available.

Availability Restricted. Included in the non-public version of this deliverable (annex C.1).

B Accepted papers

VOnDA: A Framework for Ontology-Based Dialogue Management

Bernd Kiefer and Anna Welker and Christophe Biwer

Abstract We present VOnDA, a framework to implement the dialogue management functionality in dialogue systems. Although domain-independent, VOnDA is tailored towards dialogue systems with a focus on social communication, which implies the need of a long-term memory and high user adaptivity. For these systems, which are used in health environments or elderly care, margin of error is very low and control over the dialogue process is of topmost importance. The same holds for commercial applications, where customer trust is at risk. VOnDA's specification and memory layer relies upon (extended) RDF/OWL[1] , which provides a universal and uniform representation, and facilitates interoperability with external data sources, e.g., from physical sensors.

1 Introduction

Natural language dialogue systems are becoming more and more popular, be it as virtual assistants such as Siri or Cortana, as Chatbots on websites providing customer support, or as interface in human-robot interactions in areas ranging from human-robot teams in industrial environments [17] over social human-robot-interaction [1] to disaster response [12].

A central component of most systems is the *dialogue manager*, which controls the (possibly multi-modal) reactions based on external triggers and the current internal state. When building dialogue components for robotic applications or in-car assistants, the system needs to take into account inputs in various forms, first and foremost the user utterances, but also other sensor input that may influence the dialogue, such as information from computer vision, gaze detection, or even body and environment sensors for cognitive load estimation.

Bernd Kiefer, Anna Welker, Christophe Biwer
German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany e-mail: kiefer@dfki.de,anna.welker@dfki.de,christophe.biwer@dfki.de

[1] Resource Description Framework https://www.w3.org/RDF/
Web Ontology Language https://www.w3.org/OWL/

In the following, we will describe VOnDA, an open-source framework initially developed to implement dialogue strategies for conversational robotic and virtually embodied agents. The implementation mainly took place in the context of the ALIZ-E and PAL projects, where a social robotic assistant supports diabetic children managing their disease. This application domain dictates some requirements that led to the decision to go for a rule-based system with statistical selection and RDF/OWL underpinning.

Firstly, it requires a lot of control over the decision process, since mistakes by the system are only tolerable in very specific situations, or not at all. Secondly, it is vital to be able to maintain a relationship with the user over a longer time period. This requires a long-term memory which can be efficiently accessed by the dialogue system to exhibit familiarity with the user in various forms, e.g., respecting personal preferences, but also making use of knowledge about conversations or events that were part of interactions in past sessions. For the same reason, the system needs high adaptability to the current user, which means adding a significant number of variables to the state space. This often poses a scalability problem for POMDP-based approaches, both in terms of run-time performance, and of probability estimation, where marginal cases can be dominated by the prominent situation. A third requirement for robotic systems is the ability to process streaming sensor data, or at least use aggregated high-level information from this data in the conversational system.

Furthermore, data collection for user groups in the health care domain is for ethical reasons even more challenging than usual, and OWL reasoning offers a very flexible way to access control.

VOnDA therefore specifically targets the following design goals to support the system requirements described before:

- Flexible and uniform specification of dialogue semantics, knowledge and data structures
- Scalable, efficient, and easily accessible storage of interaction history and other data, like real-time sensor data, resulting in a large information state
- Readable and compact rule specifications, facilitating access to the underlying RDF database, with the full power of a programming language
- Transparent access to standard programming language constructs (Java classes) for simple integration with the host system

VOnDA is not so much a complete dialogue management system as rather a fundamental implementation layer for creating complex reactive systems, being able to emulate almost all traditional rule- or automata-based frameworks. It provides a strong and tight connection to a reasoning engine and storage, which makes it possible to explore various research directions in the future.

In the next section, we review related work that was done on dialogue frameworks. In section 3, we will give a high-level overview of the VOnDA framework, followed by a specification language synopsis. Section 5 covers some aspects of the system implementation. Section 6 describes the application of the framework in the PAL project's integrated system. The paper concludes with a discussion of the work done, and further directions for research and development.

2 Related Work

The existing frameworks to implement dialogue management components roughly fall into two large groups, those that use symbolic information or automata to specify the dialogue flow (IrisTK [18], RavenClaw [3], Visual SceneMaker [7]), and those that mostly use statistical methods (PyDial [20], Alex [8]). Somewhat in between these is OpenDial [13], which builds on probabilistic rules and a Bayesian Network.

For reasons described in the introduction, VOnDA currently makes only limited use of statistical information. A meaningful comparison to purely learned systems like PyDial or Alex therefore becomes more complex, and would have to be done on an extrinsic basis, which we can not yet provide. We studied comparable systems focusing mainly on two aspects: the specification of behaviours, and the implementation of the dialogue memory / information state.

The dialogue behaviours in IrisTK and SceneMaker are specified using state charts (hierarchical automata). Additional mechanisms (parallel execution, history keeping, exception mechanisms like interruptive edges) make them more flexible and powerful than basic state charts, but their flexibility and generalisation capabilities are limited.

RavenClaw [3] uses so-called *task trees*, a variant of flow charts that can be dynamically changed during run-time to implement dialogue agents for different situations in the dialogue, and an *agenda*, which selects the appropriate agent for the current dialogue state. The resemblance to agent-based architectures using preconstructed plans is striking, but the improved flexibility also comes at the cost of increased complexity during implementation and debugging.

OpenDial [13] tries to combine the advantages of hand-crafted systems with statistical selection, using probabilistic rules which can be viewed as templates for probabilistic graphical models. The parameters for the models can be estimated using previously collected data (supervised learning), or during the interactions with reinforcement learning techniques. Being able to specify structural knowledge for the statistical selection reduces the estimation problem if only a small amount of data is available, and allows to explicitly put restrictions on the selection process.

3 High-Level System Description

VOnDA follows the Information State / Update paradigm [19]. The information state represents everything the dialogue agent knows about the current situation, possibly containing information about dialogue history, the belief states of the participants, situation data, etc., depending on the concrete system. Any change in the information state will trigger a reasoning mechanism of some sort, which may result in more changes in the information state, or outputs to the user or other system components.

VOnDA implements this paradigm by combining a rule-based approach with statistical selection, although in a different way than OpenDial. The rule specifica-

tions are close to if-then statements in programming languages, and the information state is realised by an RDF store and reasoner with special capabilities (HFC [10]), namely the possibility to directly use n-tuples instead of triples. This allows to attach temporal information to every data chunk [9, 11]. In this way, the RDF store can represent *dynamic objects*, using either *transaction time* or *valid time* attachments, and as a side effect obtain a complete history of all changes. HFC is very efficient in terms of processing speed and memory footprint, and has recently been extended with stream reasoning facilities. VOnDA can use HFC either directly as a library, or as a remote server, also allowing for more than one database instance, if needed. The initial motivation for using an RDF reasoner was our research interest

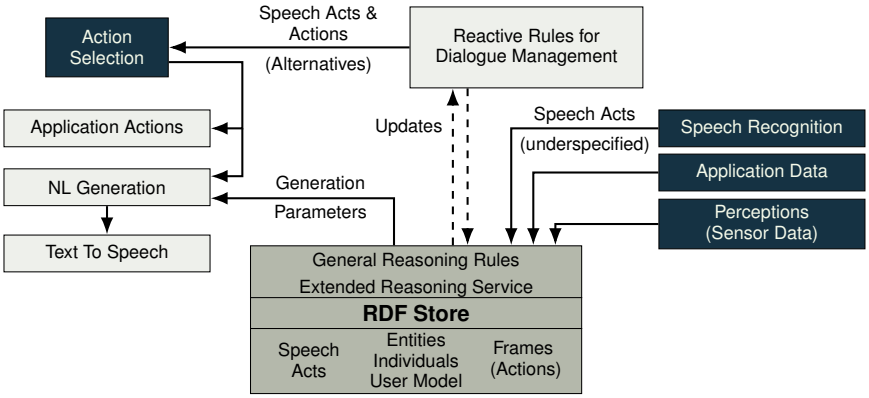


Fig. 1 VOnDA Architecture

in multi-session, long-term interactions. In addition, this also allows processing incoming facts in different layers. Firstly, there is the layer of custom reasoning rules, which also comprises streaming reasoning, e.g., for real-time sensor data, and secondly the reactive rule specifications, used mainly for agent-like functionality that handles the behavioural part. This opens new research directions, e.g., underpinning the rule conditions with a probabilistic reasoner.

The RDF store contains the terminological and the dynamic knowledge: specifications for the data types and their properties, as well as a hierarchy of dialogue acts, semantic frames and their arguments, and the data objects, which are instantiations of the data types. The data type specifications are also used by the compiler to infer the types for property values (see section 4), and form a declarative API to connect new components, e.g., for sensor or application data.

We are currently using the DIT++ dialogue act hierarchy [4] and shallow frame semantics along the lines of FrameNet [16] to interface with the natural language understanding and generation units. Our dialogue act object currently consist of a dialogue act token, a frame and a list of key-value pairs as arguments to the frame (`Offer(Transporting, what=tool, to=workbench)`). While this form

of shallow semantics is enough for most applications, we already experience its shortcomings when trying to handle, for example, social talk. Since the underlying run-time core is already working with full-fledged feature matrices, only a small syntax extension will be needed to allow for nested structures.

A set of *reactive condition-action rules* (see figure 4) is executed whenever there is a change in the information state. These changes are caused by incoming sensor or application data, intents from the speech recognition, or expired timers. Rules are labelled if-then-else statements, with complex conditions and shortcut logic, as in Java or C. The compiler analyses the base terms and stores their values during processing for dynamic logging. A rule can have direct effects, like changing the information state or executing system calls. Furthermore, it can generate so-called *proposals*, which are (labelled) blocks of code in a frozen state that will not be immediately executed, similar to closures.

All rules are repeatedly applied until a fixed point is reached where no new proposals are generated and there is no information state change in the last iteration. Subsequently, the set of proposals is evaluated by a statistical component, which will select the best alternative. This component can be exchanged to make it as simple or elaborate as necessary, taking into account arbitrary features from the data storage.

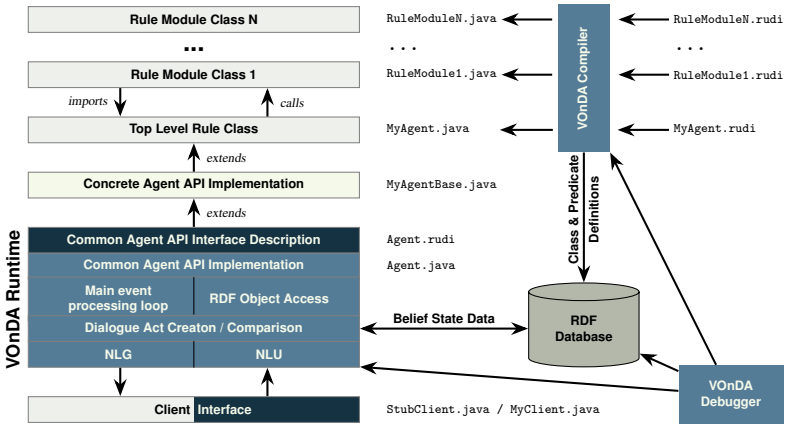


Fig. 2 A schematic VOnDA agent

A VOnDA project consists of an ontology, a custom extension of the abstract `Agent` class (the so-called *wrapper class*), a client interface to connect the communication channels of the application to the agent, and a set of rule files that are arranged in a tree, using `import` statements. The blue core in Figure 2 is the run-time system which is part of the VOnDA framework, while all elements above are application specific parts of the agent. A `Yaml` project file contains all necessary information for compilation: the ontology, the wrapper class, the top-level rule file and other parameters, such as custom compile commands.

The ontology contains the definitions of dialogue acts, semantic frames, class and property specifications for the data objects of the application, and other assertional knowledge, such as specifications for "forgetting", which could be modeled in an orthogonal class hierarchy and supported by custom deletion rules in the reasoner.

Every rule file can define variables and functions in VOnDA syntax which are then available to all imported files. The methods from the wrapper class are available to all rule files.

The current structure assumes that most of the Java functionality that is used inside the rule files will be provided by the `Agent` superclass. There are, however, alternative ways to use other Java classes directly, with support for the same type inference as for RDF classes.

4 Dialogue Specification Language

VOnDA's rule language at first sight looks very similar to Java/C++. However, there are a number of specific features which make it convenient for the implementation of dialogue strategies. Maybe the most important one is the handling of RDF objects and classes, which can be treated similarly to those of object oriented programming languages, including the (multiple) inheritance and type inference that are provided by the RDF class hierarchies.

```
1 user = new Animate;
2 user.name = "Joe";
3 set_age:
4 if (user.age <= 0) {
5   user.age = 15;
6 }
```

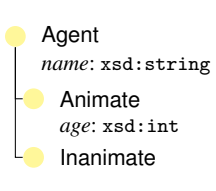


Fig. 3 Ontology and VOnDA code

Figure 3 contains an example of VOnDA code, and how it relates to RDF type and property specifications, schematically drawn on the right. The domain and range definitions of properties are picked up by the compiler and used in various places, e.g., to infer types, do automatic code or data conversions, or create "intelligent" boolean tests, such as the one in line 4, which will expand into two tests, one testing for the existence of the property for the object, and in case that succeeds, a test if the value is greater than zero. If there is a chain of more than one field resp. property access, every part is tested for existence in the target code, keeping the source code as concise as possible. Also, for reasons of brevity, the type of a new variable needs not be given if it can be inferred from the value assigned to it.

New RDF objects can be created with `new`, similar to Java objects; they are immediately reflected in the database, as are all changes to already existing objects.

Many operators are overloaded, especially boolean operators such as `<=`, which compares numeric values, but can also be used to test if an object is of a spe-

cific class, for subclass tests between two classes, and for subsumption of dialogue acts. There are two statements with a special syntax and semantics: `propose` and

```
1  if (!saidInSession(#Greeting(Meeting)) {
2    timeout("wait_for_greeting", 7000){ //Wait 7 secs before taking initiative
3      if (! receivedInSession(#Greeting(Meeting))
4        propose("greet") {
5          da = #InitialGreeting(Meeting);
6          if (user.name) da.name = user.name;
7          emitDA(da);
8        }
9    }
10
11   if (receivedInSession(#Greeting(Meeting))
12     propose("greet_back") { // We assume we know the name by now
13       emitDA(#ReturnGreeting(Meeting, name={user.name}));
14     }
15 }
```

Fig. 4 VOnDA code example

`timeout`. `propose` is VOnDA's current way of implementing probabilistic selection. All (unique) propose blocks that are in active rule actions are collected, frozen in the execution state in which they were encountered, such as closures known from functional programming languages. When rule processing stops, a statistical component picks the "best" proposal and its closure is executed.

`timeouts` also generate closures, but with a different purpose. They can be used to trigger proactive behaviour, or to check the state of the system after some time period, or in regular intervals. A timeout will only be created if there is no active timeout with that name.

Figure 4 also contains an example of the short-hand notation for shallow semantic structures (starting with `#`). Since they predominantly contain constant (string) literals, this is the default when specifying such structures. The special syntax in `user={user.name}` allows to insert the value of expressions into the literal, similar to an *eval*.

This section only described the most important features of VOnDA's syntax. For a detailed description, the reader is referred to the user documentation[2].

5 Compiler / Run-Time Library

The compiler turns the VOnDA source code into Java source code using the information in the ontology. Every source file becomes a Java class. Although the generated code is not primarily for the human reader, a lot of care has been taken in making it still understandable and debuggable. The compile process is separated into three

[2] `https://github.com/bkiefer/vonda/blob/master/doc/master.pdf`

stages: parsing and abstract syntax tree building, type checking and inference, and code generation.

The VOnDA compiler's internal knowledge about the program structure and the RDF hierarchy takes care of transforming the RDF field accesses into reads from and writes to the database. Beyond that, the type system, resolving the exact Java, RDF or RDF collection type of (arbitrary long) field accesses, automatically performs the necessary casts for the ontology accesses.

The run-time library contains the basic functionality for handling the rule processing, including the proposals and timeouts, and for the on-line inspection of the rule evaluation. There is, however, no blueprint for the main event loop, since that depends heavily on the host application. It also contains methods for the creation and modification of shallow semantic structures, and especially for searching the interaction history for specific utterances. Most of this functionality is available through the abstract `Agent` class, which has to be extended to a concrete class for each application.

There is functionality to directly communicate with the HFC database using queries, in case the object view is not sufficient or too awkward. The natural language understanding and generation components can be exchanged by implementing existing interfaces, and the statistical component is connected by a message exchange protocol. A basic natural language generation engine based on a graph rewriting module is already integrated, and is used in our current system as a template based generator. The example application also contains a VoiceXML based interpretation module.

Debugger / GUI

VOnDA comes with a GUI [2] that helps navigating, compiling and editing the source files belonging to a project. It uses the project file to collect all the necessary information.

Upon opening a project, the GUI displays the project directory (in a *file view*). The user can edit rule files from within the GUI or with an external editor like Emacs, Vim, etc. and can start the compilation process. After successful compilation, the project view shows what files are currently used, and marks the top-level and the wrapper class files. A second tree view (*rule view*) shows the rule structure in addition to the module structure. Modules in which errors or warnings were reported during compilation are highlighted, and the user can quickly navigate to them using context menus.

Additionally, the GUI can be used to track what is happening in a running system. The connection is established using a socket to allow remote debugging. In the rule view, multi-state check boxes are used to define which rules should be observed under which conditions. A rule can be set to be logged under any circumstances, not at all or if its condition evaluated to true or to false. Since the rules are represented in a tree-like structure, the logging condition can also be set for an entire subgroup

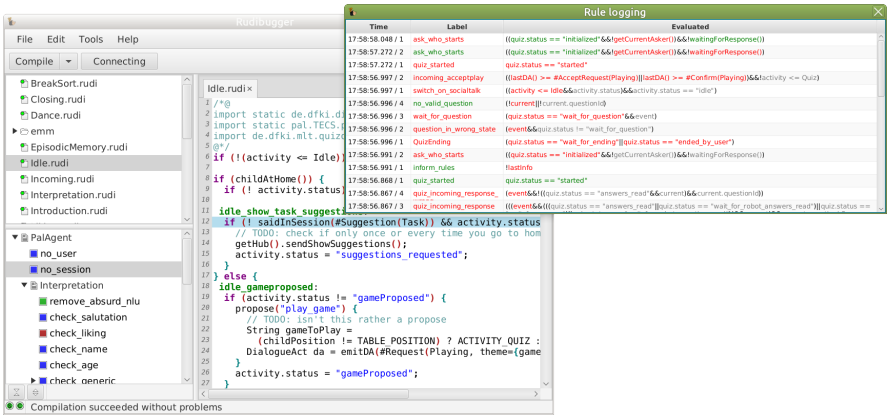


Fig. 5 The VOnDA GUI window

of rules, or for a whole module. The current rule logging configuration can be saved for later use.

The *logging view* displays incoming logging information as a sortable table. A table entry contains a time stamp, the rule's label and its condition. The rule's label is coloured according to the final result of the whole boolean expression. Each base term of the condition is coloured accordingly, or greyed out if short-cut logic led to premature failure or success of the expression. Inspecting the live system helps pin-point problems when the behaviour is not as expected. The log shows how the currently active part of the information state is processed, and the window offers easy navigation using the mouse from the rule condition to the corresponding source code.

6 Applications

VOnDA is used in the integrated system of the EU project PAL [15], which uses human-robot interaction to support children with diabetes type 1 in coping with their disease. Children interact with a real NAO robot[3], or with an Android app that connects to the core system and exhibits a virtual character that is as similar to the robot as possible, also in its behaviour.

The dialogue component, which is largely responsible for the agent's behaviour, is implemented using the VOnDA framework. In addition, HFC, the RDF store that VOnDA builds upon, is the main database of the system, storing all relevant information and being the central data exchange hub. The system runs as a cloud-based robotic solution, spawning a new system instance for every user. It has been success-

[3] Softbank Robotics `https://www.ald.softbankrobotics.com`

fully tested with more than 40 users at a time on a medium sized virtual machine[4] with only moderate load factors, giving a positive indication of the scalability of HFC and the VOnDA approach.

There are two helper modules integrated into the dialogue component which quite extensively exploit the connection between the database and the rule part, namely the *Episodic Memory* and the *Targeted Feedback*. While the targeted feedback reacts to current events in the running session, like entering a bad or good glucose value, or the current achievement of a task, the episodic memory aggregates data from the past and eventually converts them into so-called episodes that are used for interactions in subsequent sessions. Both are only triggered if relevant changes in the database occur, for example incoming data from the MyPAL app about games or achievements, and serve different conversational purposes, namely showing familiarity with the user and her/his everyday life, versus reacting to current positive or negative incidents.

VOnDA has also been used in a recent project aiming to implement a generalised, ontology-based approach to open-domain talk [21]. The Smoto system uses an additional HFC server running WordNet [14, 6] as semantic database, thereby gaining knowledge about semantic concepts that can be used in the dialogue and to find appropriate reactions on arbitrary user input.

7 Discussion and Further Work

We believe that there are still many interesting application areas for hybrid statistical and hand crafted systems, e.g., if they are relatively small, or there is little domain-specific data available. Many currently deployed systems that build on much simpler technology like VoiceXML can certainly profit from hybrid approaches such as OpenDial or VOnDA.

VOnDA is under active development. We designed it such that it can be integrated in most applications and opens many ways for improvements and additions. As a rule-based framework that is close to being a programming language, VOnDA is able to completely emulate the automata-based frameworks. In fact, we are currently working on a graphical editor à la SceneMaker and the precompilation of hierarchical state charts into VOnDA code. We hope this will facilitate the implementation of new applications for inexperienced users and help with rapid prototyping, while retaining the greater flexibility and modularization capabilities. In this way, we combine the intuitive way of specifying simple strategies with the full flexibility of the framework.

VOnDA could also be used to implement modules that simulate the agents of RavenClaw. To get a functionality similar to RavenClaw's agenda, its action selection module would have to be implemented as a dialogue state tracker, activating the most probable agent at each dialogue step.

