An Analysis of Using Semantic Parsing for Speech Recognition

Rodolfo Corona

Abstract

This thesis explores the use of semantic parsing for improving speech recognition performance. Specifically, it explores how a semantic parser may be used in order to re-rank the n-best hypothesis list generated by an automatic speech recognition system. We also explore how system performance is affected when retraining the system's acoustic model using a portion of the re-ranked data.

1 Introduction

As access to technological advances continues to grow in our society, the consumer demand for these advances has increased. With this increase in users comes a need for making these technologies more intuitive and user-friendly. One avenue that has been pursued to facilitate this is that of speech recognition.

Prominent technology companies such as Apple, Google, and Microsoft make it possible for a variety of their products to be interfaced by simply speaking to them. Allowing users to interact with a product by using only their voice not only has the potential to increase efficiency through single-utterance commands, but also relieves the user from having to learn how to use a possibly complicated interface. While **automatic speech recognition** (ASR) technology has experienced considerable advances, its efficacy is only just beginning to reach a point which would allow for a wide adoption of the technology. Thus, it is desirable to continue exploring methods which could make way for greater recognition performance.

In this thesis, we investigate methods in which semantic parsing may be used to re-rank the n-best hypothesis list of a speech recognizer in order to reduce error rates in recognition. Semantic parsing, which is discussed in more detail in section 2.3, may be defined as the process through which the meaning of natural language expressions is inferred and mapped to a computer interpretable formalism. The motivating intuition behind our work is that given a list of ASR hypotheses, the correct hypothesis should generally also be the most "meaningful" to the semantic parser, something which is formalized through parse confidence scores. Further, we investigate how this re-ranking method may be used to automatically induce new training examples for the recognizer's statistical models.

Through our experiments, we find that our re-ranking algorithm results in a significant improvement in language understanding at the cost of ASR transcription accuracy.

2 Background

In this section we describe relevant background in ASR, particularly describing the CMU Sphinx-4 speech recognition system which was used in this work. Semantic parsing as is relevant to this work is also briefly discussed.

2.1 Speech Recognition

The goal of ASR systems is to find the most likely sentence $\hat{W}$ that was uttered by a user given their speech input, something which is expressed by the following equation:

$$\hat{W} = \operatorname*{argmax}_{W \in L} P(W|O)$$

Where L is the given language, $W \in L$ are possible sentences within it, and O is the observed audio input (Jurafsky & Martin, 2009). Using Bayes' rule, this expression may be better modelled as:

$$\hat{W} = \operatorname*{argmax}_{W \in L} \frac{P(O|W)P(W)}{P(O)} = \operatorname*{argmax}_{W \in L} P(O|W)P(W)$$

The denominator $P(O)$ may be ignored since it is a common factor amongst all candidate sentences.

The probability for a given sentence $P(W)$ may be computed using a **language model**, which is a statistical distribution of word sequences in a language. The set of all possible words, or **tokens**, in a language model is known as the **vocabulary**. In general, given a word sequence w_1^N, the model will assign the probability $P(w_1^N)$ to the sequence proportionally to how often the sequence was seen during training. As the length of sequences N increases, sequence appearances will become increasingly sparse in the training corpus. In order to combat the sparsity problem, this distribution may be approximated using **n-grams**(Rosenfeld, 2000), which are word sequences of length n. During training, the model will instead construct the distribution of unique n-grams. With this approach, $P(w_1^N)$ is approximated as follows:

$$P(w_1^N) \approx \prod_{i=1}^{N} P\left(w_i | w_{i-n+1}^{i-1}\right)$$

Intuitively, larger n-grams allow for more accurate models but require larger amounts of training data in order to be effective. In order to take advantage of higher order n-grams for which enough information is available while still having the option of using lower order n-grams if necessary, n-gram **backoff language models** may be used. N-gram backoff language models contain distributions for all n-grams from 1 up to n. When computing the probability of a sequence $P(w_1^N)$, the discounted probability of the highest order n-gram with a count above a threshold k is used.

The conditional probability $P(O|W)$ may be computed using an **acoustic model**. Acoustic models are constructed using Hidden Markov Models (HMMs)(Rabiner, 1989), which consist of a series of states, modelled by a directed graph, with transition probabilities between them. Each state contains a set of possible observations that may be made when in said state as well as their probabilities. In an HMM, the states are not directly observed (i.e. are said to be hidden) and the most likely sequence of states must be inferred using a sequence of given observations. This inference may be performed using procedures such as the Viterbi algorithm (Forney, 1973). Using this framework for ASR, per-word HMMs are constructed where **phones**, which are basic units of speech, serve as the hidden states, while feature vectors extracted from the audio input serve as the observations. Additionally, most ASR implementations further split each phone state into a three state HMM modelling the beginning, middle, and end of the phone, something which is beneficial since phones tend to have different features at different stages of their utterance. An example of a word-level HMM may be seen in Figure 1.

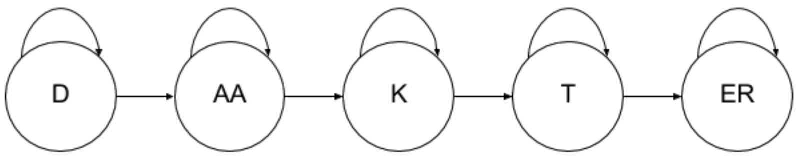


Figure 1: *An example of an acoustic model HMM for the word "doctor". The nodes in the graph represent the phones which compose the word.*

The main evaluation metric used for speech recognition accuracy is known as **word error rate** (WER), which, if we let o and c be the recognition output and correct phrase respectively, may be computed as follows:

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

Where S is the the number of words that were substituted in c by o (e.g. confusing *gray* with *great*), D is the number of deletions (i.e. the number of words in c not present in o), I is the number of insertions (i.e. the number of words in o not present in c), and N is the total number of words in c.

2.2 The CMU Sphinx-4 Speech Recognition System

Sphinx-4 is a speech recognition system jointly developed by Carnegie Mellon University, Sun Microsystem Laboratories, and Mitsubishi Electronic Research

Laboratories (Lamere, Kwok, Gouvea, et al., 2003; Lamere, Kwok, Walker, et al., 2003; Walker et al., 2004). The system is designed to be heavily modular, allowing users to experiment with different language and acoustic models, decoders, and a variety of other components.

The system is composed of three main modules, the *FrontEnd*, the *Linguist*, and the *Decoder* (Walker et al., 2004). The FrontEnd takes raw audio data as input, and generates a feature vector that may be used during the decoding process by the Decoder.

The Linguist constructs a search graph for sentence hypotheses based on a given language model, acoustic model, and dictionary (i.e. a phone pronunciation lexicon for words in the domain vocabulary). The default Sphinx-4 language model is an ARPA[1] formatted trigram backoff language model. The search graph is constructed as a directed acyclic graph (DAG) by connecting each word HMM taken from the acoustic model and connecting them with edges whose transition weights are derived from the language model. This graph is then used to construct a **trellis**, which is a graph which models state transitions over time. Trellis construction begins with the acoustic model HMM starting state, S_0, being mapped to time step t_0. All states to which there is a transition from S_0 are then mapped to t_1, a procedure which is then repeated for each state at each time step. Due to memory constraints that may arise from large