[4] 4 core Xeon E5-2683@2.00GHz, 16 GB RAM

Using the well-established RDF/OWL standard as specification layer makes it very easy to add or change application specific data structures, especially because of the existing tool support. We already use the reasoning facilities for type and partially for temporal inference, but given the possibility of attaching also confidence or credibility information to the RDF data, a more integrated probabilistic approach with soft preconditions could be implemented, e.g., on the basis of Dempster-Shafer theory [5]. Moreover, additional meta knowledge, such as trustworthiness or validity periods could be declared using multiple inheritance, which opens many interesting research directions.

Other next steps will be the addition of default adaptors for obviously needed external modules like automatic speech recognition, more flexible language understanding, and the like. We will also work on the improvement of the GUI, including features such as a watch window and/or a timeline to track changes of specific values in the database, and a tool that analyses the dependencies between rules on the basis of the conditions' base terms.

From the research perspective, there are two very interesting lanes: integrating probabilistic reasoning as a first-class option, which is directly integrated with the rule conditions, and adding an additional layer to facilitate the implementation of BDI-like agents, to study the connections and dependencies between conversational and non-conversational behaviours.

Source Code and Documentation

The VOnDA core system can be downloaded at `git@github.com:bkiefer/vonda.git`. The main page has detailed instructions for the installation of external dependencies. The debugger currently lives in a separate project: `git@github.com:yoshegg/rudibugger.git`. Both projects are licensed under the Creative Commons Attribution-NonCommercial 4.0 International License[5], and are free for all non-commercial use. A screen cast showing the GUI functionality and the running PAL system is available at `https://youtu.be/nSotEVZUEyw`.

Acknowledgements

The research described in this paper has been funded by the Horizon 2020 Framework Programme of the European Union within the project PAL (Personal Assistant for healthy Lifestyle) under Grant agreement no. 643783.

This paper is dedicated to our colleague and friend Hans-Ulrich Krieger, the creator of HFC. Hope you found peace, wherever you are.

[5] `http://creativecommons.org/licenses/by-nc/4.0/`

References

1. ALIZ-E project. http://www.aliz-e.org/ (2010). URL `http://www.aliz-e.org/`
2. Biwer, C.: rudibugger - Graphisches Debugging der Dialogmanagementtechnolgie VOnDA. Bachelor's Thesis, Saarland University (2017). DOI 10.13140/RG.2.2.36556.31368
3. Bohus, D., Rudnicky, A.I.: The RavenClaw dialog management framework: Architecture and systems. Computer Speech & Language **23**(3), 332–361 (2009)
4. Bunt, H., Alexandersson, J., Choe, J.W., Fang, A.C., Hasida, K., Petukhova, V., Popescu-Belis, A., Traum, D.R.: ISO 24617-2: A semantically-based standard for dialogue annotation. In: LREC, pp. 430–437. Citeseer (2012)
5. Dempster, A.P.: The Dempster-Shafer calculus for statisticians. International Journal of approximate reasoning **48**(2), 365–377 (2008)
6. Fellbaum, C.: WordNet. Wiley Online Library (1998)
7. Gebhard, P., Mehlmann, G., Kipp, M.: Visual SceneMaker—a tool for authoring interactive virtual characters. Journal on Multimodal User Interfaces **6**(1-2), 3–11 (2012)
8. Jurčíček, F., Dušek, O., Plátek, O., Žilka, L.: Alex: A statistical dialogue systems framework. In: International Conference on Text, Speech, and Dialogue, pp. 587–594. Springer (2014)
9. Krieger, H.U.: A temporal extension of the Hayes/ter Horst entailment rules and an alternative to W3C's n-ary relations. In: Proceedings of the 7th International Conference on Formal Ontology in Information Systems (FOIS), pp. 323–336 (2012)
10. Krieger, H.U.: An efficient implementation of equivalence relations in OWL via rule and query rewriting. In: Semantic Computing (ICSC), 2013 IEEE Seventh International Conference on, pp. 260–263. IEEE (2013)
11. Krieger, H.U.: A detailed comparison of seven approaches for the annotation of time-dependent factual knowledge in rdf and owl. In: Proceedings 10th Joint ISO-ACL SIGSEM Workshop on Interoperable Semantic Annotation (2014)
12. Kruijff-Korbayová, I., Colas, F., Gianni, M., Pirri, F., de Greeff, J., Hindriks, K., Neerincx, M., Ögren, P., Svoboda, T., Worst, R.: TRADR project: Long-term human-robot teaming for robot assisted disaster response. KI-Künstliche Intelligenz **29**(2), 193–201 (2015)
13. Lison, P., Kennington, C.: Developing spoken dialogue systems with the OpenDial toolkit. SEMDIAL 2015 goDIAL p. 194 (2015)
14. Miller, G.A.: Wordnet: a lexical database for english. Communications of the ACM **38**(11), 39–41 (1995)
15. PAL: The PAL Project Website. `http://www.pal4u.eu/` (2018). Last access: 11.12.2018
16. Ruppenhofer, J., Ellsworth, M., Petruck, M.R., Johnson, C.R., Scheffczyk, J.: FrameNet II: Extended theory and practice. Institut für Deutsche Sprache, Bibliothek (2016)
17. Schwartz, T., Zinnikus, I., Krieger, H.U., Bürckert, C., Folz, J., Kiefer, B., Hevesi, P., Lüth, C., Pirkl, G., Spieldenner, T., et al.: Hybrid teams: flexible collaboration between humans, robots and virtual agents. In: German Conference on Multiagent System Technologies, pp. 131–146. Springer (2016)
18. Skantze, G., Al Moubayed, S.: IrisTK: a statechart-based toolkit for multi-party face-to-face interaction. In: Proceedings of the 14th ACM international conference on Multimodal interaction, pp. 69–76. ACM (2012)
19. Traum, D.R., Larsson, S.: The information state approach to dialogue management. In: Current and new directions in discourse and dialogue, pp. 325–353. Springer (2003)
20. Ultes, S., Barahona, L.M.R., Su, P.H., Vandyke, D., Kim, D., Casanueva, I., Budzianowski, P., Mrkšić, N., Wen, T.H., Gašić, M., et al.: Pydial: A multi-domain statistical dialogue system toolkit. Proceedings of ACL 2017, System Demonstrations pp. 73–78 (2017)
21. Welker, A.: Wissensbasiertes Dialogmanagement für Social Talk. Bachelor's Thesis, Saarland University (2017). DOI 10.13140/RG.2.2.16423.65441

C Master Theses

Multi-Task Learning for Goal-Oriented Spoken Dialogue Systems

Sarah McLeod, Saarland University
smcleod@coli.uni-saarland.de

March 11, 2019

Contents

List of Figures

List of Tables

Introduction

Dialogue Systems are an active area of research for Natural Language Processing (NLP). Advancements in machine learning and the increasing availability of corpora have facilitated advancements in language technologies that make it easier for humans to interact with systems through spoken languages. Conversational agents such as Siri and Alexa are quickly finding their way into every home and onto every cell phone. The history of spoken dialogue systems dates back to the first chat bot, ELIZA, developed in 1966. Since then spoken dialogue systems have been used for travel planning (Georgila, Lemon, Henderson, and Moore 2009), finding restaurant information (Su et al. 2016), in car navigation (Lemon, Georgila, et al. 2006), and intelligent robots (Lemon, Gruenstein, and Peters 2002). The focus of this thesis is modern goal-oriented spoken dialogue systems. Specifically, a methodology to use Multi-Task Learning to bootstrap dialogue management is proposed. This work will support the development of dialogue systems for human-robot interaction in the Talking Robots group in the German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) Multilingual Language Technologies Lab.

To give context to the research goals, a brief overview and history of spoken dialogue systems and Multi-Task Learning (MTL) is provided. The section on spoken dialogue systems closely follows Chapter 29 of (Jurafsky and Martin 2017) with input from (D.-N. Chen et al. 2016). We also summarize important topics in MTL presented in (Caruana 1998), (Y. Zhang and Yang 2017) and (Reuder 2017a). The proposal section outlines the motivation for the thesis, available data, and work plan. The experimentation section includes statistics on the data used, descriptions of the neural architectures that were tested, and results of model evaluation.

Spoken Dialogue Systems

A spoken dialogue system is a computer system that enables human computer interaction, primarily through speech. Spoken dialogue systems can generally be grouped into two categories: task oriented systems and non-task oriented systems, also called chatbots. task-oriented (or goal-oriented, or task-based) systems are designed for conversations with users where the goal is to facilitate completion of a specific task, for example booking a flight or making a

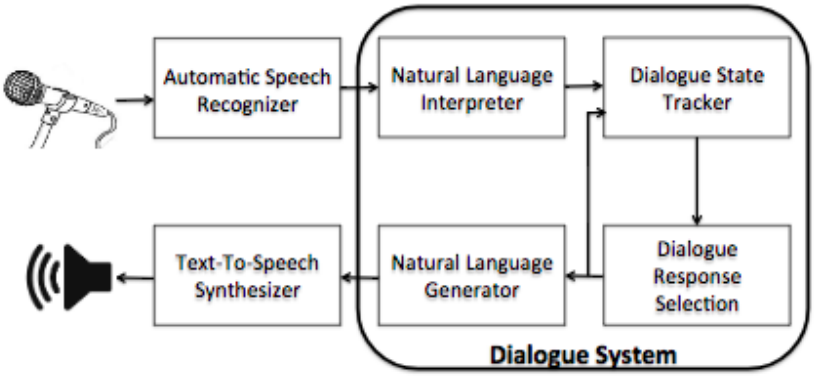


Figure 1: Components of a dialogue system. Image taken from (Serban, Lowe, et al. 2015)

restaurant reservation. These systems have specific measures of success that drive the actions taken by a dialogue manager. Chatbots are designed to engage with users in unstructured conversations that mimic the chit-chat observed in human-human communication. Chatbots are intended for longer dialogues, whereas task-oriented dialogues to date are typically shorter, although not by definition. In both cases the dialogue system must be able to understand the user, decide how to respond, and be capable of conducting a conversation beyond just question answering. Applications for spoken dialogue systems include travel planning, automatic call routing, tutoring systems, personal assistants such as Siri, Cortana, Alexa, and systems a user can query to get information about restaurants or local weather.

Spoken dialogue systems are either rule-based or corpus based. The first rule-based systems include ELIZA (Weizenbaum 1966) and PARRY (Colby, Weber, and Hilf 1971), ELIZA being the most famous early chatbot. Both systems were built with a focus on clinical psychology, but ELIZA was built to simulate a Rogerian Psychologist; PARRY to study schizophrenia. In Rogerian psychology the goal is to draw a patient out by reflecting patients statements back at them. ELIZA mimicked this by applying a set of pattern/transformation rules. For example, if the patient were to say, "You hate me" a rule would transform this into the system response, "What makes you think I hate you". The PARRY system, which appeared a few years after ELIZA, included a model of its own mental state as well as affect variables for its level of anger and fear. Topics identified in the input would cause the agent's level of anger or fear to rise or fall, affecting the responses it would give. An interesting historical note - the PARRY system is the first known system to pass the Turing test; psychiatrists couldn't distinguish text transcripts of interviews with PARRY from transcripts of interviews with real paranoids, and this was in 1972! (Colby, Weber, and Hilf

1971).

The focus of this thesis is goal-oriented dialogue systems, however a brief description of chatbots is included here to help distinguish the important differences between the two technologies. Corpus based chatbots fall into two subcategories: those based on information retrieval and those based on machine learning and the sequence transduction model. The principle behind information retrieval based chatbots is to respond to a user's turn X by repeating some appropriate turn Y from a corpus of natural (human) text. Alternatively, one can treat response generation as a task of transducing from the user's turn to the system's turn. These models for response generation use current sequence to sequence (seq2seq) models, with some modifications to the basic seq2seq model needed for response generation. An important difference between chatbots and task-oriented systems is that chatbots do very little modeling of the conversational context. Instead they determine the best response given the users immediate input. For this reason they are frequently referred to as *response generation* systems. (Jurafsky and Martin 2017) describe the ELIZA and PARRY algorithms in more detail and provide recent examples of corpus based chatbots.

Modern goal-oriented dialogue systems are based on a domain ontology, or a structure that represents the information the system can extract from a user. The domain ontology is captured in a frame, denoted as characteristic set of features, called slots, and their values. The frame-based architecture was first introduced with the GUS system in 1977 (Bobrow et al. 1977) and elements of that architecture are still used today. Figure 1 shows the typical components of a modern goal-oriented spoken dialogue system. If the system interacts with a user through speech the automatic speech recognition component converts the user's speech into text, and the Text-to-Speech unit converts the text of the system response to speech. If the dialogue system interacts with users through text alone then these components can be left out. The natural language understanding unit converts the text to a frame representation. Actions such as domain classification, user intent (dialogue act) classification, or entity extraction are also commonly done as part of natural language understanding. The dialogue manager controls the structure of the dialogue and contains two subcomponents: the dialogue state tracker and the dialogue response selection. The state tracker uses the history of user inputs and intent labels to compute a distribution over possible dialogue states for the current turn. The dialogue response selection takes as input the current state and outputs the system response.

When designing dialogue systems careful consideration should be given to dialogue control. The speaker that has the control in a conversation is said to have the initiative. In human-to-human conversation the initiate switches back and forth between participants. This is called mixed-initiative. Early frame-based dialogue systems used system-initiative, where the system asked questions or responded with statements to clarify user input. This allowed the system to extract information and fill slots step by step. In these systems users are expected to respond only to the last utterance given by the system, so they're limited to a response with only a few words or phrases. Early frame-based system-initiative dialogue systems used finite state machines hand designed by a dialogue de-

signer. This has the advantage of the system always knowing which utterance the user is responding to, but it is far from natural human conversational structure. These systems can take several turns to complete even simple tasks and are often quite frustrating for users. In contrast, in user-initiative the user has control of the dialogue and it is the system's responsibility to respond to user utterances. This allows the user to converse with the system more naturally, but requires the ASR and NLU units to process a wider vocabulary and grammar. In current research the focus is mixed-initiative systems, where the agent has control but the user has the ability to provide (additional or unsolicited) information or change the task. This requires more complicated modeling of turn-taking.

The architecture of modern goal-oriented spoken dialog systems is shown in Figure 1. Early work employed the use of hand written rules for many system components. Recent advances in machine learning and the existence of larger and more thoroughly annotated corpora have facilitated a shift towards systems where components are corpus trained. In fact each component itself represents active areas of research. Domain classification (Hakkani-Tur et al. 2016), intent classification (Deng et al. 2012), (Hakkani-Tur et al. 2016) and slot filling (Hakkani-Tur et al. 2016),(X. Li et al. 2017), (D.-N. Chen et al. 2016) are all active areas for natural language understanding; state tracking (Zhao and Eskenazi 2016), (J. Williams et al. 2013),(J. D. Williams 2012) and dialogue policy optimization (Wen et al. 2016),(Su et al. 2016), (J. D. Williams and Zweig 2016),(Zhao and Eskenazi 2016) are also active areas of research in dialogue management. (H. Chen et al. 2017) give a detailed overview of deep learning for spoken dialogue systems and cite many more examples of research for each system component.

Successful goal-oriented dialogue systems must accurately determine the intent(s) of a user, understand and identify the relevant information they have provided, and, based on that information, select the appropriate response at each turn in the conversation. Modern goal-oriented dialogue systems model conversation as a partially observable Markov decision process (POMDP) (Gasic and Young 2014) and research into statistically optimizing dialogue management with Reinforcement Learning (RL) is an active area of research (Gasic and Young 2014; Lemon and Pietquin 2007; Bordes and Weston 2016). However, learning optimal dialogue policies with RL can be challenging since large state and action spaces are needed, and this requires large amounts of training data (Lemon and Pietquin 2007; Wen et al. 2016; X. Li et al. 2017). Additionally, networks trained with RL learn in a trial-and-error process, guided by a potentially delayed reward function. This trial-and-error process can lead to poor performance in the early training stages, which in turn can lead to a negative user experience (Su et al. 2016). For this reason many systems still rely on carefully hand-crafted rules or corpus trained user-simulators to initialize dialogue policies.

Recently, supervised learning has been used for pre-training of dialogue policies (Su et al. 2016), (Henderson, Lemon, and Georgila 2007), (J. D. Williams and Zweig 2016), however the previous approaches only considered one aspect of

dialogue during training. In discourse multiple components of linguistic structure interact and co-constrain one other (**Grosz**). This gives rise to the question whether it would be beneficial to view dialogue policy training as a multi-task learning (MTL) problem. MTL is an active area of research and has been shown to improve performance on a number Natural Language Processing (NLP) tasks Reuder 2017b; Y. Zhang and Yang 2017. In this work a method to use MTL to further bootstrap the initialization of optimal dialogue policies is proposed. The next section describes multi-task learning in more detail.

Multi-Task Learning

The typical approach in machine learning is to focus on one task at a time. Large problems are segmented into smaller subtasks and each subtask is learned independently. In many ways it makes sense to break up complicated problems into smaller, more approachable tasks. This is a common methodology to begin to tackle large and difficult problems. However, some argue (Caruana 1998) (Waibel, Sawai, and Shikano 1989) that if this is the default in statistical learning then we're ignoring information in related tasks that can help the system learn a primary task. Rather than learning a single task with a single loss function, MTL uses multiple loss functions to learn more than one task simultaneously.

The primary goal of MTL is to improve generalization. To achieve this multiple tasks are learned simultaneously while sharing an underlying representation. In artificial neural networks this is commonly expressed as shared hidden layers with separate output layers for each task. This enables features to be learned which support multiple tasks. This also allows hidden units to specialize for specific tasks, or remain small where they are not relevant. (Reuder 2017a), (Caruana 1998) and (Y. Zhang and Yang 2017) give examples of MTL for neural and non-neural models, including decision trees, linear models, and Baysian algorithms. The research proposed in this thesis is focused on neural models, therefore the remaining discussion on MTL assumes an artificial neural network trained with backpropagation.

An inductive learner is one that uses additional information to improve performance on a primary task. MTL is one way to introduce inductive bias, Transfer Learning (TL) is another. MTL uses the training signals of related tasks to bias the learner towards a representation that explains more than one task. In TL the goal is to extract knowledge from one or more source tasks and transfer it to a target task (Pan and Yang 2009). In MTL, attention is given to all tasks, but in transfer learning the target task plays a more important role than the source tasks. Also, MTL assumes labeled data for each task. This is not always the case for TL.

How do we know the information from related tasks improves generalization? (Caruana 1998) hypothesized several ways in which adding additional task-specific output layers to a network could improve generalization performance: (1) adding noise to backpropagation sometimes improves performance and as

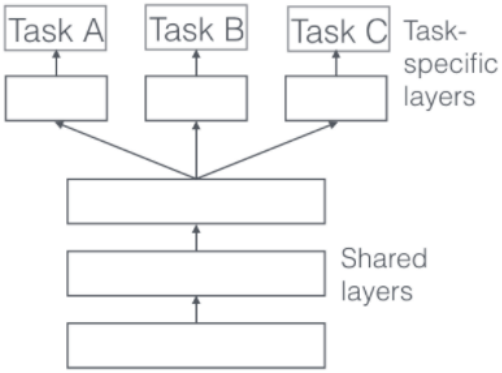


Figure 2: Hard Parameter Sharing in MTL. Image from (Reuder 2017a)

long as tasks are *uncorrelated* their contribution to the aggregate gradient may serve as a source of noise, (2) adding more tasks may change the weight update dynamics to favor networks with more tasks, and (3) since MTL shares hidden layers between all tasks this serves as a form of capacity restriction. (Caruana 1998) perform a "shuffle test" to prove the benefits of MTL rely on the training signals of auxiliary tasks being related to a main task. First recall that for each instance in the training set there is a set of input features, the main task training signal, and a set of extra task training signals. Prior to training they shuffle the extra task training signals among all the instances in the training set, i.e., randomly reassign the training signals for the extra tasks among the instances. This breaks the relationship between the main task and the extra tasks without altering other properties of the extra tasks. The idea being if MTL depends on the extra information in the training signals being meaningfully related to the main task, shuffling will eliminate that relationship, and thus should eliminate the benefits of MTL. If the benefits from MTL depend on some other property of having multiple outputs, shuffling will not affect this and the benefits should remain after shuffling. (Caruana 1998) performed this test on multiple experiments and showed that the performance of MTL with shuffled data reduces to the case where tasks are learned independently. To rule out reduced network capacity as the source of benefit in MTL, they also compare MTL networks to single-task learning networks of various sizes, or a single-task learned network that is optimized.

Multi-task learning (or multi-objective learning or jointly learning) is inspired by human learning; when learning new tasks we often apply knowledge learned from related tasks. For example, once a baby learns to identify faces it can use this ability when learning to identify other objects. To better understand the benefits of MTL we introduce the mechanisms underlying it, most of which were introduced in (Caruana 1998) and described in (Reuder 2017a). Again, these mechanisms assume a neural network trained with backpropagation.

Implicit data augmentation. MTL enables data amplification, an *effective*

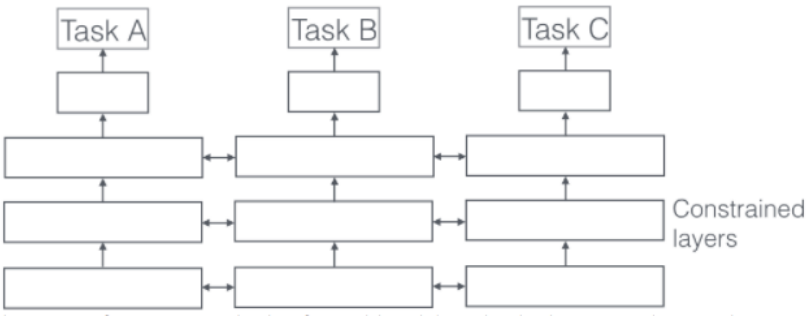


Figure 3: Soft Parameter Sharing in MTL. Image from (Reuder 2017a)

increase in a sample size due to the training signals of related tasks (Caruana 1998). The goal of any statistical learner is to learn an appropriate representation while ignoring data specific error. A network trained to learn multiple tasks can learn a shared representation that averages over the noise profile of each individual task.

Attribute selection or attention focusing. With high dimensional data or data with few training examples it can be difficult for a learner to distinguish important features. The training signals of related tasks can help the network determine which features are relevant, and which are not.

Eavesdropping. Consider the scenario where features F are easy to learn for task A but difficult to learn for Task B. This could be because B interacts with F in a more complicated way, or because other features are interfering with B's ability to learn F. A network that learns A will learn F, but a network that learns B may not. A network that learns both A and B allows B to eavesdrop on the hidden layers for A to learn F.

Representation bias. MTL prefers representations that multiple tasks prefer. Specifically, (Caruana 1998) showed that during backpropagation search is biased towards a representation at the intersection of what each task would learn individually.

In Deep Neural Networks MTL is generally done in one of two ways: hard parameter sharing or soft parameter sharing. Figures 2 and 3 show typical architecture for hard and soft parameter sharing. In hard parameter sharing tasks share hidden layers and each task has an individual output layer. For soft parameter sharing each task has it's own network layers and regularization is used to encourage the hidden layer representations to be similar. The ℓ_2 norm and trace norm are commonly used for regularization. (Reuder 2017b) and (Y. Zhang and Yang 2017) give an overview of modern architectures for MTL, including sluice networks and Deep Relationship Networks.

The essence of MTL is sharing while learning, therefore an important consideration is what to share and how to share it. (Y. Zhang and Yang 2017) distinguish between homogeneous-feature MTL and heterogeneous-feature MTL, as well as homogeneous MTL and heterogeneous MTL. They define MTL as:

> given m learning tasks $\{T_i\}_{i=1}^m$ where all of the tasks or a subset of them are related, *multi-task learning* aims to help improve the learning of T_i using the knowledge contained in m tasks.

In supervised learning T_i is accompanied by a training set D_i consisting of n_i training samples, i.e. $D_i = \{x_j^i, y_j^i\}_{j=1}^{n_i}$ where $x_j^i \in \mathbb{R}^{d_i}$ is the jth training instance in T_i and y_i is its label. When two tasks share the same feature space, i.e. d_i equals d_j for any $i \neq j$ this is referred to as homogeneous-feature MTL. The opposite is heterogeneous-feature MTL, where each task uses a different feature representation, for example text for language or pixels for images. In contrast, homogeneous MTL learns tasks using a single type of machine learning. Heterogeneous MTL tasks use many types of machine learning, for example supervised learning and reinforcement learning.

After deciding which tasks to share (Y. Zhang and Yang 2017) suggest deciding which characteristics of the data should be used for sharing. They group current research can be grouped into 3 categories: feature-based, instance-based, and parameter-based. Feature-based MTL aims to learn common features among different tasks as a way to share knowledge. Instance-based MTL aims to identify data instances in one task or tasks that are useful for other tasks, and then share this knowledge via the identified instances. Parameter-based MTL uses the model parameters (for example, coefficients in linear models) in one task to help learn model parameters in other tasks, for example, with regularization. (Y. Zhang and Yang 2017) note that existing MTL studies mainly focus on feature-based and parameter-based methods and few works belong to the instance-based method.

Another important step in MTL is to formalize concrete ways to share information. In feature-based methods networks learn feature representations common to all tasks, but (Y. Zhang and Yang 2017) distinguish between models that use shallow models (feature learning) and models that use deep neural networks (deep learning). They also further subdivide feature learning into feature transformation and feature selection. In feature transformation the input layers receive training instances from all tasks and the output layers have m outputs for each task. The transformation from the original representation to the learned one depends on the weights connecting the input layer and the hidden layer as well as the activation function adopted in the hidden units. In feature selection the model selects a subset of the original features as the new representation for all of the tasks.

In parameter-based MTL, there are five main approaches: low-rank approach, task clustering approach, task relation learning approach, dirty approach and multi-level approach. (Y. Zhang and Yang 2017) give an excellent summary. The low-rank approach interprets the relatedness of multiple tasks as the low-rank of the parameter matrix of these tasks. The task clustering approach assumes that all of the tasks form a few clusters and that tasks in the same cluster are related to each other. In many cases task relatedness is considered a priori information, however this is not always the case. The task relation learning approach aims to learn relations between tasks from data automatically

where the task relation can take the form of covariance. The dirty approach decomposes the model parameters of all the tasks into two components and the two components capture different forms of sparsity. Here a unified objective function includes regularizers and constraints on each of the components. Because of the characteristics of two regularizers on each of the components, a parameter matrix can eliminate unimportant features for all tasks when the corresponding rows in both components are sparse. The multi-level approach, an extension of the dirty approach, models the parameters as the sum of multiple components and instead of treating components independently as in the dirty approach, the multi-level approach can relate multiple components to model complex task structure.

In many MTL models the objective function is formulated as a loss function, such as hinge loss or squared loss, plus a regularization term. This term acts as a penalty term enforcing shared representations within the network. (Y. Zhang and Yang 2017) cite numerous examples of research and objective function formulation. (Y. Zhang and Yang 2017) also discuss comparisons between methods.

MTL has many applications in Natural Language Processing (NLP). (Toshniwal et al. 2017) use signals available in the speech recognition pipeline as auxiliary tasks. (Arık et al. 2017) predict phoneme duration and frequency profile as an auxiliary task for Text-to-Speech synthesis. For Machine Translation, the goal of MTL it to jointly train models to and from different languages using encoder-decoder network architecture. (Dong et al. 2015) jointly train the decoders, (Zoph and Knight 2016) jointly train the encoders, while (Johnson et al. 2016) jointly train both. (Reuder 2017b) and (Y. Zhang and Yang 2017) cite numerous additional examples of MTL for NLP, including for semantic parsing, information retrieval, and chunking.

Modern research into chatbots aims to learn a system end-to-end (O. and Le 2015),(Shang, Lu, and H. Li 2015), (Serban, Sordoni, et al. 2015). The problem is often treated as a sequence transduction task and many use the encoder-decoder network to automatically map from user input to system response. Recent work in frame-based systems aims to transfer this work to goal-oriented dialogues. It may seem like modeling goal-oriented dialogue is simpler because the task is constrained, but it does present some unique challenges, particularly because domain specific tasks require a more complicated modeling of system responses. For example, frame based systems must be able to ask clarifying questions and query a user database (Bordes and Weston 2016), (Wen et al. 2016). Because of these complexities many goal-oriented systems are initialized with a set of hand written rules. These rules ensure highly accurate system responses, but lack the coverage needed for scalability.

Most recently a community goal has been to train end-to-end goal-oriented dialogue systems. End-to-end systems map from input to output directly, without modeling specific subcomponents. (Wen et al. 2016) design a dialogue manager that is end-to-end trainable with supervised learning (SL), but is still modularly connected. (Su et al. 2016) learn optimal dialogue strategies from corpus data with SL, then improve the model with reinforcement learning (RL). (Pad-

makumar, Thomason, and Mooney 2017) train a semantic parser and policy network in batches, giving the policy network access to the updated semantic parser after every batch. (J. D. Williams and Zweig 2016) use a Recurrent Neural Network (RNN) with Long-Short-Term memory (LSTM) cells to automatically map from a sequence of user turns (represented as raw text and extracted entities) to actions, inferring a representation of state. The system is first optimized with SL, then continued training is done with RL. (Hakkani-Tur et al. 2016) use a bi-directional RNN with LSTMs to jointly model slot filling, intent determination, and domain classification for different domains. (D.-N. Chen et al. 2016) use a knowledge-guided structural attention network (K-SAN) to model intent prediction and slot filling simultaneously. Both published results on the ATIS corpus. Two recent papers that share goals closely related to this research are (Zhao and Eskenazi 2016) and (X. Li et al. 2017). (Zhao and Eskenazi 2016) jointly learn policies for state tracking and dialogue strategies, using Deep Recurrent Q-Network (DRQN). They learn an optimal policy that either generates a verbal response or modifies the current estimated dialog state based on the new observations.(X. Li et al. 2017) use a single RNN with LSTM to jointly learn user intent as well as slot filling; state tracking and dialogue policies are learned with end-to-end RL. Their dialogue manager is initiated by supervised learning of labels generated by a rule system. End-to-End training is continued with RL using a user simulator. Results are published on data from movie-ticket booking domain. These papers highlight the many new and different techniques being used to train goal-oriented spoken dialogue systems. We take this as a good sign, and believe the goal of our research is timely to the field.

The research in this thesis expands previous work on MTL for NLP. We explore hard parameter sharing and feature-sharing to build artificial neural networks that jointly learn user intent, slot-filling and optimal dialogue policies. In typical spoken dialogue systems a user utterance is processed serially, and the results of intent classification and slot-filling are passed to the dialogue manager. Our belief is that the information needed for reliable slot-filling and intent classification is also valuable for learning dialogue policies; therefore our proposed aproach is to learn all three tasks in parallel.

Recent MTL approaches for automatic speech recognition use the training signals available in the speech recognition pipeline as auxiliary tasks, and show improvement over single task systems, cf. (Arık et al. 2017),(Toshniwal et al. 2017). Our proposal was informed by this research, specifically the hypothesis that the supervised signals available in the dialogue management pipelines are similarly beneficial for dialogue policy optimization. If each task shares what it learns, then a learner may find it easier to learn them together rather than in isolation. Additionally, we hypothesize that the nature of MTL as implicit data augmentation will help bootstrap the learning of optimal dialogue policies from a medium sized initial data set.

Proposal

The following sections describe the motivation, available data, proposed architecture and work plan.

Motivation

The learning of optimal dialogue policies is traditionally framed as a reinforcement learning task and thus requires a significant amount of data to train, particularly for systems with large state and action spaces (Lemon and Pietquin 2007), (Wen et al. 2016), (X. Li et al. 2017). Additionally, dialogue policies are key to a systems ability to successfully respond to user requests, and any dialogue system trained from data needs to be robust to new interactions. Networks trained with reinforcement learning learn in a trial-and-error process, guided by a potentially delayed reward function. This trial-and-error process can lead to poor performance in the early training stages, which in turn can lead to a negative user experience (Su et al. 2016). For this reason many systems still rely on carefully hand-crafted rules or user simulators to initialize dialogue policies. Recently, supervised learning has been used as a method for supervised pre-training of reinforcement learning (Su et al. 2016), (Henderson, Lemon, and Georgila 2007), (J. D. Williams and Zweig 2016) however the previous approaches only considered one aspect of dialogue during training. In discourse multiple components of linguistic structure interact and co-constrain one other (**Grosz**). This gives rise to the question whether it would be beneficial to view dialogue policy training as a multi-task learning (MTL) problem. This thesis extends previous work and uses user-intent and slot-filling as auxillary tasks for supervised pre-training of dialogue policies. The belief is that MTL can be used to bootstrap dialogue policy optimization without the need for hand-written rules but still learn a robust dialogue manager that minimizes user frustration when fielded with users.

The goal of this work is to use MTL to learn dialogue system tasks simultaneously, rather than independently. Specifically, a supervised MTL architecture to learn optimal dialogue policies in goal-oriented dialogue systems, while treating user intent classification and slot-filling as auxiliary tasks is proposed.

Research Questions

(Caruana 1998) and (Y. Zhang and Yang 2017) cite numerous tasks where the shared representation learned with MTL improves generalization. The primary aim of this thesis is to extend previous work and explore when and how MTL can improve generalization for dialogue policy networks. Does simultaneous learning of user-intent and slot-filling improve supervised pre-training of optimal dialogue policies? Are either user-intent classification or slot-filling alone sufficient auxillary tasks for dialogue policy optimization? Do user-intent and slot-filling also see a benefit from MTL? If an improvement in generalization can be shown, then these techniques will support the creation of robust goal-oriented spoken dialogue systems from minimal initial data sets.

Available Data

In order to learn all three tasks simultaneously a corpus with annotations for all of the tasks is required. There are a number of datasets suitable for training goal-oriented dialogue systems (Serban, Lowe, et al. 2015), however most contain annotations only for only one or two of the tasks we are learning. The Malluba FRAMES (El Asri et al. 2017) corpus is publicly available and contains about 1,300 dialogues collected wizard-of-oz style from 12 participants over a period of 20 days. This corpus was collected to study the role of memory in goal-oriented dialogue systems and therefore contains detailed annotations for both user and system actions. The data is provided in JSON format and each turn is annotated with author (user or system), the text of the author's utterance, and a data frame where dialogue acts for that turn are captured in a name field and arguments field. The name field captures the type of dialogue act and the arguments field captures the slot and value pairs for that utterance. There are twenty types of dialogue acts in total and an average of fifteen turns per dialogue. This corpus contains the annotations required for our study and has sufficiently long dialogues to support the research questions. In addition, since it is in the travel and vacation booking domain it shares a domain with other data sets that are commonly used to train specific dialogue system tasks. For these reasons we chose this corpus as the primary corpus.

We have also evaluated our models on a version of the DARPA COMMUNICATOR data with additional annotations for context and speech acts (Georgila, Lemon, Henderson, and Moore 2009). The original COMMUNICATOR corpus included annotations for user utterances only. Georgila et. al added annotations for system utterances, which is required for this thesis work. The COMMUNICATOR corpus has been used in a significant amount of previous research, particularly for state-tracking. This corpus enables comparisons between our work and previous research on spoken dialogue systems.

The first experiment, experiment zero, will recreate the work of (Hakkani-Tur et al. 2016), who jointly learn user intent and slot-filling using a RNN with LSTM memory cells. For this the ATIS (Price 1990) data set will be used, as

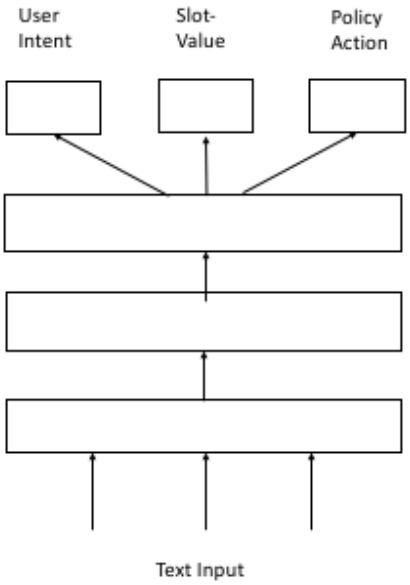


Figure 4: The proposed architecture for this thesis is a ANN with hard parameter sharing for MTL.

in the original work.

Proposed Architecture

Initial experiments will incorporate the work of (Hakkani-Tur et al. 2016), (X. Li et al. 2017), and (Su et al. 2016) and use a Deep Neural Network Architecture. The goal is not to replace components in a spoken dialogue system, but rather to bootstrap the training of a robust dialogue manager. (Caruana 1998) note that MTL models do not have to, and in some instances should not, replace the individual models incorporated into them. Rather, the goal is to learn a representation shared across tasks that improves generalization across all tasks. The hope is that with minimal annotated training data a robust dialogue manager can be built to minimize user frustration when fielded with system users.

User intent is typically modeled as a classification task, where the goal is to classify the user utterance into one of many pre-defined categories. Slot-filling is typically modeled as a sequence labeling problem, where each word is assigned a specific semantic label. We intend to pursue architecture similar to (Hakkani-Tur et al. 2016), which jointly learns user intent and slot-filling using a RNN with LSTM memory cells. The research plan also includes exploration of other networks. As in (Hakkani-Tur et al. 2016) we will use the well known IOB (in-out-begin) tagging format to label slot-tags and fill in slots for the se-

mantic frame. The learning objective is to maximize the conditional probability of the slots and the intent given the word sequence. The weights of the LSTM model are then trained using backpropagation to maximize the conditional likelihood of the training set labels. The predicted tag set is a concatenated set of IOB-format slot tags and intent tags. Recreating this work and comparing our results to (Hakkani-Tur et al. 2016) on the ATIS data set will be our first experiment. This will also provide an opportunity to gain experience with the specific requirements for training in MTL. For example, MTL performs best when each task achieves peak performance at the same time, and a few training runs are usually required to find the appropriate learning rate for each task to achieve this. (Caruana 1998)

Next, we will extend this initial experiment to our proposed model, which learns user-intent classification, slot-filling, and optimal dialogue policies simultaneously. The dialogue response network will share hidden layers with the other tasks and have a task specific Softmax output for predicting the system dialogue act. This prediction will be a multi-class label over the number of dialogue acts. The training objective for each sample is to minimize a joint cross-entropy loss between model action labels and predictions. Figure 4 shows the proposed architecture for this work.

We will follow the procedures used in (Caruana 1998) and compare our MTL networks to networks trained on each task independently. Metrics such as Precision, Recall, F-Measure, and Accuracy will be used to compare performance.

Work Plan

The research will be completed in three experiments. Experiment 0 will first replicate the work of (Hakkani-Tur et al. 2016) and learn only user-intent and slot-filling simultaneously. Next, this architecture will be used to learn slot-filling, user-intent classification, and system action classification simultaneously on the Frames and COMMUNICATOR data. In experiment 1 different neural architectures will be used to learn each of the three tasks individually (see 4) using the FRAMES and COMMUNICATOR corpora. Both experiment 0 and 1 will serve as baselines with which to compare the performance of the multi-task architecture. Experiment 2 will study the multi-task architecture.

Experimentation

In this research slot-filling and user-intent classification are treated as auxiliary tasks for the primary task of dialogue system action classification. Slot-filling and user-intent classification were chosen as auxiliary tasks because both are used by the dialogue manager to decide which action a system should take at each turn in a dialogue, and therefore contain relevant information that can be used to improve the generalization of a system response generator initially trained by supervised-learning. The following sections describe the corpora, data preprocessing, and experimental design and results. All of the data preprocessing and experimentation was done in Python and a package of this software will be delivered as part of this thesis.

Experiments were divided into three groups: baseline experiments 0 and 1, and the final experiment with the thesis architecture, experiment 2. In experiment 0 the experiments described in (Hakkani-Tur et al. 2016) were replicated and then extended to new corpora using the software released by the authors. These extended experiments serve as an additional comparison to the results of the multi-task model from experiment 2. In experiment 1 individual models for each of the three tasks were designed and trained on each corpus. Following the methodology suggested in (Caruana 1998), these models were tuned for each corpus and architecture and serve as a baseline for the multi-task results in experiment 2. Networks were trained and evaluated on three data sets: the Maluuba Frames (El Asri et al. 2017), DARPA COMMUNICATOR (Georgila, Lemon, Henderson, and Moore 2009), (Georgila, Lemon, and Henderson 2005) and ATIS (Price 1990).The Maluuba Frames and DARPA COMMUNICATOR Corpora were used in baseline and multi-task experiments; the ATIS corpus does not contain annotations for system action classification and was therefore only used in experiments 0 and 1.

Table 1 lists the total number of slot types, user-intent labels and system action labels for each corpus. Table 2 shows the training, validation and test set brake down for each corpus. The next sections describe each of these data sets and in more detail.

Corpus	Slot Types	Speech Acts	Sys. Actions	Avg. UL
ATIS	79	17	NA	11.13
Frames	21	45	52	8.05
COMM ATT	34	10	56	1.86
COMM BBN	37	18	48	2.30
COMM CMU	45	32	72	2.58
COMM SRI	37	17	25	2.44

Table 1: The number of slot types, the user speech acts, and system actions labels for each corpus, as well as the average length of user input utterances.

Data and Preprocessing

This work uses the version of the ATIS corpus used in (Hakkani-Tur et al. 2016), the Maluuba Frames corpus (El Asri et al. 2017), and the DARPA COMMUNICATOR corpus with additional information-state annotations (Georgila, Lemon, Henderson, and Moore 2009), (Georgila, Lemon, and Henderson 2005).

ATIS Corpus

The Air Travel Information System (ATIS) Spoken Language Systems pilot corpus (Price 1990) is a collection of spontaneous speech and associated annotations that was designed to measure the progress of spoken language systems. The data was collected wizard-of-oz style, with one user playing a travel planner and two human-wizards simulating the travel system. Travel planers could request information about flights, fares, airlines, cities, airports and ground services and were given a scenario designed to exercise a specific area of the database, like flights or fares; some details, like cities and flight times, were left to the travel planner to fill in during the conversation. The goal was to collect natural, spontaneous speech to fully evaluate the progress of spoken language systems.

This corpus was included with the software released by (Hakkani-Tur et al. 2016), therefore no preprocessing was required. The data is annotated with 79 unique slot types and 17 unique user acts. This corpus was not annotated with system actions and is therefore only used in baselines experiments for slot-filling and user intent classification. The ATIS section of appendix A lists the specific labels used as well as a graph of their distribution in the data.

Corpus	Train Set	Dev Set	Test Set	Input Vocab
ATIS	4478	500	894	900
Frames	6131	1532	1916	3249
COMMUNICATOR (ATT)	3980	442	781	545
COMMUNICATOR (BBN)	3168	351	622	490
COMMUNICATOR (CMU)	2793	310	548	580
COMMUNICATOR (SRI)	4076	452	800	569

Table 2: The number of train, development, and test examples for each corpus.

Maluuba Frames Corpus

The Maluuba Frames (El Asri et al. 2017) corpus was designed to train conversational agents to track multiple aspects of a dialogue across frames and is therefore a richly annotated corpus. This corpus was also collected wizard-of-oz style and system users and wizards communicated through a text chat interface for the purpose of travel booking. Users were given a task, for example:

> Find a vacation between September 1st and September 8th to Havana from Stuttgart for under $700. Dates are not flexible. If not available, then end the conversation.

The goal was to develop complex conversations where users could compare packages and consider multiple options. To preprocess the corpus the original JSON files were parsed to extract the text of the user utterance, the slot labels, the intent (speech act) of the user, and the system response (system dialogue act). The extracted information was then formatted to match the annotations used in (Hakkani-Tur et al. 2016), to facilitate continued experimentation. In this annotation schema the common IOB format was used to annotate slot-tags for each token. Tokens are matched to the slot they fill and multi-token values are labeled with B (begin) and I (inside) to indicate the extent of the tokens that fill that slot. Tokens that are not relevant to the any slot are tagged with O (outside). The next example illustrates an input utterance and its corresponding annotation.

> what about a trip from gotham city to neverland for the same budget

> O O O O O B-or_city I-or_city O B-dst_city O O O O

A beginning of sentence *BOS* and end of sentence *EOS* token was added to each input utterance. To associate a user-intent label and system action label with each utterance, these tags were concatenated and inserted in the last position of the corresponding sequence label. Therefore the full text of the input utterance becomes:

BOS what about a trip from gotham city to neverland for the same budget EOS

The completed corresponding annotation becomes:

O O O O O O B-or_city I-or_city O B-dst_city O O O O inform:sorry#no_result

Individual intent labels and system action labels are separated by the pound sign (#) and the user-intent and system action annotations are separated by a colon.

Not all of the available data fields in the Frames corpus were used. Many of the fields have binary values which make no reference to the corresponding tokens in the input utterance. These binary fields were ignored.

During data preprocessing is was discovered that many of the utterances were labeled with more than one user-intent label and system action label, and frequently some labels were repeated more than once. For example, a user's utterance is often labeled as *inform inform* rather than just *inform*, or the action the system took is labeled as *offer offer offer* if the system provides three offers to the user. These labels are important for the task, however having repeated labels that could appear 1 to n times could require a complicated loss function for statistical learning. One option was to reduce the duplicate labels down to a single label for that class, however a manual review of a subset of the corpus showed a potential correlation between duplicate user-intent labels and the information provided in the corresponding utterance. Removing the duplicate labels risked potentially removing information useful for training. However, if the labels were left as is a system would need to determine the appropriate classes(s) for an utterance as well as the number of times those classes should be used. Since this scenario goes beyond the standard multi-class, multi-label classification problem a choice was made to concatenate the labels into a new class transforming the problem into a multi-class single label problem. This is particularly beneficial because multi-class single label problems are more robust to train. The risks in this type of data transformation is a potential explosion of classes creating a majority of sparse classes. For this corpus the number of user-intent classes with at least 5 examples went from 13 to 45 and system action classes grew from 17 to 52, both of which are reasonable sets of classes for a neural network to learn.

Figure 5 shows the distribution of slot labels in this corpus. Figures A.3a and A.3b in appendix A show the distribution of unique classes created by concatenating user-intent and system action labels. Just as in the ATIS corpus, for all tasks a few classes dominate the data.

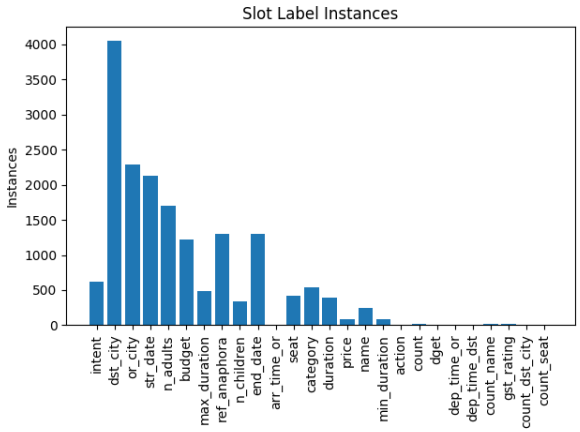


Figure 5: Instances of slot labels in the Frames corpus.

DARPA COMMUNICATOR Corpus

The version of the DARPA COMMUNICATOR corpus used in this work includes the annotations from the original corpus plus additional information state annotations which were automatically added by a system designed by (Georgila, Lemon, Henderson, and Moore 2009). The corpus includes annotations for each system evaluated as part of the COMMUNICATOR program. As in (Henderson, Lemon, and Georgila 2007) only the results from ATT, BBN, CMU and SRI were used. These subsets were chosen to make experimentation on this corpus feasible within the time line of this masters thesis while also facilitating experimentation on corpora with different characteristics than the primary corpus.

Unlike the ATIS and Frames data sets the COMMUNICATOR corpus is a collection of human-computer interactions. Speech was collected from users calling into the COMMUNICATOR travel planning system and the system side of the dialog was annotated. (Georgila, Lemon, Henderson, and Moore 2009) extended these annotations to include the user-intent annotations as well as dialog-level and task-level annotations.

This corpus was preprocessed following the procedure used on the Frames corpus - the text of the user utterance, slots labels, user-intent labels, and system action labels were extracted and the text was paired with the corresponding annotation, which includes slot labels formatted into IOB format. As in the Frames corpus this data was annotated with duplicate class labels. In the majority of examples this seemed to be an artifact of the automatic annotation process, for example when an utterance of a single token, "yes" is annotated with the user-intent label *yes_answer yes_answer yes_answer*. In other instances there was evidence of a correlation between the number of duplicate labels and the information provided in the utterance. For example, duplicates of the *pro-*

vide_info user-intent label correlated with amount of information provided by a user in that utterance.

> BOS i want to travel from atlanta to london england i want to leave on september twenty fourth and return on october first EOS
> provide_info provide_info provide_info provide_info

Just as in the Frames data there was a risk of removing information that could potentially help a learner, particularly for the multi-task models. However, utterances with a single token of either *yes* or *no* labeled with duplicate user-intent label of *yes_answer* or *no_answer* accounted for about 70 % of the errors on the test data in preliminary experiments. The risk to information loss was deemed minimal for these examples, so they were additionally preprocessed to reduce duplicate labels to a single label. The remaining duplicate user-intent and system action labels were concatenated into new class labels.

Figure 6 shows the distribution of slot-labels for each subset of the COMMUNICATOR corpus. Many of the slot-labels are shared by all subsets of the corpus, but there are some classes unique to a specific sub-corpus. Appendix A includes the complete list of specific tags used for each task and Figures [ADD INTENT DIST GRAPH] and [ADD ACTION DIST GRAPH] illustrate the distribution of these tags in each subset of this corpus.

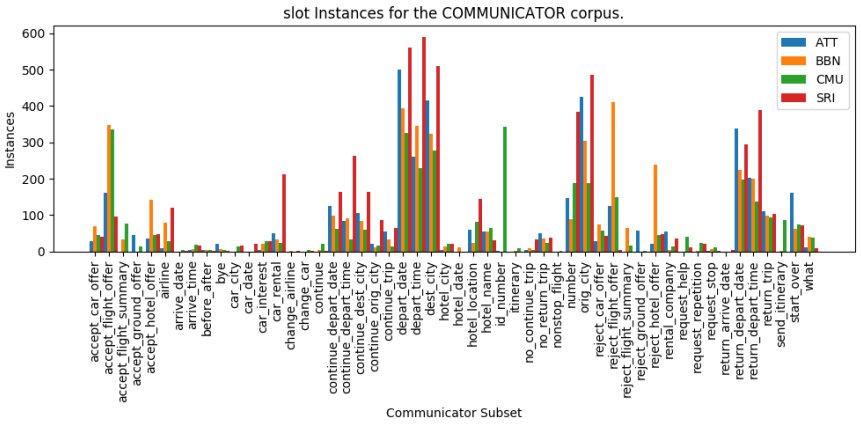


Figure 6: Instances of slot labels in the COMMUNICATOR corpus for the four subsets used in this thesis.

Model	best F	Avg F
ATIS blstm	95.48%	94.70%
Frames	74.26%	73.05%
COMMUNICATOR ATT	48.17%	45.98%
COMMUNICATOR BBN	50.34%	48.77%
COMMUNICATOR CMU	53.96%	52.59%
COMMUNICATOR SRI	59.74%	58.55%

Table 3: The best f-score and average f-score on slot-filling alone for each corpus using the architecture released by (Hakkani-Tur et al. 2016).

Thesis Experiments

Experiment 0

In (Hakkani-Tur et al. 2016) the authors describe a system for jointly learning slot-filling, domain classification, and user intent classification which treats the problem as a sequence labeling task. They use the IOB style annotations for slots and associate the sentence final utterance token with a single label generated by concatenating the associated domain and user-intent labels. The model weights are learned by maximizing the conditional likelihood of the training set labels.

Following the procedures used by the authors, networks were trained with hidden layers of sizes of 50, 100 and 150 and drop-out ratios of 0, 0.25, and 0.50. The learning rate was initialized to .001, batch size was 10, tanh activation was used as the activation function, and weight initialization was done with glorot uniform. All models were trained on a single NVIDIA GeForce GTX 1070.

Table 3 shows the best and average f-score for slot-filling alone on each corpus. The average f-score is calculated over 10 different weight initializations. (Hakkani-Tur et al. 2016) experimented with RNN, LSTM, LSTM-A and BLSTM models, but noted that comparable results were achieved on each and therefore only report results on the BLSTM. Similarly, only the BLSTM is used in the experiments for this thesis. Table 3 includes results on the ATIS corpus as reported in (Hakkani-Tur et al. 2016) and from experiments completed for this thesis on the Maluuba Frames and DARPA COMMUNICATOR corpora. The authors did not specify the method used to calculate Precision, Recall and F-score (e.g., macro or micro averaged), but a weighted macro-averaged score would be the best option, to account for the fact that not all classes are equally likely. The metric scores on the ATIS corpus were reproduced with a weighted macro-averaged precision, recall and f-score, suggesting that was the method used by the authors. All metrics calculated for this research uses weighted macro-averaged Precision, Recall and F1 measure.

The performance of this architecture drops when trained on the Frames and COMMUNICATOR corpora. The COMMUNICATOR corpus has fewer slot-labels compared to the ATIS corpus (51 versus 79), and the utterances in the corpus are, on average, significantly shorter and generally comprise only a brief

Model	Slot F	Avg F	Accuracy
Frames	73.39%	71.84%	57.62%
COMMUNICATOR ATT	48.95%	47.43%	95.65%
COMMUNICATOR BBN	52.25%	50%	94.53%
COMMUNICATOR CMU	65.90%	54.03%	81.70%
COMMUNICATOR SRI	61.31%	59.35%	86.5%

Table 4: Joint slot-filling and user intent classification on the Frames and COMMUNICATOR corpora using the architecture released by (Hakkani-Tur et al. 2016)

answer to a specific question asked by the system. The number of slot-labels in the Frames corpus is about one-third of the number of labels used in the ATIS corpus, and the utterance vocabulary in the Frames corpus is significantly larger- 3249 words versus 900 in the ATIS corpus. These differences in corpus characteristics may contribute to the observed difference in performance.

Additional experiments were done for joint slot-filling and user intent classification, and joint slot-filing, user intent and system action classification. (Hakkani-Tur et al. 2016) use additional Cortana data for their joint modeling and did not release this data. This means their published results are not directly comparable to the results reported here. The best and average slot f-score and the best user intent accuracy on the test data are recorded in Table 4. On the Frames data the performance of the slot-filling task is reduced with joint-modeling, however performance on all subsets of the COMMUNICATOR corpus is improved.

Model	SLot F1	Intent Acc	Action Acc
Frames	71.40%	49.62%	38.99%
COMMUNICATOR ATT	46.05%	94.75%	55.95%
COMMUNICATOR BBN	50%	92.60%	44.37%
COMMUNICATOR CMU	53.07%	79.20%	42.52%
COMMUNICATOR SRI	59.13%	85.13%	62.25%

Table 5: Joint slot-filling, user intent classification and system action classification on the Frames and COMMUNICATOR corpora using the architecture released by (Hakkani-Tur et al. 2016).

Table 5 shows the results for jointly learning all three tasks. The system is able to perform all three tasks, but there is a drop in performance for both auxiliary tasks. This phenomenon was also observed by the authors in the original publication. This model also only predicts a subset of the system action tags annotated in the training data. The precision, recall and f-score metrics for the predicted system action classes are in Figure 7.

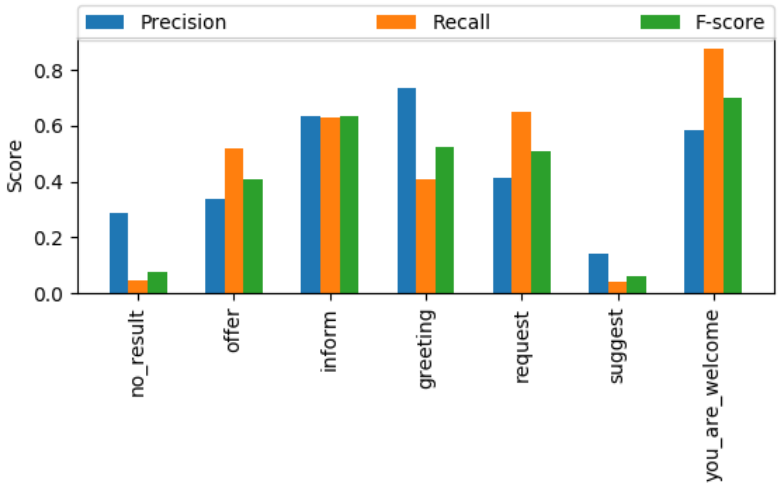


Figure 7: Performance on system action labels in the Frames corpus using the architecture released by (Hakkani-Tur et al. 2016).

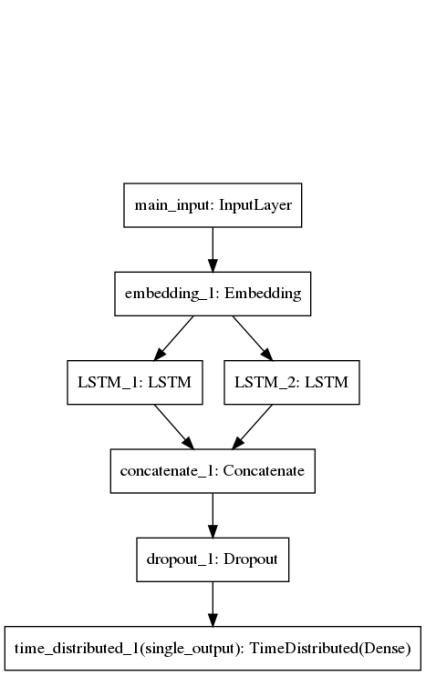
Experiment 1

In this experiment single-task models were created to perform slot-filling, user-intent classification, and system action classification individually. These models serve as baselines for the multi-task learning results in experiment 2. All of the baseline models were optimized for each corpus and task, following the suggestions in (Caruana 1998).

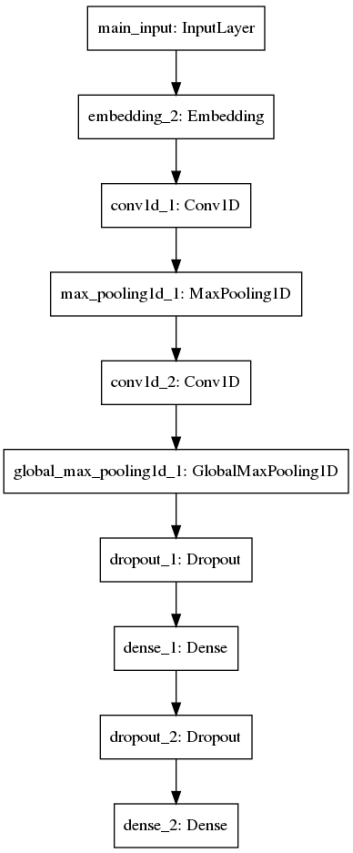
Models

There are many architectures available for modeling natural language problems with Artificial Neural Networks. Long Short-Term Memory Units (LSTMs), B-directional LSTMs, Sequence-to-Sequence models, and Convolutional Neural Networks have all been used extensively to model NLP problems and have been thoroughly tested and shown to be useful for a number of NLP tasks. Therefore these are the architectures of focus in this thesis. All models for experiment 1 and 2 were built using the Keras functional API and will be delivered as part of this thesis.

LSTM and BLSTM baseline models are composed of an input layer, an embedding layer, two hidden layers and a softmax output layer. In the BLSTM



(a) BLSTM model diagram



(b) CNN model diagram

Figure 8: Model diagrams for single-task baseline models.

model the second hidden layer processes the input data backwards, starting from the last word in a sentence. Figure 8 shows the architecture diagram for the BLSTM model. The LSTM model is similar to the BLSTM model and is therefore not shown separately. The SEQ2SEQ model uses a single encoder and a single decoder. The CNN model was based on an example included in the Keras Documentation (this Keras example) designed for a sentence classification task. That CNN model was modified and consists of an input layer, word-embedding layer, a 1D convolutional layer followed by a MaxPooling layer, a second 1D convolutional layer followed by a Global Max Pooling layer, a fully connected layer, then a final softmax layer. The diagram for the CNN model can be found in Figure 8.

Results

Optimal configurations were found by hyper-parameter search. Glove (Glove) word embeddings were used as pre-trained word embeddings. The Adam optimizer was used with a learning rate of 0.001 and a decay of 0. Initialization was done with glorot uniform and tanh was used as the activation function for the LSTMs. Early stopping was used to prevent over-fitting. All networks were trained on batch sizes of 15, 25, and 50 and drop-out ratios of 0, 0.25, and 0.5 on the fully-connected layers. This resulted in twenty seven experiments for each corpus for the LSTM and BLSTM models. For the CNN models, experiments were run with filters of size 100, 200 and 300, kernel size of 3, max pooling of 3, and 100, 200 or 300 units in the fully connected layer. This resulted in 80 experiments per corpus. All models were trained on a single NVIDIA GeForce GTX 1070.

Tables 6, 7, and 8 show the performance of the models on each corpus. The metrics reflect the highest score on the test set for each corpus and task using the architecture of the experiment. The results on these models are similar to the results from experiment 0 - the highest performance is achieved on the ATIS corpus. As in experiment 0 the model performance is lower on the COMMUNICATOR corpus for slot-filling than on the Frames corpus, but the performance on user-intent classification is higher in the COMMUNICATOR corpus.

Slot-Filling

The highest f-score achieved for each model on the slot-filling task can be found in Table 6. For the SRI subset the slot f-scores are 7 points higher than the score reported in (Henderson, Lemon, and Georgila 2007), even though these experiments use all of the slots annotated in the corpus while (Henderson, Lemon, and Georgila 2007) uses only four slots: origin city, destination city, departure date, and departure time.

Results for the LSTM and BLSTM networks are comparable, therefore the focus of analysis is only on the BLSTM network. Analysis of the Frames corpus shows the BLSTM network performs well on filling the *or_city*, *dest_city*, *str_date* and *end_date* slots and slightly lower on the *budget* slot. For the *name* slot (used to label a hotel name or airline name, etc.) and the *ref_anaphora* slots performance is worse. Comparing the model outputs to the annotated ground truth shows that in some instances the model just completely missed the tokens relevant to the name slot. In the examples below the first string is the input utterance, the second line is the ground-truth annotation, and the third line is the model output.

BOS ok get us that market palace then EOS
O O O O O B-name I-name O O
O O O O O O O O O

BOS i will go with the stardust hotel in valencia EOS BOS can you confirm the booking EOS
O O O O O O B-name I-name O B-dst_city O O O O O O O O

Model	ATIS	Frames	ATT	BBN	CMU	SRI
LSTM	93.29%	72.31%	47.14%	46.91%	54.99%	60.98%
BLSTM	93.73%	72.56%	47.88%	48.82%	55.41%	60.93%
SEQ2SEQ	73.52%	36.08%	46.91%	38.33%	42.28%	47.24%

Table 6: F1 performance of the single task models for slot-filling.

O O O O O O O O O B-dst_city O O O O O O O O

In other cases it labels some but not all of the associated tokens:

BOS my preference would be the golden excalibur hotel for $ 57772 usd EOS
O O O O O O B-name I-name I-name O O O O O
O O O O O O O I-name I-name O O O O O

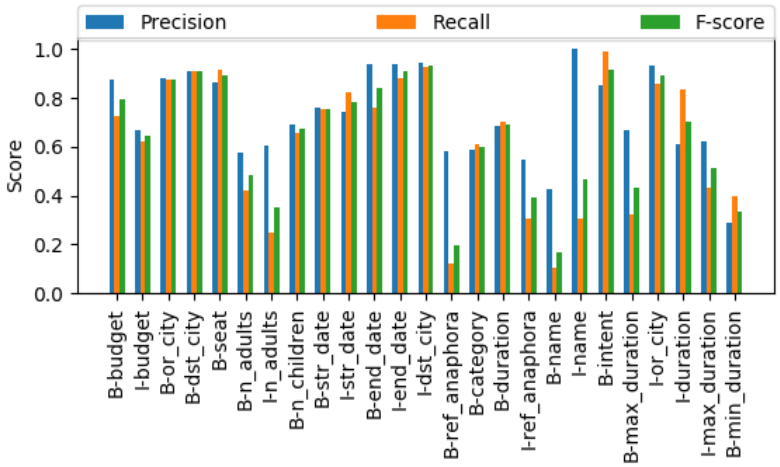


Figure 9: Precision, Recall and F-score per slot-tag in the FRAMES corpus using a BLSTM model.

The model misses the first token associated with the *name* slot, but is more consistent in identifying the sequential tokens in a multi-token hotel name. There are less than 500 examples of this slot type in the whole corpus (Table 5) but the models performance on this slot is worse than for the *category*

slot, which has a similar number of examples in the corpus. Hotel names can vary widely and the model misses words when hotel names are three words or more. This is also evidence that the model has not learned the relationship between B and I tags.

The tokens associated with the *n_adults* and *n_children* slots vary widely in this corpus, and the system performs better on the *n_children* tag, possibly because the word children often comes right after the number of traveling children.

In a number of cases the model completely misses the tokens associated with this slot:

BOS so myself and my child EOS
O O B-n_adults O B-n_children I-n_children O
O O O O O O O
BOS i go by myself EOS
O O O O B-n_adults O
O O O O O

In other cases the model incorrectly associates a number between one and five with the *category* tag even though the word adults comes right after it:

BOS i have already told you 5 adults and 14 kids EOS
O O O O O O B-n_adults O O B-n_children O O
O O O O O O B-category O O B-n_children O O

In other examples the system found relevant information that wasn't explicitly included in the ground truth annotations:

BOS we are 8 in total EOS BOS 6 adults and 2 children EOS
BOS we can go whenever lets see what you have to offer EOS
O O O O O O O O B-n_adults O O B-n_children O O O O O O
O O O O O O O O O O
O O O B-n_adults O O O O B-n_adults O O B-n_children O O
O O O O O O O O O O O O O O

BOS one ticket from atlanta to tofino please EOS
O B-n_adults O O B-or_city O B-dst_city O O
O O O O B-or_city O B-dst_city O O

In this corpus there are a diverse set of phrases used to indicate the number of adults and children traveling. Perhaps additional linguistic information in the form of features would improve the model. Reformatting numbers to replace every digit with a NUM token could also improve results. This would significantly reduce the vocabulary for numbers while still providing information on the structure of numbers and how they are used.

Figure 10 shows the results of the slot-filling task on a BLSTM trained on the SRI subset of the DARPA COMMUNICATOR corpus. Figure 10 only shows

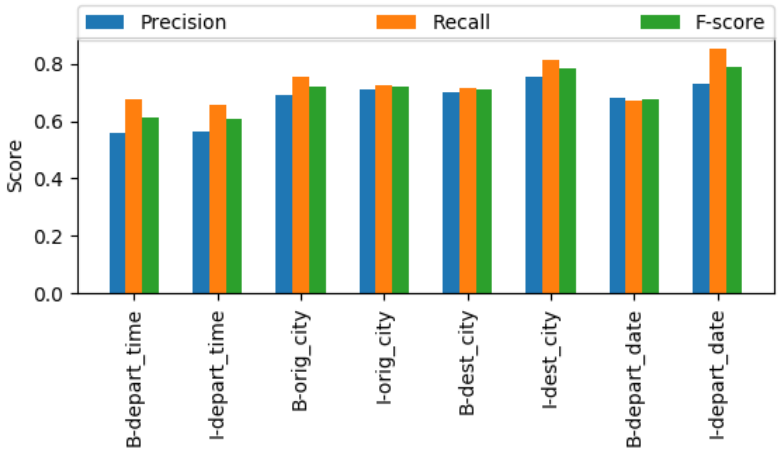


Figure 10: Precision, Recall and F-score for a subset of slot-tags in the COMM SRI corpus using a BLSTM model.

results for the subset of the slots that were also analyzed in Henderson, Lemon, and Georgila 2007.

Reviewing the model outputs for the *depart_time* slot shows that the model correctly identifies tokens associated with dates, but does not consistently label the tokens with the correct tag. Perhaps features to reinforce that I (inside) tags should be the same as their B (begin) counterparts would improve results.

BOS august twelve EOS
O B-depart_date I-depart_date O
O B-return_depart_date I-depart_date O

BOS october eighth EOS
O B-depart_date I-depart_date O
O B-continue_depart_date I-continue_depart_date O

BOS one p. m. EOS
O B-depart_time I-depart_time I-depart_time O
O B-depart_time I-return_depart_time I-depart_time O

Reviewing results for the *accept_flight_offer* tag, which is one of the tags the model performs the lowest on (see Figure C.1), shows that often the model uses the wrong tag.

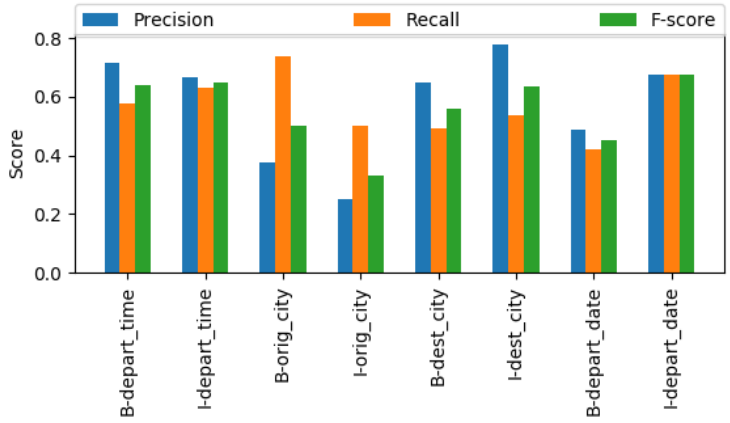


Figure 11: Slot-tag precision, recall, and f-score on ATT COMMUNICATOR data.

BOS the second EOS
O O B-accept_flight_offer O
O O B-number O

BOS the first one EOS O O B-accept_flight_offer I-accept_flight_offer O O O B-number I-accept_flight_offer O

Again we see the model confusing numbers with other tags.

H. Chen et al. 2017 cite recent work where neural generative models based on the encoder-decoder architecture perform well on dialogue system tasks. However, in the experiments for this thesis the performance of the sequence-to-sequence models on each corpus was much lower than on the LSTM and BLSTM models. Manual review of the errors showed that often the model did not predict the correct sequence length and therefore did not predict the correct slot labels. It's likely further experimentation could optimize a seq2seq model for these tasks, however since significant results were achieved on basic LSTM and BLSTM models it was decided not to pursue encoder-decoder model for any other experiments.

Model	ATIS	Frames	ATT	BBN	CMU	SRI
LSTM	96.86%	59.60%	95%	93.09%	80.66%	86.63%
BLSTM	96.75%	59.60%	94.88%	92.93%	79.20%	86.38%
CNN	96.75%	59.34%	60.70%	93.35%	79.38%	86.25%

Table 7: Accuracy of the single task models for user intent classification.

Table 7 shows the results of the architectures for the user-intent classification task, where the scores are the best accuracy on the test set. The performance of the CNN model is comparable to the LSTM and BLSTM model for most data sets and they train much faster. For example, the BLSTM model with a batch-size of 15, 50 units in each hidden layer, and an embedding dimension of 200 trains in 9 minutes and 30 seconds while the CNN model with a batch-size of 15, 100 filters and a fully-connected layer of 100 nodes trains in 2 minutes and 76 seconds on the same data set. These scores are comparable to the scores achieved on the joint slot-filling and user-intent classification task using the architecture released by (Hakkani-Tur et al. 2016).

Analysis of the FRAMES data on the BLSTM model shows that for some samples the system simply selects the wrong class. As before, in the examples below the first line is the input text, the second line the target intent labels and the third line the model output.

BOS definitely economy im a poor intern EOS
switch_frame
inform

BOS is the six day package at a better hotel EOS
request_compare
switch_frame

Other times the model seems to select a related but still incorrect class:

BOS okay then well go with that please EOS
inform
switch_frame#inform

BOS is it available for the same dates EOS BOS also how much would that one be EOS
switch_frame#inform#request
switch_frame#request

BOS hello i represent a group of people who want to get out of munich EOS BOS our group comprises 4 adults and 6 children EOS
inform
inform#inform

BOS yeah lets do that with business seats EOS
affirm#inform
switch_frame#inform

In other examples the system used a label that may have been missed in the original annotation:

BOS perfect EOS BOS book EOS BOS thanks EOS
inform
inform#thankyou

An interesting observation is the performance of the CNN model on the ATT subset of the COMMUNICATOR corpus. The performance of the CNN model is comparable to the performance of the LSTM and BLSTM models for all data sets, except for the ATT subset. A review of the mislabeled utterances shows that the network was converging, however it was converging to a bad local minimum which caused the system to predict the most common token, *provide_info*, the majority of the time. This is a common failure mode in neural networks, and it's particularly interesting that this was observed for the ATT subset of the COMMUNICATOR corpus, but not for the other subsets which share characteristics in terms of vocabulary size and average utterance length.

The CNN model is a small and constrained CNN with single convolutional layers followed by pooling layers. Perhaps a more highly engineered CNN model, for example a version of Google's Inception model (see here for more details), would perform better on the COMMUNICATOR ATT corpus subset. Since this drop in performance was only observed for one subset of the COMMUNICATOR corpus it was decided not to pursue additional individual optimization of the CNN model for that corpus subset.

Review of the results of the BLSTM model on the SRI test data shows that sometimes the model predicts the incorrect label:

BOS paris france EOS
provide_info
NONE

On other examples it correctly predicts a related but incorrect label. Perhaps jointly training slot-filling and user-intent classification will help improve results for cases like this.

BOS i would like an intermediate five car EOS
provide_info#provide_info
provide_info

Table 8 lists the accuracy scores for baseline models on system action classification. Review of the results of the FRAMES and COMMUNICATOR SRI models on the test data shows that most errors were due to the system predicting the incorrect label. Pre-training a dialogue policy from corpora is a difficult

Model	Frames	ATT	BBN	CMU	SRI
LSTM	43.48%	54.93%	44.21%	40.15%	62.38%
BLSTM	43.11%	55.31%	44.86%	40.15%	61.75%
CNN	42.59%	48.27%	44.85%	39.42%	62%

Table 8: Accuracy of the single task models for system action classification.

task, particularly because an optimal policy may not exist in a corpus. Current research frequently uses supervised pre-training to initialize policy networks, but the majority of publications only include metrics on the performance of the model after additional online training with reinforcement learning. (Bing and Lane 2017) cite the per-response accuracy comparing a number of models, including Memory Networks and Sequence-to-Sequence models, for learning system responses from corpora. Those models were trained using an annotated version of the DSTC corpus created by (Bing and Lane 2017) so the baseline results aren't directly comparable; however, the results for this task fall into the range of the results from the models in that publication.

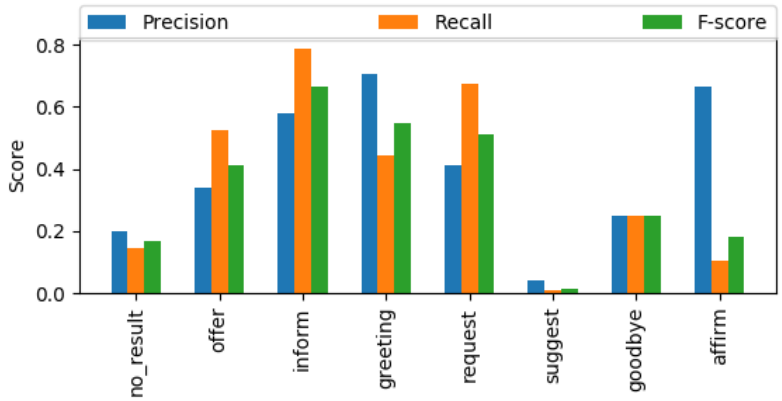


Figure 12: Performance of system action classification on the Frames data.

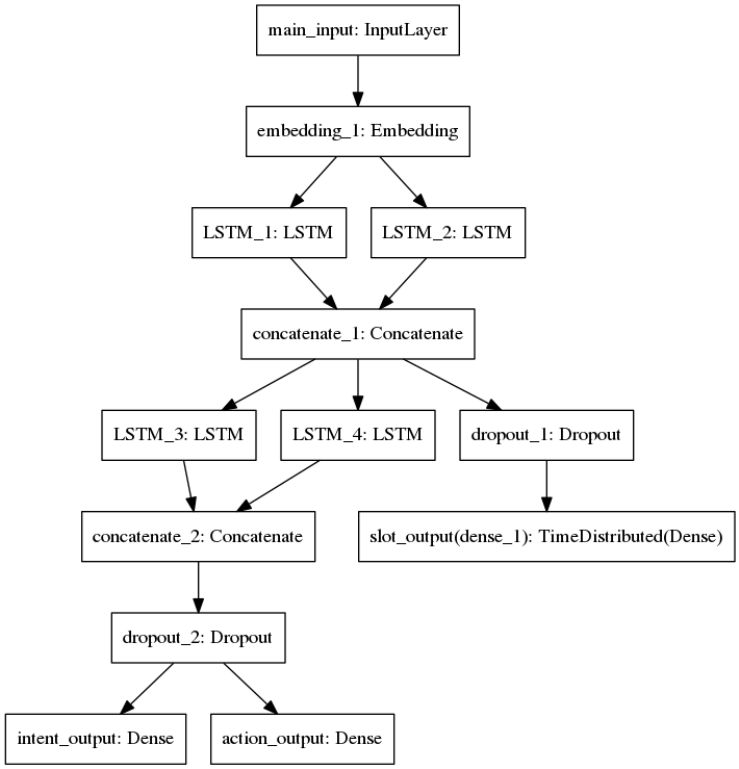


Figure 13: Model diagram for multi-task model BLSTM1.

Experiment 2

In this experiment several multitask neural architectures were designed to treat slot-filling and user-intent classification as auxiliary tasks to supervised pre-training of dialogue policies. Results are compared to the single-task baseline models described in the previous sections. The next section describes each of the models in more detail. The results section includes metrics and analysis on test data.

Models

All models were designed with BLSTMs, CNNs, or a combination of the two. These networks are the focus in this thesis because these models are well known, well tested, and have detailed support in most machine learning frameworks. In the majority of baseline experiments LSTM and BLSTM performance was equivalent or favored the BLSTM model, therefore continued experimentation focused on BLSTM networks.

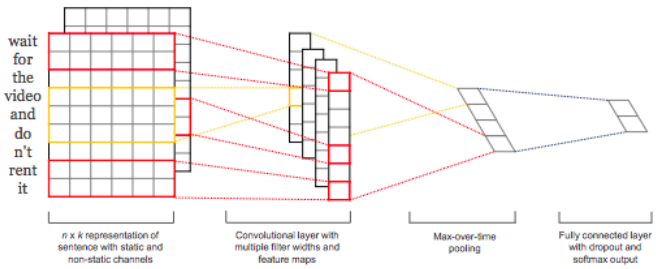


Figure 14: CNN1 model architecture. This diagram is borrowed from (Yoon 2014).

BLTM Models

Three BLSTM models were tested. The diagram for model BLSTM1 is shown in Figure 13. In this model the input is passed to a trainable embedding layer, which is then connected to the first BLSTM layer. This layer is given an auxiliary output which is used to train slot-filling. The output of the first BLSTM is connected to a second BLSTM layer which is given two separate output layers, one for the second auxiliary task, user-intent classification, and another for the primary task of system action classification. In this model the training loss for the slot filling task is backpropagated to update the first BLSTM layer. Loss for the second auxiliary task and primary task update all layers. To explore the role of each auxiliary task ablation testing was performed. The BLSTM1 model was trained on all three tasks simultaneously (BLSTM1a), on slot-filling and the primary task alone (BLSTM1b), and on user-intent classification and the primary task alone (BLSTM1c). The second model, BLSTM2, included a skip connection from the embedding layer to the second BLSTM layer. The third model, BLSTM3, replaced the first BLSTM layer with a slot-filling BLSTM trained on the same data. During multi-task training on BLSTM3 model only user-intent classification and system action classification were trained (the slot-filling output layer was ignored). The goal was to explore the possible benefit of transfer learning from a previously trained model. The Keras model diagrams for BLSTM2 and BLSTM3 models can be found in appendix B.

CNN Models

The first CNN model, CNN1, uses a design inspired by (Yoon 2014). A trainable embedding layer is connected to a convolutional layer that produces multiple feature maps from multiple filters with width 2,3,5, and 7. Max-pooling over time is performed, and this is connected to a fully connected layer. See Figure 14. Filter widths, the number of feature maps, and the number of nodes in the fully connected layer were chosen based on the suggestions of (Ye Zhang and Wallace 2015). Early experiments on the BLSTM models showed a potential benefit to using user-intent classification alone as the auxiliary task. Since this model is

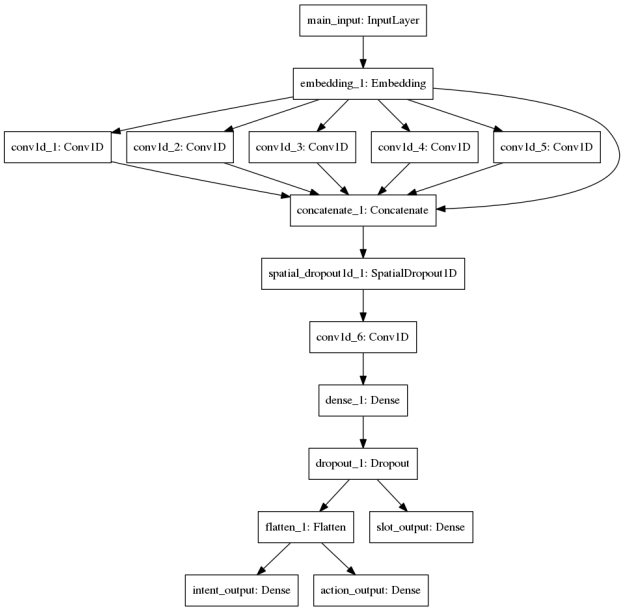


Figure 15: Model architecture for CNN INCEP2 model.

inspired by a model designed for sentence classification, these experiments used only user-intent classification as the auxiliary task (ignoring slot-filling).

Additional experiments were done with models inspired by Google's Inception architecture (Szegedy et al. 2014). The primary goal of the Inception architecture is to find out how an optimal local sparse structure in a convolutional vision network can be approximated and covered by readily available dense components (Szegedy et al. 2014). As neural networks become deeper and wider, their ability to learn from data improves. Unfortunately the training resources required grow as a network becomes larger, increasing training time. Sparse networks seem like an obvious direction to battle the increase in training resources, but sparse networks also present challenges during training. The inception architecture was an attempt to discover a dense architecture that approximates this sparse structure, without the training difficulties. The medium of focus was vision models, but many vision models have translated well to natural language process (for example, CNNs), so the decision was made to experiment with them here. See Figure 15 for the architecture diagram for CNN INCEP1 model, which is composed of a single "inception tower": a convolutional layer with multiple filters of different filter widths connected to a spatial drop-out layer, connected to a single convolutional layer, connected to a fully-connected layer. The INCEP2 model is composed of three "inception towers".

The final multi-task model is a hybrid CNN + BLSTM model. In this model the embedding layer is connected to a convolutional layer layer with three filters

Model	Frames	ATT	BBN	CMU	SRI
BLSTM1a		P			
BLSTM1b					
BLSTM1c	P+R ✓	P			
BLSTM2	P+R ✓			P	
BLSTM3	P ✓				
BLSTM3b	NA				
CNN1		R ✓			P
CNN INCEP1		R ✓		P	
CNN INCEP2		R ✓		P ✓	
CNN+BLSTM	P ✓	R ✓		P ✓	

Table 9: The flag indicates whether a particular models f-score improvement over the baseline was statistically significant.

of different widths. The output of this layer is concatenated to the embeddings and connected to the BLSTM1 model. The goal was to explore the possibility of extracting features with a CNN layer, that could then be used to improve performance on the BLSTM models. This model combined components of CNN and BLSTM models described in the previous sections. The complete model diagram can be found in Appendix B.

Results

Optimal configurations were found by hyper-parameter search. All networks were trained with batch sizes of 15, 25, 50 and 100 and drop-out ratios of 0, 0.25, and 0.5 on the fully-connected layers. Glove (Glove) word embeddings were used as pre-trained word embeddings. The Adam optimizer was used with learning rate, decay, and other parameters set to the defaults in Keras. All weights were initialized with glorot uniform. The LSTM layers used tanh as the activation function. During training the validation loss was monitored and early stopping was used to prevent over-fitting. All models were trained on a single NVIDIA GeForce GTX 1070.

For each corpus many of the multi-task models achieved a higher metric score than the baseline on the test data, however significance testing showed not all of these improvements were statistically significant. Table 9 provides a summary of the results on the multi-task models. For each model-corpus combination capital P or R indicates a statistically significant improvement in precision or recall over the corresponding baseline model on the held-out test data; a check mark indicates a statistically significant improvement in either f-score or accuracy. There is no single model that improves performance for every corpus, but the CNN+BLSTM model does improve for three our of the five corpora. The sparsity of results as reflected in 9 suggests that more data would be required to make stronger conclusions.

Tables 10, 11, 12, 13, and 14 give the accuracy, precision, recall and f-score

Data	Experiment	Acc	Precision	Recall	F-score
Frames	*Exp 0*	38.99%	33.42%	39.24	34.77%
	Baseline BLSTM	43.11%	31.84%	44.10%	36.26%
	BLSTM1a MT	**44.21%**	31.03%	**45.22%**	36.08%
	BLSTM1b MT	**43.58%**	31.53%	**44.58%**	35.88%
	BLSTM1c MT	**44.94%***	**34.39%***	**45.97%***	**37.93%***
	BLSTM2 Skip	**44.78%***	**38.63%***	**45.81%***	**36.29%**
	BLSTM3 PRE	**43.79%**	**34.56%***	**44.79%**	**37.49%***
	Baseline CNN	42.59%	30.70%	43.57%	35.37%
	CNN1	39.46%	**31.62%**	40.36%	34.59%
	CNN INCEP1	36.48%	30.28%	37.32%	32.78%
	CNN INCEP2	36.43%	30.43%	37.26%	32.80%
	CNN+BLSTM	**42.75%**	**33.04%***	**43.73%**	**36.82%***

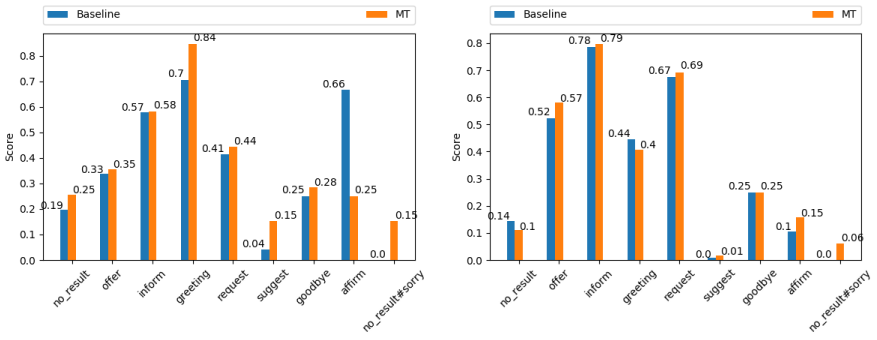
Table 10: FRAMES performance on multitask ANN models.

metrics for the held-out test data on each multi-task model, for each corpus. The numbers in bold indicate where a multi-task model achieved a higher metric than the baseline model; numbers with an asterisk next to them indicate results that are statistically significant. Significance testing was done with randomized approximation (Yeh 2000). The following sections describe the analysis of each experiment in more detail.

MALUUBA FRAMES Corpus

The Frames data set is the largest of the corpora tested and has utterances, on average, that are much longer than those in the COMMUNICATOR corpus (average utterance length about 8 tokens versus 2). Both BLSTM and CNN multi-task models achieved statistically significant results over the corresponding baseline models on the test data (Table 10). The multi-task model BLSTM1c has about four times the number of trainable parameters than the baseline - 451,601 trainable parameters v.s. 105,854 - but is the smallest of the models that achieves statistically significant improvement over the baseline (456,839 trainable parameters for BLSTM2 and 501,901 for BLSTM3). The CNN+BLSTM model achieves significant improvements in f-score over the CNN baseline, but it's performance is still lower than all of the other multi-task BLSTM models, and is the largest of all the well performing models with 964,128 trainable parameters.

To explore model performance the precision, recall and f-score metrics per-tag were calculated. Precision and recall results on a set of the most relevant tags can be found in figures 16a and 16b. For actions *offer*, *no_result#sorry*, *request*, and *suggest* the multi-task model achieved an improvement in both precision and recall. For the *affirm* token the precision dropped by more than half, but the recall increased by half a percent. The multi-task model also did better on the *no_result#sorry* tag, improving over the baseline which did not predict that tag.



(a) Per-tag precision.

(b) Per-tag recall.

Figure 16: Per-tag metrics for a selection of tags for BLSTM1c model on Frames data.

Looking more closely at the *affirm* action the multi-task model correctly labels an instance the baseline mislabels:

> and this leaves on fri aug 26 yes
> ground truth: affirm
> baseline: no_result
> multi-task: affirm

However, the multi-task model produced many more false positives, mislabeling offer and inform actions. For example:

> is that leaving on exactly on tuesday september 6
> ground truth: inform
> baseline: inform
> multi-task: affirm
>
> is that all youve got
> ground truth: offer
> baseline: inform
> multi-task: affirm

Looking at the *greeting* action, the multi-task model correctly labeled a few instances of this label the baseline missed. The baseline model also mislabeled the *request* tag with *greeting*, for example:

> hi im looking to book my honeymoon
> ground truth: request
> baseline: greeting
> multi-task: request

Data	Experiment	Acc	Precision	Recall	F-score
ATT	*Exp 0*	55.95%	50.97%	59.46%	52.55%
	Baseline BLSTM	55.31%	51.06%	58.76%	52.56%
	BLSTM1a MT	54.80%	**52.47%***	58.23%	52.52%
	BLSTM1b MT	54.80%	**54.40%**	58.23%	52.31%
	BLSTM1c MT	**55.95%**	**52.41%***	**59.46%**	**53.12%**
	BLSTM2 Skip	**55.95%**	47.46%	**59.46%**	51.74%
	BLSTM3a PRE	54.55%	51.40%	57.96%	52.13%
	BLSTM3b PRE	55.06%	**55.11%**	58.50%	52.29%
	Baseline CNN	48.27%	54.73%	51.29%	46.93%
	CNN1	**55.31%***	50.80%	**58.75%***	**51.90%***
	CNN INCEP1	**54.80%***	51.58%	**58.23%***	**52.62%***
	CNN INCEP2	**55.19%***	53.75%	**58.64%***	**53.28%***
	CNN+BLSTM	**55.31%***	47.33%	**58.78%***	**51.38%***

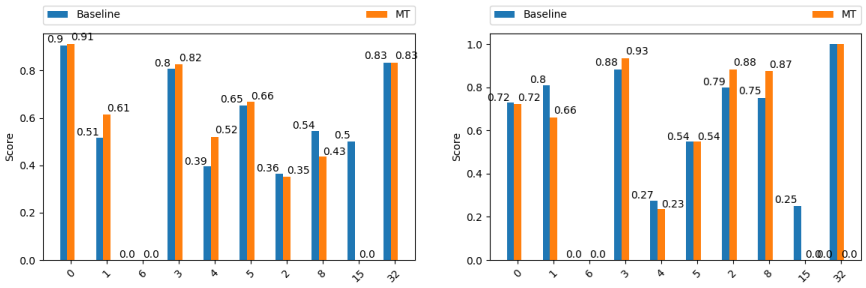
Table 11: COMM ATT performance on multitask ANN models.

DARPA COMMUNICATOR Corpus

The system action label names in the COMMUNICATOR corpus are much longer than the names in the Frames corpus, and therefore can't fit compactly into a figure. Therefore all of the labels are referenced by number in this section. The accompanying text includes the name of the tag referenced by that number.

Table 11 shows the metrics for each multi-task model on the ATT subset of the communicator corpus. The BLSTM1a and BLSTM1c models achieve a statistically significant improvement in precision over the baseline, but this does not lead to significant improvement in f-score. Models BLSTM1b and BLSTM3b increase the precision over the baseline by a large number, but significance testing showed these results were not significant. Further analysis revealed that the improvement in precision was most likely due to a large improvement in one tag, which is also one of the most frequent tags: *unmatched*. BLSTM1b and BLSTM3b models correctly label half of the instances of the *unmatched* tags, while model BLSTM1c mislabels all instances (Figure C.2). Since this is the 6th most frequent token in the data set, and the metrics are calculated based on the weighted average, this likely caused a large increase in precision. When calculating the performance across classes the BLSTM1c model is the only model that improves significantly over the baseline.

All of the multi-task CNN models achieve statistically significant improvement over the CNN baseline, but most achieve an f-score similar to or no better than the baseline BLSTM model. The INCEP2 model achieves the highest f-score on this data, but it is also the largest model with 4,101,073 trainable parameters. The baseline BLSTM has 106,258 trainable parameters, BLSTM1c has 107,470, and the CNN+BLSTM model has 762,243. Figure 17 compares the per-tag precision and recall for a subset of tags on the baseline BLSTM model and INCEP2 model. The largest improvements are achieved on tags 1 (*implicit_confirm#request_info*) and 3 (*implicit_confirm#unmatched*. Tag 15



(a) Per-tag precision.

(b) Per-tag recall.

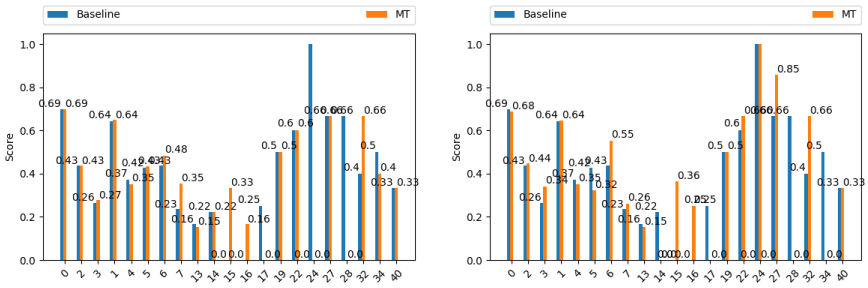
Figure 17: Per-tag metrics for a selection of tags for INCEP2 model on ATT data. FIX DATA LABELS.

(*status_report#offer#present_info#offer*) is correctly used by the baseline model but missed by the multi-task model.

Data	Exp	Acc	Precision	Recall	F-score
BBN	*Exp 0*	44.37%	48.77%	47.66%	40.59%
	Baseline BLSTM	43.16%	49.29%	48.35%	43.52%
	BLSTM1a MT	44.86%	48.98%	48.35%	**43.71%**
	BLSTM1b MT	44.05%	46.98%	47.47%	42.52%
	BLSTM1c MT	44.53%	48.14%	48.01%	42.96%
	BLSTM2 Skip EMB	44.85%	46.96%	48.35%	**43.56%**
	BLSTM3a SLOT PRE	44.21%	46.12%	47.66%	42.27%
	BLSTM3b SLOT PRE	44.21%	49.24%	47.66%	43.19%
	Baseline CNN	44.85%	48.68%	48.35%	43.93%
	CNN1	**45.82%**	47.07%	**49.39%**	43.93%
	CNN INCEP1	44.21%	47.85%	47.66%	43.29%
	CNN INCEP2	44.53%	**49.14%**	48%	43.74%
	CNN+BLSTM	**45.34%**	48.11%	**48.87%**	43.73%

Table 12: COMM BBN performance on multitask ANN models.

On the BBN subset of the COMMUNICATOR corpus none of the models achieved a statistically significant improvement over the baseline models on any metric. The model that achieves the highest f-score on the data is the baseline CNN model, which has 520,850 trainable parameters. The best performing multi-task BLSTM and CNN models, BLSTM1a and CNN1, do not improve over the baseline models and have at least twice as many trainable parameters (1,000,627 and 1,832,246, respectively). Figure 18a compares the per-tag f-score for a subset of the system action labels on the BLSTM baseline and multi-task model BLSTM1a (which uses both auxiliary tasks). The per-tag f-scores are very close for the majority of the tags, however there are a few tags the baseline model uses correctly and the multi-task model misses, and vice versa. The baseline model uses tags 17 (*apology#instruction#request_info*), 24 (*status_report#request_info*), and 28 (*implicit_confirm#apology#explicit_confirm*) which the multi-task model does not, and the multi-task model uses 15 (*apology#instruction#present_info#offer*) and 16 (*implicit_confirm#implicit_confirm#request_info*) which the baseline model does not. Similar results are observed in figure 18b. This suggests that there isn't enough data and therefore the models are learning different things. The BBN subset is the second smallest data set with less than half the amount of training examples as the Frames corpus.



(a) BLSTM1a model f-score comparison.

(b) CNN1 model f-score comparison.

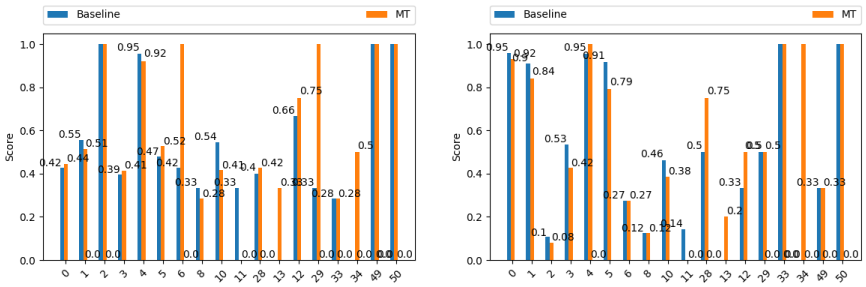
Figure 18: Per-tag metrics for a selection of tags for BLSTM1a and CNN1 models on BBN data. REARANGE GRAPH.

Data	Exp	Acc	Prec	Recall	F-score
CMU	*Exp 0*	42.52%	34.68%	49.89%	36.11%
	Baseline BLSTM	37.02%	39.14%	47.93%	37.06%
	BLSTM1a MT	40.15%	**39.70%**	47.93%	**37.49%**
	BLSTM1b MT	**40.33%**	**39.70%**	**48.15%**	**38.06%**
	BLSTM1c MT	38.69%	37.56%	46.19%	36.58%
	BLSTM2 Skip EMB	39.05%	**41.51%***	46.62%	**37.55%**
	BLSTM3a SLOT PRE	39.05%	39.56%	46.62%	**37.34%**
	BLSTM3b SLOT PRE	**40.33%**	**39.09%**	**48.15%**	**38.01%**
	Baseline CNN	39.42%	36.43%	47.06%	36.44%
	CNN1	38.87%	**37.95%**	46.41%	**36.73%**
	CNN INCEP1	39.05%	**41.39%***	46.62%	**37.59%**
	CNN INCEP2	**39.96%**	**41.01%***	**47.71%**	**38.47%***
	CNN+BLSTM	**40.88%**	**40.80%***	**48.80%**	**38.53%***

Table 13: COMM CMU performance on multitask ANN models.

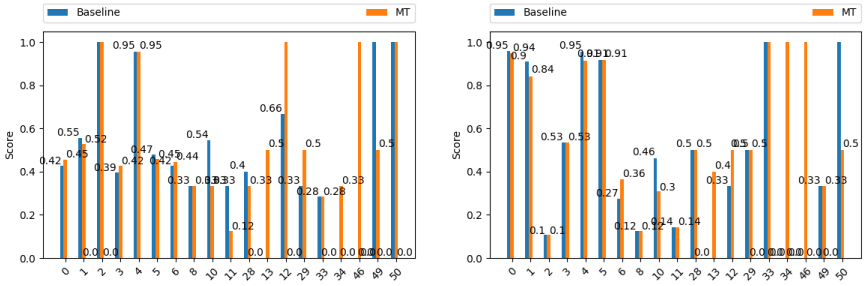
Table 13 shows the results for the multi-task models on the CMU subset of the COMMUNICATOR corpus. None of the BLSTM multi-task models achieve a statistically significant improvement in accuracy or f-score over the baseline model, however the BLSTM2 model does achieve a significant improvement in precision. Figure 19 shows the comparison of precision between the baseline BLSTM model and the BLSTM2 model for a subset of the action labels. Performance is close for most of the labels, however for tag 11 (implicit_confirm#status_report#implicit_confirm#present_info#offer) the baseline model uses the tag correctly, while the multi-task model misses it completely. On tags 6 (*apology#request_info*) and 29 (*instruction#instruction#opening_closing#instruction#instruction#reque* the multi-task model achieves higher precision, and it correctly labels tags 13 (*apology#instruction*) and 34 (*offer*) where the baseline has zero precision. The CMU subset is the smallest of the data sets tested with less than half the number of training examples as the Frames corpus. There may be a data starvation issue here, as with the BBN subset.

The INCEP2 and CNN+BLSTM models both achieve a statistically significant improvement in f-score over the CNN baseline. Figure 20 shows the precision and recall graphs for a subset of the system action tags on the CNN+BLSTM model. The baseline model performs better on tags 1 (*implicit_confirm#request_info*) and 10 (*status_report#present_info#offer*), and the multi-task model correctly uses labels 13 (*apology#instruction*), 34 (*offer*) and 36 (*explicit_confirm#instruction*) which the baseline completely misses. Tag 34 is a particularly important improvement, since this is one of the most critical tasks of a goal-oriented dialogue system. The CNN+BLSTM model is also at least twice as big as the baseline models with 981,204 trainable parameters versus 255,674 in the BLSTM baseline model and 488,174 in the CNN baseline model.



(a) Per-tag precision. (b) Per-tag recall.

Figure 19: Per-tag metrics for a selection of tags for BLSTM2 model on CMU data.



(a) Per-tag precision. (b) Per-tag recall.

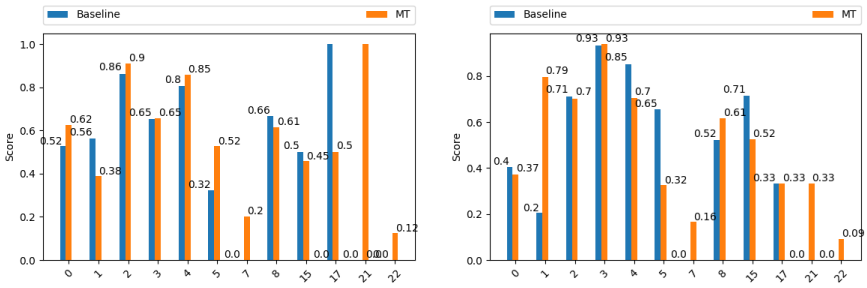
Figure 20: Per-tag metrics for a selection of tags for CNN+BLSTM model on CMU data.

Data	Exp	Acc	Precision	Recall	F-score
SRI	*Exp 0*	62.25%	57.41%	63.04%	57.72%
	Baseline BLSTM	61.75%	57.66%	62.53%	57.36%
	BLSTM1a MT	**62.25%**	**58.68%**	**63.04%**	**58.12%**
	BLSTM1b MT	**61.88%**	56.34%	**62.66%**	57.32%
	BLSTM1c MT	61.75%	**58.19%**N	62.53%	57.21%
	BLSTM2 Skip EMB	**63.25%**	**57.93%**	**64.05%**	**58.61%**
	BLSTM3a SLOT PRE	**62.38%**	**59.27%**	**63.16%**	**58%**
	BLSTM3b SLOT PRE	**62.13%**	**59.16%**	**62.91%**	**58.30%**
	Baseline CNN	62%	56.27%	62.78%	57.39%
	CNN1	**62.75%**	**58.66%***	**63.54%**	**59.09%**
	CNN INCEP1	60.38%	**58.47%**	61.14%	56.09%
	CNN INCEP2	61%	**56.88%**	61.77%	56%
	CNN+BLSTM	**62.5%**	**56.81%**	**63.29%**	**58.49%**

Table 14: COMM SRI performance on multitask ANN models.

All of the subsets of the COMMUNICATOR corpus are smaller than the Frames corpus, but the SRI subset is the largest of them with about 55 % of the number of training examples in the Frames corpus. Table 14 shows the results of the SRI data on the multi-task models. None of the models achieve a statistically significant improvement in accuracy or f-score over the baseline models. The CNN1 model achieves the highest f-score of all the models and does achieve a significant improvement in Precision over the baseline.

Figure 21 shows the precision and recall scores for a selection of tags on the baseline CNN model compared to multi-task model CNN1. The baseline model achieves perfect precision on tag 17 (*unmatched#implicit_confirm*), but both the baseline and CNN1 models miss the same number of utterances that should have been labeled with this tag. For tag 1 (*apology#instruction*) the multi-task model achieves a much higher recall than the baseline model, but the baseline model achieves a higher precision. Also, for tags 7 (*status_report*), 21 (*apology#acknowledgement#request_info*) and 22 (*apology#request_info#instruction#request_info*) the multi-task label correctly uses these tags, where the baseline model misses them completely.



(a) Per-tag precision.

(b) Per-tag recall.

Figure 21: Per-tag metrics for a selection of tags for the CNN1 model on SRI data.

Table 15 shows the primary task and auxiliary task scores for the multi-task model that achieved the highest f-score on the primary task for each corpus. In multi-task learning the goal is to optimize for a primary task while using the information from related tasks. The auxiliary outputs are ignored during inference, so minimal analysis was done on those tasks. Some loss in the performance on the auxiliary tasks is expected, as noted in (Caruana 1998). However, on the ATT corpus user-intent classification accuracy improved by 32%. On the Frames, BBN and SRI corpora user-intent accuracy dropped by at least 3% from the single-task baseline; performance on user-intent classification on the CMU corpus was about the same for baseline and multi-task models. There was a larger loss on the slot-filling task. Performance dropped 8% on the ATT corpus and 11% on the CMU corpus.

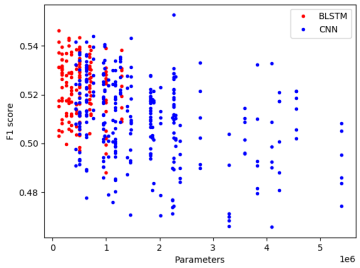
Data	Model	Slot F1	User Acc	Action F1	Params
Frames	BLSTM1c	NA	58.09%	37.93%	451,601
ATT	INCEP2	43.86%	92.70%	53.28 %	4,101,073
BBN	CNN1	NA	90.68%	43.93%	1,832,246
CMU	CNN+BLSTM	49.05%	79.38%	38.53%	981,204
SRI	CNN1	NA	82.88%	59.09%	501,478

Table 15: Task comparison best performing multi-task models

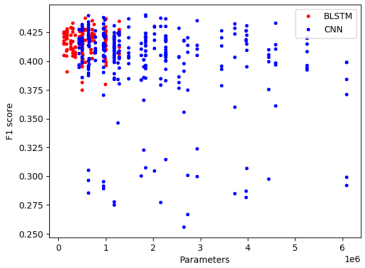
In (Hakkani-Tur et al. 2016) the goal was to jointly-learn multiple tasks by formulating the problem as a sequence prediction problem. For the Frames, BBN, and SRI corpora both the baseline single-task BLSTM and CNN models and the multi-task models improved over the accuracy achieved using the architecture released by the authors. The accuracy increased by 30% with the BLSTM multi-task model and 24% with the CNN multi-task model on the

Frames corpus. On the ATT corpus the multi-task BLSTM1c and BLSTM2 models achieve the same accuracy as the Exp 0 architecture, and the Exp 0 architecture achieved the highest accuracy on the CMU corpus. When comparing models by primary task f-score the majority of baseline and multi-task models improve over the Exp 0 architecture. F-score is the harmonic mean of precision and recall and takes into account how precise a model is when it labels an utterance with a specific tag, as well as the instances where it should have used a tag but didn't. On the Frames corpus the multi-task models achieve a higher recall for the *inform* tag. Both BLSTM baseline and multi-task models achieve a higher f-score on the *affirm* tag, which the Exp 0 architecture scores zero on.

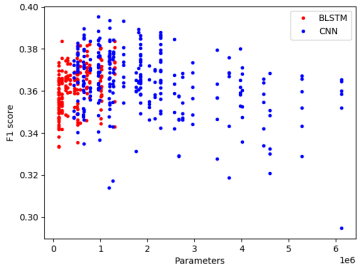
One question worth asking is: Would the performance on these tasks improve with larger models. Larger and deeper neural networks can be used to learn more complicated functions, however with small data sets large models can often memorize the data, and thus fail to learn a representation then generalizes well. Figure 22 contains graphs of the number of trainable model parameters versus f-score for all models in each corpus. If a positive correlation between larger model sizes and high f-scores exists, that would suggest larger models could improve performance. For the BLSTM models there seems to be no correlation between larger f-score and increased model size for any data set. Subfigure (e) suggests a correlation between smaller CNN models and high f-score; slightly less for the SRI corpus.



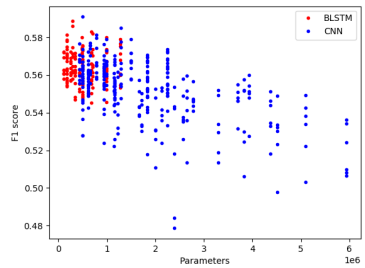
(a) ATT corpus.



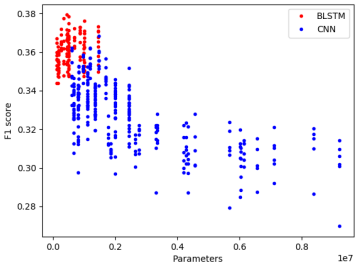
(b) BBN corpus.



(c) CMU corpus.



(d) SRI corpus.



(e) Frames corpus.

Figure 22: Number of trainable parameters v.s. primary task f-score for each corpora.

Conclusion

In this thesis multi-task neural models were designed that used slot-filling and user-intent classification as auxiliary tasks for supervised pre-training of dialog policies. The goal was to bootstrap dialogue policy optimization without the need for hand-written rules and to mitigate the need for a large corpus, which is a requirement for effective reinforcement learning. A dialogue manager that is only trained from data is not flexible enough for new user interactions, so continued learning with reinforcement learning is necessary. However, with a robust pre-trained model the system can avoid common failures that can often occur in the early stages of reinforcement learning, thus reducing user frustration during online training.

No single multi-task model improved performance over the baselines for all data sets tested, but many of the models achieved statistically significant improvement over the single-task baselines and also showed improvement over the the architecture released by (Hakkani-Tur et al. 2016). Deploying the multi-task models as part of a complete dialogue system that interacts with users would be the next step in determining if these results translate to an improved user experience. Continued research should also include repeating the experiments of this thesis on a larger corpus to, hopefully, gather more detailed evidence as to when and how multi-task learning is beneficial for supervised pre-training of dialogue policies.

The CNN models showed statistically significant improvement on three data sets and were faster to train than BLSTM models, even when larger. BLSTM models consistently provided significant improvement on the Frames corpus, but improvement was less consistent on the COMMUNICATOR corpus. In the Frames corpus the discourse is more natural since it was collected with human-human interactions. In the COMMUNICATOR corpus after the initial request most user utterances are limited to one or two word responses to questions presented by the system. This results in a dialogue that is more system initiative and in this scenario the relevant context is now in the preceding system turn, not the user's turn. The CNN+BLSTM model scored significant improvement on the Frames corpus as well as the ATT and CMU subsets of the COMMUNICATOR corpus. This suggests a hybrid BLSTM and CNN model could provide lift when initializing dialogue policies. A network with this architecture could be very large, but training a large network is likely to be cheaper than the other methods for initializing policy networks - whether cost is calculated in the time

of an expert to hand-craft rules, or the time needed for online training with reinforcement learning.

Memory networks (Bordes and Weston 2016) and hierarchical BLSTM networks (Bing and Lane 2017) have been used to create policy networks with positive results. Perhaps a hierarchical multi-task network could take advantage of the hierarchical nature of dialogue data and be leveraged to improve dialogue policy optimization, particularly on corpora similar to the COMMUNICATOR corpus where the needed relevant context is in the dialogue history.

Another research area that may be worth trying is transfer learning - in addition to or as a replacement for multi-task learning. We made the following interesting experience while experimenting with the BLSTM multi-task models: By accident we used the slot-filling model trained on the Frames corpus as the pretrained slot-model for BLSTM3 experiments on the COMMUNICATOR corpus. On the CMU and SRI subsets using the Frames pre-trained slot-filling model, rather than the model trained on the corpus of the experiment, scored higher than the baseline models. These results suggest that if data sets are annotated with similar classes, these could be used to pretrain models that could then be optimized on the data set of interest. This could also help with the creation of robust dialogue policies from minimal initial datasets.

Continued feature engineering or curriculum learning may also provide improvements. In curriculum learning a model is presented with progressively harder examples and given the opportunity to learn the easier examples before moving on to the harder examples. There are a number of ways to formalize "easy" and "hard" examples, and experiments in shape recognition and language modeling suggest that this "start small" approach could help guide training towards better regions in parameter space (Bengio et al. 2009).

Appendices

Appendix A

Corpus Characteristics

ATIS

This corpus includes labels for slot-filling and user speech acts. Figures A.1 and A.2 show the distribution of labels in the corpus. For presentation clarity only the most common slot-labels are shown.

The slot labels used in this corpus:

- aircraft_code
- airline_code
- airline_name
- airport_code
- airport_name
- arrive_date.date_relative
- arrive_date.day_name
- arrive_date.day_number
- arrive_date.month_name
- arrive_date.today_relative
- arrive_time.end_time
- arrive_time.period_mod
- arrive_time.period_of_day
- arrive_time.start_time
- arrive_time.time
- arrive_time.time_relative
- city_name
- class_type connect
- cost_relative
- day_name
- day_number
- days_code
- depart_date.date_relative
- depart_date.day_name
- depart_date.day_number
- depart_date.month_name
- depart_date.today_relative
- depart_date.year
- depart_time.end_time
- depart_time.period_mod
- depart_time.period_of_day
- depart_time.start_time
- depart_time.time
- depart_time.time_relative
- economy
- fare_amount
- fare_basis_code
- flight_days
- flight_mod
- flight_number
- flight_stop
- flight_time
- fromloc.airport_code
- fromloc.airport_name
- fromloc.city_name
- fromloc.state_code
- fromloc.state_name
- meal
- meal_code
- meal_description_mod
- month_name

- period_of_day
- restriction_code
- return_date.date_relative
- return_date.day_name
- return_date.day_number
- return_date.month_name
- return_date.today_relative
- return_time.period_mod
- return_time.period_of_day
- round_trip state_code
- state_name
- stoploc.airport_name
- stoploc.city_name
- stoploc.state_code
- time time_relative
- today_relative
- toloc.airport_code
- toloc.airport_name
- toloc.city_name
- toloc.country_name
- toloc.state_code
- toloc.state_name
- transport_type

The user intent labels used in this corpus:

- atis_abbreviation
- atis_aircraft
- atis_airfare
- atis_airline
- atis_airport
- atis_capacity
- atis_cheapest
- atis_city
- atis_distance
- atis_flight
- atis_flight_no
- atis_flight_time
- atis_ground_fare
- atis_ground_service
- atis_meal
- atis_quantity
- atis_restriction

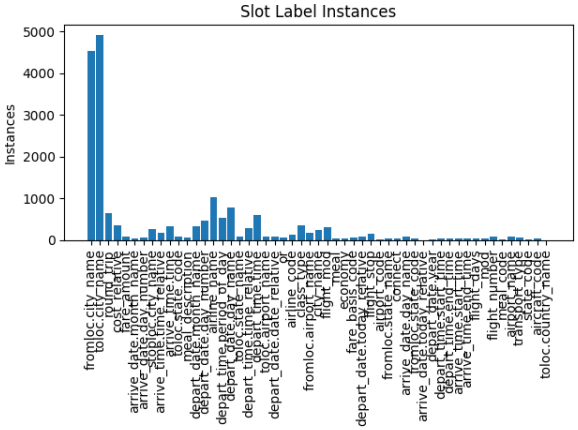


Figure A.1: Instances of the most common slot labels in the ATIS corpus.

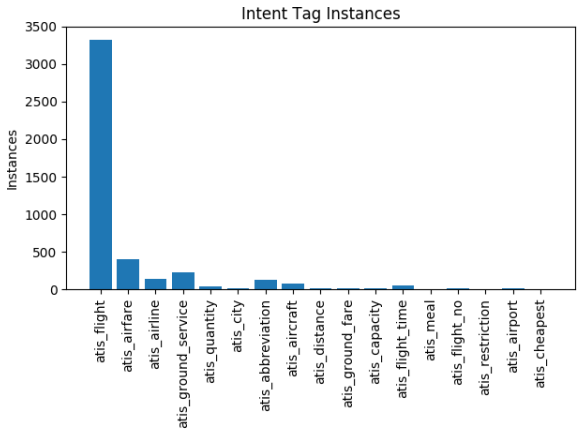


Figure A.2: Instances of user-intent labels in the ATIS corpus.

Maluuba Frames

This corpus includes data annotated with slot values, user-intent labels and system action labels.

The slot labels used in this corpus are:

- action
- arr_time_or
- budget
- category
- count
- count_dst_city
- count_name
- count_seat
- dep_time_dst
- dep_time_or
- dget
- dst_city
- duration
- end_date
- gst_rating
- intent
- max_duration
- min_duration
- n_adults
- n_children
- name or_city
- price
- ref_anaphora
- seat
- str_date

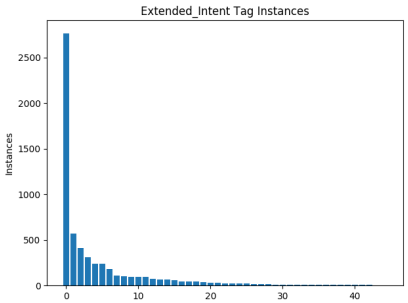
The user-intent labels used in this corpus are listed below. The # is used to separate concatenated tokens.

- inform
- inform#inform
- switch_frame#request
- switch_frame
- request
- switch_frame#inform
- affirm
- request_compare
- switch_frame#moreinfo
- request_alts
- negate#inform
- greeting#inform#inform
- inform#request
- thankyou
- greeting

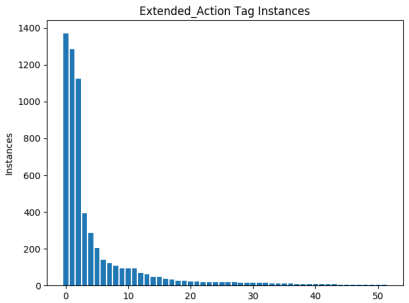
- greeting#inform
- request#switch_frame
- QUESTION
- confirm
- negate
- inform#inform#greeting
- request#inform
- moreinfo
- switch_frame#confirm
- affirm#inform
- inform#switch_frame
- switch_frame#inform#request
- goodbye
- negate#request_alts
- affirm#switch_frame
- switch_frame#affirm
- inform#inform#request
- inform#request_alts
- request_alts#inform
- switch_frame#inform#inform
- inform#negate
- switch_frame#inform#thankyou
- inform#confirm
- inform#request_compare
- negate#switch_frame#request
- inform#thankyou
- switch_frame#request_alts
- switch_frame#request#inform
- moreinfo#switch_frame
- switch_frame#request_compare

The system action labels used in this corpus are listed below. The # is used to separate concatenated tokens.

- inform
- offer
- request
- suggest,
- no_result
- no_result#sorry
- offer#inform
- suggest#inform,
- sorry#no_result#suggest
- greeting
- confirm
- no_result#suggest
- offer#suggest
- affirm
- request#inform
- you_are_welcome
- sorry#inform
- offer#hearmore
- goodbye#inform
- goodbye
- request#greeting
- confirm#request
- request#suggest
- affirm#inform
- offer#request
- sorry
- offer#sorry
- thankyou
- no_result#inform
- offer#no_result#sorry
- canthelp
- thankyou#inform
- offer#no_result
- confirm#suggest
- sorry#suggest
- sorry#canthelp
- offer#suggest#inform
- goodbye#sorry
- no_result#sorry#inform
- hearmore,
- suggest#hearmore,
- sorry#suggest#inform
- no_result#request
- no_result#sorry#request
- confirm#inform
- suggest#affirm
- reject#sorry
- canthelp#inform
- negate
- offer#affirm
- hearmore#inform



(a) Distribution of user-intent labels.



(b) Distribution of system action labels.

Figure A.3: Distribution of tag labels in the Frames corpus.

DARPA COMMUNICATOR

This corpus includes data annotated with slot values, user-intent labels and system action labels.

The following list is the union of the slot types used in the ATT, BBN, CMU and SRI subsets of this corpus. Some of the tags don't appear in all 4 subsets. When this is the case the list following the tag indicates which subset(s) that tag is used in.

- depart_date
- orig_city
- dest_city
- return_depart_date
- depart_time
- return_depart_time
- start_over
- accept_flight_offer
- number
- continue_depart_date
- reject_flight_offer
- return_trip
- continue_dest_city
- continue_depart_time
- hotel_location
- reject_ground_offer [ATT]
- rental_company [ATT,CMU,SRI]
- hotel_name
- continue_trip
- car_rental
- no_return_trip
- accept_ground_offer [ATT,CMU]
- accept_hotel_offer
- reject_car_offer
- accept_car_offer
- reject_hotel_offer
- bye [ATT,BBN,CMU]
- continue_orig_city
- what
- airline
- hotel_city
- before_after [ATT,CMU]
- reject_flight_summary [BBN,CMU]
- accept_flight_summary [BBN,CMU]
- car_interest [BBN,CMU,SRI]

- hotel_date [BBN]
- no_continue_trip [BBN,SRI]
- request_stop [BBN,CMU]
- arrive_time [BBN,CMU,SRI]
- id_number [CMU]
- send_itinerary [CMU]
- request_help [CMU,SRI]
- request_repetition [CMU,SRI]
- continue [CMU]
- car_city [CMU,SRI]
- itinerary [CMU]
- car_date [SRI]

The following list is the union of the user-intent labels used in the ATT, BBN, CMU and SRI subsets of this corpus. Some of the tags don't appear in all 4 subsets. When this is the case the list following the tag indicates which subset(s) that tag is used in.

- provide_info
- NONE
- yes_answer
- no_answer
- command
- provide_info#provide_info
- reprovide_info [ATT,BBN,CMU]
- no_answer#provide_info
- question
- yes_answer#provide_info
- provide_info#provide_info#provide_info [BBN,CMU,SRI]
- no_answer#command [BBN,CMU]
- yes_answer#provide_info#provide_info [BBN,CMU]
- no_answer#yes_answer [BBN,CMU]
- no_answer#provide_info#provide_info [BBN,CMU]
- provide_info#yes_answer [BBN,SRI]
- command#provide_info [BBN,CMU]
- provide_info#provide_info#provide_info#provide_info [BBN,CMU,SRI]
- NONE#provide_info [CMU]
- NONE#yes_answer [CMU]
- NONE#NONE [CMU]
- yes_answer#yes_answer [CMU]
- NONE#command [CMU]
- command#yes_answer [CMU]
- provide_info#NONE [CMU]
- NONE#no_answer [CMU]
- yes_answer#command [CMU]
- no_answer#provide_info#provide_info#provide_info [CMU]
- no_answer#NONE [CMU]
- yes_answer#NONE [CMU]
- NONE#yes_answer#provide_info [CMU]
- yes_answer#yes_answer#provide_info [CMU]
- provide_info#command [CMU]
- yes_answer#yes_answer#yes_answer#yes_answer#yes_answer [SRI]
- yes_answer#yes_answer#yes_answer [SRI]
- no_answer#no_answer#no_answer#no_answer#no_answer [SRI]
- no_answer#no_answer#no_answer [SRI]
- provide_info#provide_info#provide_info#provide_info#provide_i [SRI]

The following list is the union of the system action labels used in the ATT, BBN, CMU and SRI subsets of this corpus. Some of the tags don't appear in all 4 subsets. When this is the case the list following the tag indicates which subset(s) that tag is used in.

- present_info#present_info#offer [ATT,BBN]

- implicit_confirm#request_info [ATT,BBN,CMU]
- request_info
- implicit_confirm#unmatched [ATT]
- explicit_confirm [ATT,CMU,SRI]
- acknowledgement#request_info
- unmatched [ATT,CMU,SRI]
- unmatched#opening_closing#instruction#request_info [ATT]
- status_report#status_report#offer [ATT,CMU,SRI]
- acknowledgement#implicit_confirm#request_info [ATT]
- present_info#present_info#offer#present_info#present_info#offer [ATT]
- request_info#request_info [ATT,CMU]
- apology#explicit_confirm [ATT]
- status_report#status_report#present_info#present_info#offer [ATT]
- offer#present_info#offer [ATT,BBN]
- status_report#offer#present_info#offer [ATT]
- apology#request_info
- implicit_confirm#request_info#implicit_confirm#request_info [ATT,CMU]
- acknowledgement#implicit_confirm#unmatched [ATT]
- status_report#status_report#status_report#present_info#present_info#offer [ATT]
- unmatched#implicit_confirm#explicit_confirm [ATT]
- unmatched#instruction [ATT]
- present_info#present_info#present_info#offer [ATT]
- unmatched#opening_closing#instruction#request_info#request_info [ATT]
- status_report#opening_closing#instruction#request_info [ATT]
- request_info#instruction [ATT,CMU]
- acknowledgement#request_info#request_info [ATT]
- explicit_confirm#instruction [ATT,CMU]
- apology#present_info#present_info#offer [ATT]
- implicit_confirm#request_info#implicit_confirm#unmatched [ATT]
- implicit_confirm#request_info#acknowledgement#request_info [ATT]
- apology#status_report#request_info [ATT]
- acknowledgement#opening_closing#instruction#request_info [ATT]
- status_report#present_info#instruction#instruction#offer [ATT]
- status_report#status_report#status_report#status_report#present_info#present_info#offer [ATT]
- acknowledgement#present_info#present_info#offer [ATT]

- implicit_confirm#request_info#opening_closing#instruction#request_info [ATT]
- implicit_confirm#unmatched#request_info [ATT]
- offer
- implicit_confirm#unmatched#unmatched#implicit_confirm#explicit_confirm [ATT]
- apology#unmatched [ATT]
- acknowledgement#unmatched [ATT]
- opening_closing#instruction#request_info [ATT,CMU]
- present_info#present_info#offer#present_info#present_info#offer#present_info#present_info#offer [ATT]
- instruction#request_info [ATT,BBN,CMU]
- implicit_confirm#unmatched#explicit_confirm [ATT]
- status_report#offer [ATT,SRI]
- apology#unmatched#instruction [ATT]
- status_report#status_report#status_report#status_report#status_report#present_info#present_info#offer [ATT]
- present_info#request_info [ATT]
- present_info#present_info#offer#opening_closing#instruction#request_info [ATT]
- implicit_confirm#request_info#acknowledgement#implicit_confirm#request_info [ATT]
- apology#opening_closing#opening_closing#instruction#request_info [ATT]
- request_info#request_info#request_info [ATT]
- opening_closing#opening_closing#instruction#request_info [ATT]
- status_report#status_report#offer#request_info [ATT]
- present_info#offer [BBN,CMU,SRI]
- implicit_confirm#present_info#offer [BBN,CMU]
- apology#present_info#offer [BBN,CMU,SRI]
- implicit_confirm#status_report#status_report#present_info#present_info#offer [BBN]
- implicit_confirm#status_report#status_report#unmatched#present_info#offer [BBN]
- acknowledgement#acknowledgement#request_info [BBN]
- instruction#present_info#offer [BBN]
- implicit_confirm#status_report#status_report#present_info#offer [BBN]
- apology#instruction#present_info#offer [BBN]
- implicit_confirm#implicit_confirm#request_info [BBN]
- apology#instruction#request_info [BBN,CMU]
- implicit_confirm#unmatched#present_info#offer [BBN]
- acknowledgement#present_info#offer [BBN,CMU]
- implicit_confirm#present_info#present_info#offer [BBN]
- unmatched#present_info#offer [BBN]

- implicit_confirm#request_info#unmatched [BBN]
- acknowledgement#acknowledgement#offer [BBN]
- status_report#request_info [BBN]
- implicit_confirm#unmatched#implicit_confirm#implicit_confirm#implicit_confirm#implicit_confirm#acknowledgement# [BBN]
- implicit_confirm#offer [BBN]
- implicit_confirm#apology#explicit_confirm [BBN]
- implicit_confirm#unmatched#implicit_confirm#implicit_confirm#acknowledgement#request_info [BBN]
- implicit_confirm#unmatched#implicit_confirm#implicit_confirm#implicit_confirm#implicit_confirm#implicit_confirm#r [BBN]
- implicit_confirm#implicit_confirm#implicit_confirm#implicit_confirm#implicit_confirm#request_info [BBN]
- implicit_confirm#implicit_confirm#status_report#status_report#present_info#present_info#offer [BBN]
- implicit_confirm#status_report#status_report#present_info#present_info#unmatched#present_info#offer [BBN]
- implicit_confirm#implicit_confirm#status_report#status_report#unmatched#present_info#offer [BBN]
- apology#offer [BBN,CMU]
- implicit_confirm#implicit_confirm#implicit_confirm#request_info [BBN]
- implicit_confirm#unmatched#implicit_confirm#implicit_confirm#implicit_confirm#implicit_confirm#request_info [BBN]
- apology#explicit_confirm#tbc [BBN]
- acknowledgement#acknowledgement#acknowledgement#acknowledgement#offer [BBN]
- unmatched#request_info [BBN]
- implicit_confirm#status_report#status_report#status_report#present_info#present_info#offer [BBN]
- present_info#offer#present_info#offer [BBN]
- implicit_confirm#implicit_confirm#status_report#status_report#present_info#offer [BBN]
- acknowledgement#unmatched#implicit_confirm#unmatched#implicit_confirm#unmatched#implicit_confirm#request_in [BBN]
- implicit_confirm#status_report#status_report#present_info#present_info#present_info#present_info#offer [BBN]
- implicit_confirm#implicit_confirm#unmatched#present_info#offer [BBN]
- implicit_confirm#status_report#status_report#status_report#status_report#status_report#status_report#present_info# [BBN]
- opening_closing [CMU,SRI]
- implicit_confirm#status_report#present_info#offer [CMU]
- implicit_confirm#status_report#request_info [CMU]
- apology [CMU]
- status_report#present_info#offer [CMU,SRI]

- implicit_confirm#status_report#implicit_confirm#present_info#offer [CMU]
- apology#instruction [CMU,SRI]
- status_report [CMU,SRI]
- implicit_confirm [CMU]
- acknowledgement#present_info#present_info#instruction#instruction [CMU]
- implicit_confirm#status_report [CMU]
- instruction#instruction [CMU]
- apology#opening_closing [CMU]
- present_info#offer#request_info [CMU]
- acknowledgement#acknowledgement#acknowledgement#instruction#instruction [CMU]
- status_report#present_info#request_info [CMU]
- instruction#instruction#opening_closing#instruction#instruction#request_info [CMU]
- request_info#apology#instruction [CMU]
- acknowledgement [CMU]
- unmatched#opening_closing [CMU]
- acknowledgement#present_info#acknowledgement#instruction#instruction [CMU]
- unmatched#opening_closing#opening_closing [CMU]
- opening_closing#opening_closing [CMU]
- instruction [CMU]
- present_info#unmatched#status_report [CMU]
- present_info#offer#status_report [CMU]
- acknowledgement#status_report#present_info#offer [CMU]
- apology#instruction#instruction [CMU]
- present_info#offer#apology#present_info#offer [CMU]
- status_report#unmatched#offer [CMU,SRI]
- implicit_confirm#implicit_confirm#status_report#present_info#offer [CMU]
- status_report#implicit_confirm#present_info#offer [CMU]
- request_info#apology#request_info [CMU]
- request_info#unmatched#offer [CMU]
- acknowledgement#acknowledgement#instruction#instruction [CMU]
- implicit_confirm#status_report#implicit_confirm#unmatched#status_report#request_info [CMU]
- apology#implicit_confirm#status_report#request_info [CMU]
- implicit_confirm#status_report#implicit_confirm#unmatched#status_report#present_info#offer [CMU]
- request_info#implicit_confirm#request_info [CMU]
- request_info#implicit_confirm [CMU]

- implicit_confirm#instruction#instruction [CMU]
- explicit_confirm#explicit_confirm [CMU]
- acknowledgement#status_report [CMU]
- present_info#offer#apology [CMU]
- present_info#offer#acknowledgement [CMU]
- status_report#request_info#implicit_confirm [CMU]
- present_info#offer#implicit_confirm#present_info#offer [CMU]
- explicit_confirm#offer#instruction [CMU]
- status_report#present_info#unmatched [CMU]
- present_info#request_info#status_report [CMU]
- unmatched#implicit_confirm#unmatched#implicit_confirm [CMU]
- acknowledgement#apology#request_info [CMU]
- apology#offer#instruction [SRI]
- apology#request_info#instruction [SRI]
- apology#request_info#instruction#request_info [SRI]
- status_report#present_info#present_info#offer [SRI]
- apology#request_info#instruction#instruction [SRI]
- acknowledgement#explicit_confirm [SRI]
- status_report#apology#present_info#offer [SRI]
- unmatched#implicit_confirm [SRI]
- apology#acknowledgement#request_info [SRI]
- status_report#apology#instruction [SRI]

ADD user-intent distribution graphs
ADD system action distribtuion graphs

Appendix B

Multitask Model Diagrams

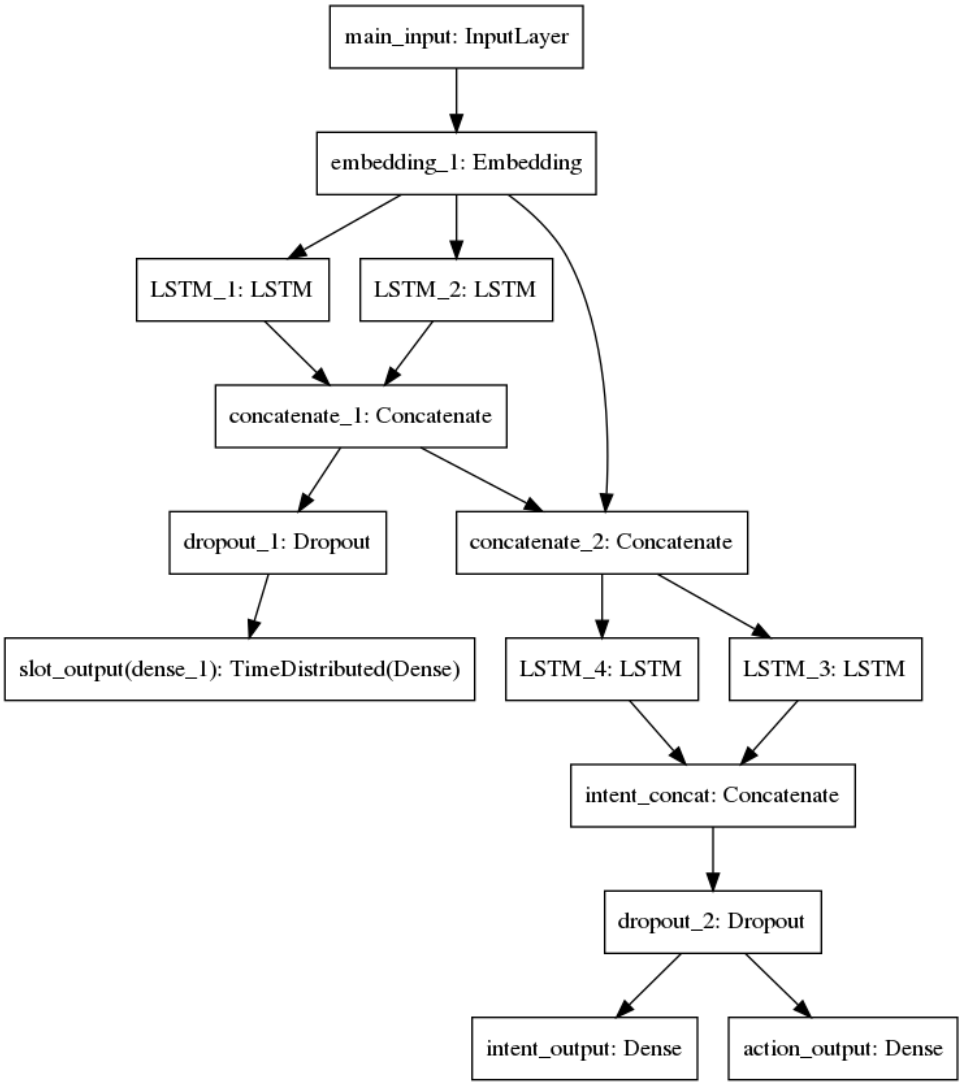


Figure B.1: Model diagrams for BLSTM2 multi-task model.

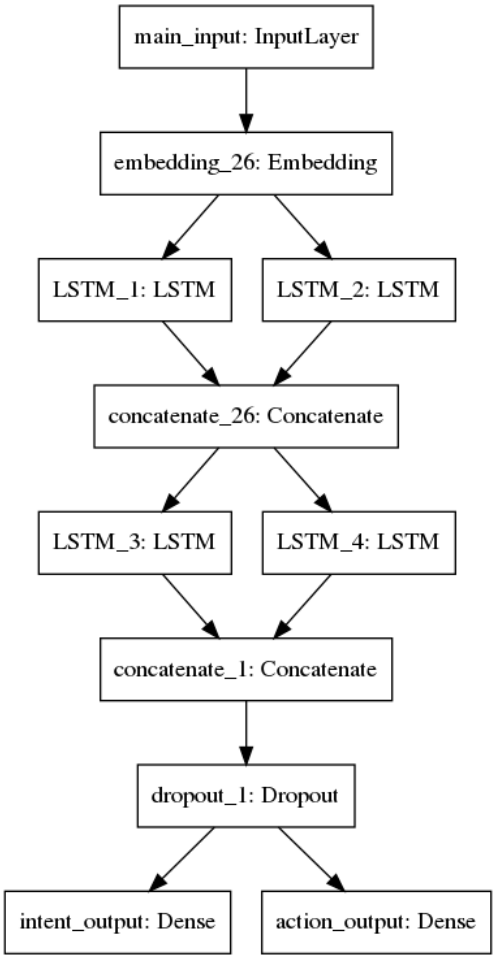


Figure B.2: Model diagram for BLSTM3 multi-task model

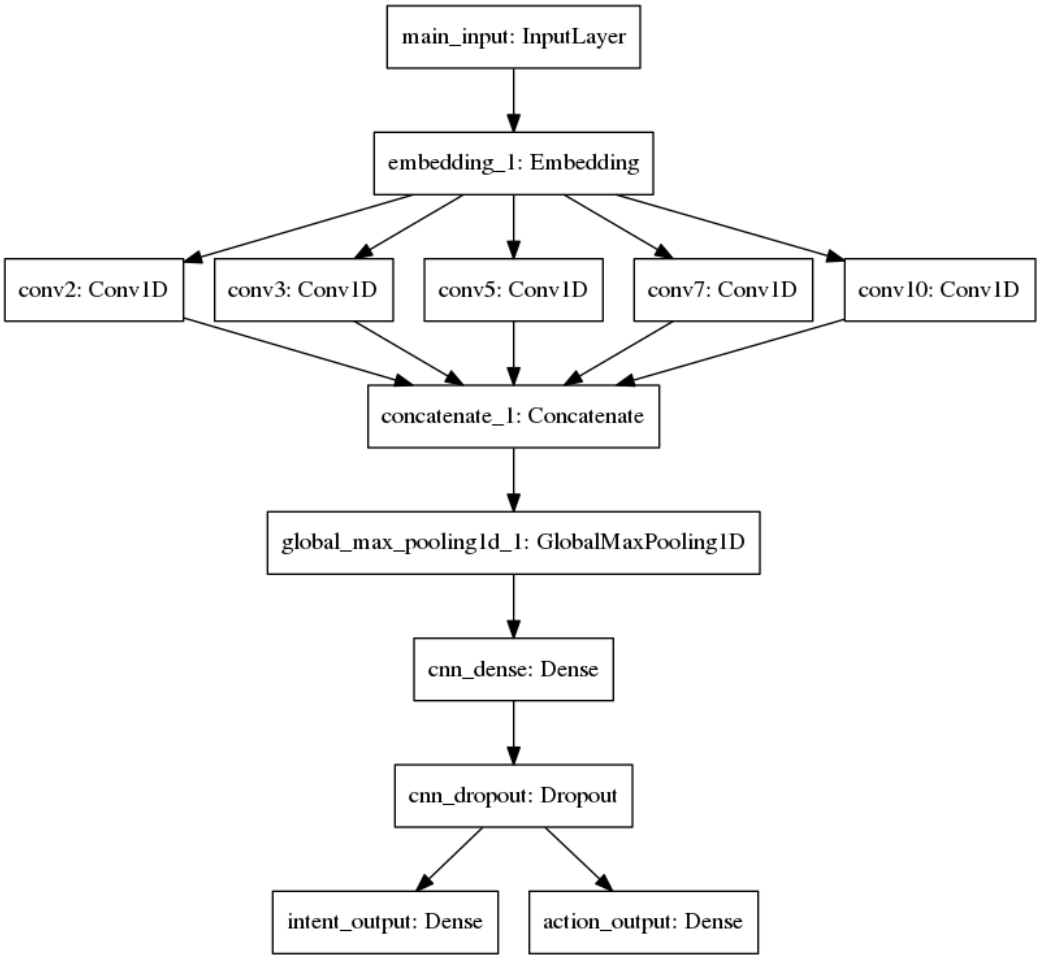


Figure B.3: Model diagram for CNN1 multi-task model.

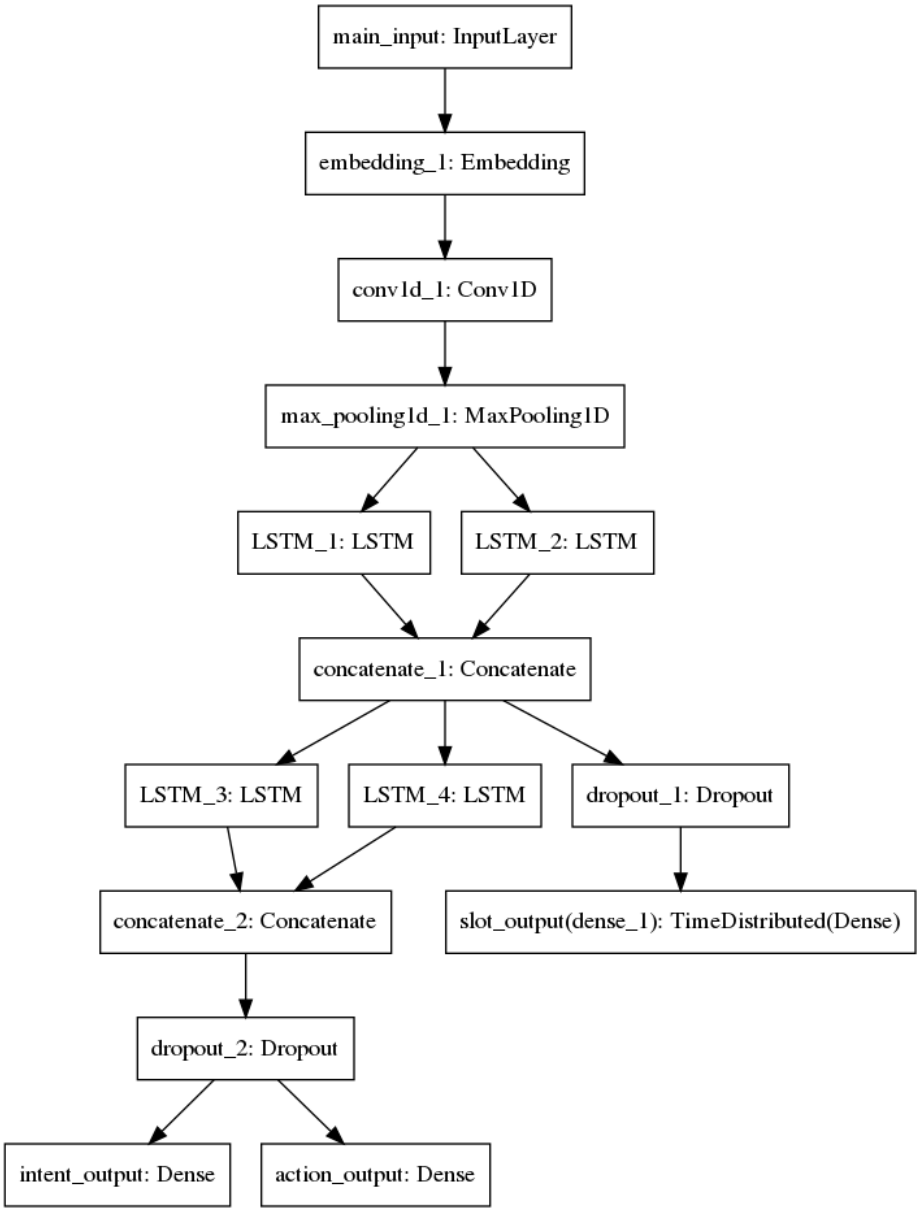


Figure B.4: Model diagram for the hybrid CNN and BLSTM multi-task model..

Appendix C

Additional Experimentation Graphs

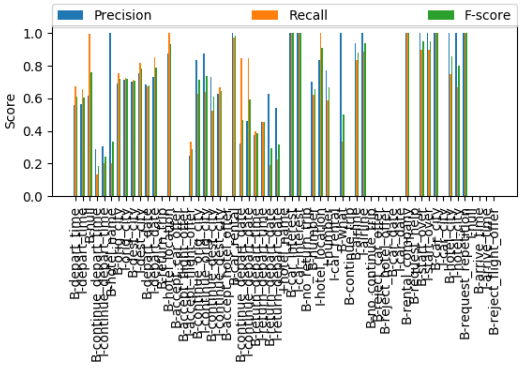


Figure C.1: Per-tag metrics for the single-task BLSTM model on SRI data.

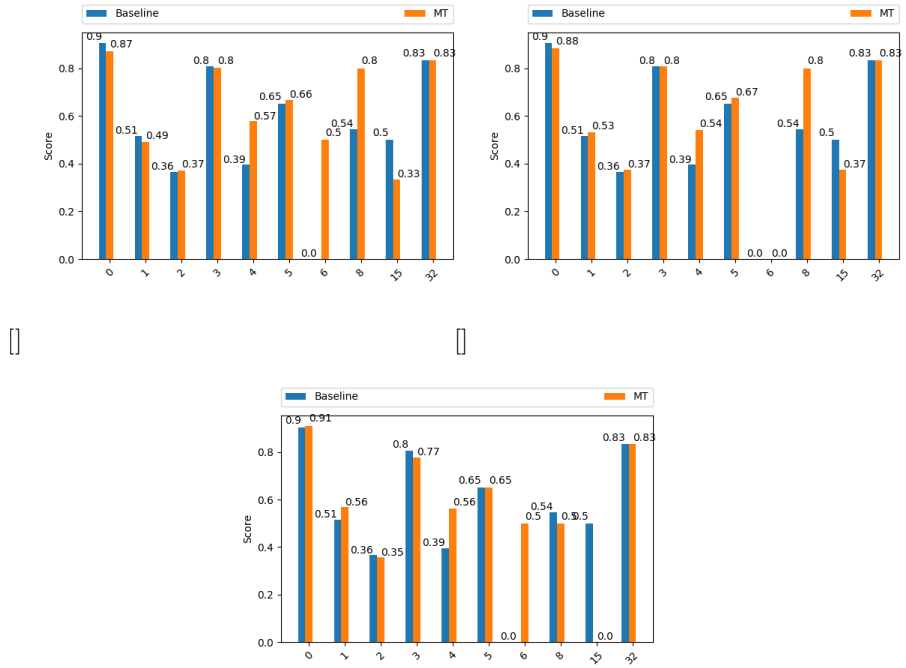


Figure C.2: Precision results for three multi-task models on the ATT data.

References

Arık, S. Ö. et al. (2017). "Deep Voice: Real-time Neural Text-to-Speech." In: *ICML 2017*.

Bengio, Y. et al. (2009). "Curriculum learning." In: *International Conference on Machine Learning (ICML)*.

Bing, Liu and Ian Lane (2017). "An End-to-End Trainable Neural Network Model with Belief Tracking for Task-Oriented Dialog". In: *Interspeech*.

Bobrow, D. G. et al. (1977). "GUS, A frame driven dialog system." In: *Artificial Intelligence* 8, pp. 155–173.

Bordes, A. and J. Weston (2016). "Learning end-to-end goal-oriented dialog." In: arXiv preprint arXiv:1605.07683.

Caruana, R. (1998). "Multitask Learning." In: *Autonomous Agents and Multi-Agent Systems* 27.1, pp. 95–133.

Chen, D-N. et al. (2016). "Syntax or semantics? knowledge-guided joint semantic frame parsing." In: *Proceedings of the 6th IEEE Workshop on Spoken Language Technology*. Pp. 348–355.

Chen, H. et al. (2017). "A Survey on Dialogue Systems: Recent Advances and New Frontiers." arXiv preprint arXiv:1711.01731.

Colby, K. M., S. Weber, and F. D. Hilf (1971). "Artificial paranoia." In: *Artificial Intelligence* 2.1, pp. 1–25.

Deng, L. et al. (2012). "Use of kernel deep convex networks and end-to-end learning for spoken language understanding." In: *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pp. 210–215.

Dong, D. et al. (2015). "Multi-Task Learning for Multiple Language Translation." In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pp. 1723–1732.

El Asri, L. et al. (2017). "Frames: A corpus for adding memory to goal-oriented dialogue systems." In: *preprint on webpage at* `https://www.microsoft.com/en-us/research/publication/frames-corpus-adding-memory-goal-oriented-dialogue-systems/`.

Gasic, M. and S. Young (2014). "Gaussian processes for POMDP-based dialogue manager optimisation". In: *IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING*.

Georgila, K., O. Lemon, and J. Henderson (2005). "Annotation of COMMUNICATOR Dialogue Data for Learning Dialogue Strategies and User Simula-

tions." In: *Proceedings of the 9th Workshop on the Semantics and Pragmatics of Dialogue (SEMDIAL:DIALOR)*, pp. 61–68.

Georgila, K., O. Lemon, J. Henderson, and J.D. Moore (2009). "Automatic annotation of context and speech acts for dialogue corpora." In: *Natural Language Engineering* 15.3, pp. 315–353.

Hakkani-Tur, D. et al. (2016). "Multi-domain joint semantic frame parsing using bi-directional rnn-lstm." In: *Proceedings of Interspeech.* Pp. 715–719.

Henderson, J., O. Lemon, and K. Georgila (2007). "Hybrid Reinforcement/Supervised Learning of Dialogue Policies from Fixed Data Sets". In: *ACL*.

Johnson, M. et al. (2016). "Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation." arXiv Preprint arXiv:1611.0455.

Jurafsky, D. and James H. Martin (2017). *Speech and Language Processing.* 3rd edition Draft.

Lemon, O., K. Georgila, et al. (2006). "An ISU Dialogue System Exhibiting Reinforcement Learning of Dialogue Policies: Generic Slot-Filling in the TALK In car System." In: *Proc. of the European Chapter of Association of Computational Linguistics.* 3119122.

Lemon, O., A. Gruenstein, and S. Peters (2002). "Multi-tasking and Collaborative Activities in Dialogue Systems." In: *Proc. of the SIGDIAL Workshop on Discourse and Dialogue.* 113124.

Lemon, O. and O. Pietquin (2007). "Machine Learning for Spoken Dialogue Systems." In: *INTERSPEECH*.

Li, X. et al. (2017). "End-to-end task-completion neural dialogue systems." In: *Proceedings of the The 8th International Joint Conference on Natural Language Processing*, pp. 733–743.

O., Vinyals and Quoc V. Le (2015). "A neural conversational model." arXiv preprint arXiv:1506.05869.

Padmakumar, A., J. Thomason, and R. Mooney (2017). "Integrated learning of dialogstrategies and semantic parsing." In: *Proceedings ofthe 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL).*

Pan, S. J. and Q. Yang (2009). "A Survey on Transfer Learning". In: *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*.

Price, P.J. (1990). "Evaluation of spoken language systems: The ATIS domain." In: *Proceedings of the DARPA Workshop on Speech and Natural Language, Hidden Valley, PA,*

Reuder, S. (2017a). *An Overview of Multi-Task Learning in Deep Neural Networks.* URL: `http://ruder.io/multi-task/index.html#fn:5`.

— (2017b). *Multi-Task Learning Objectives for Natural Language Processing.* URL: `http://ruder.io/multi-task/index.html#fn:5`.

Serban, I. V., R. Lowe, et al. (2015). "A survey of available corpora for building data-driven dialogue systems." arXiv preprint arXiv:1512.05742.

Serban, I. V., A. Sordoni, et al. (2015). "Hierarchical neural network generative models for movie dialogues." arXiv preprint arXiv:1507.04808.

Shang, L., Z. Lu, and H. Li (2015). "Neural responding machine for short-text conversation." In: *ACL*, pp. 1577–1586.

Su, P.-H. et al. (2016). “Continuously learning neural dialogue management.” arXiv preprint arXiv:1606.02689.

Szegedy, C. et al. (2014). “Going deeper with convolutions”. In: *CoRR, abs/1409.4842*, pp. 1746–1751.

Toshniwal, S. et al. (2017). “Multitask Learning with Low-Level Auxiliary Tasks for Encoder-Decoder Based Speech Recognition.” arXiv preprint arXiv:1704.01631.

Waibel, A., H. Sawai, and K. Shikano (1989). “Modularity and Scaling in Large Phonemic Neural Networks”. In: *IEEE Transactions on Acoustics, Speech and Signal Processing* 37.12, pp. 1888–1898.

Weizenbaum, J. (1966). “ELIZA—a computer program for the study of natural language communication between man and machine.” In: *Communications of the ACM* 9.1, pp. 36–45.

Wen, T-H. et al. (2016). “A Network-based End-to-End Trainable Task-oriented Dialogue System.” In: *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Williams, J. D. (2012). “A belief tracking challenge task for spoken dialog systems.” In: *NAACL-HLT Workshop on Future Directions and Needs in the Spoken Dialog Community: Tools and Data*, pp. 23–24.

Williams, J. D. and G. Zweig (2016). “End-to-end lstm-based dialog control optimized with supervised and reinforcement learning.” arXiv preprint arXiv:1606.01269.

Williams, J. et al. (2013). “The dialog state tracking challenge.” In: *Proceedings of the SIGDIAL 2013 Conference*, pp. 404–413.

Yeh, Alexander (2000). “More accurate tests for the statistical significance of result differences”. In: *Proceedings of the 18th conference on Computational linguistics, COLING ’00, Saarbrücken, Germany*.

Yoon, Kim (2014). “Convolutional Neural Networks for Sentence Classification”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar*, pp. 1746–1751.

Zhang, Y. and Q. Yang (2017). “A survey on multi-task learning.” arXiv preprint arXiv:1707.08114.

Zhang, Ye and Byron C. Wallace (2015). “A sensitivity analysis of (and practitioners’ guide to) convolutional neural networks for sentence classification.” arXiv preprint arXiv:1510.03820.

Zhao, T. and M. Eskenazi (2016). “Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning.” In: *the Annual SIGdial Meeting on Discourse andDialogue (SIGDIAL)*.

Zoph, B. and K. Knight (2016). “Multi-Source Neural Translation.” In: *NAACL*, pp. 30–34.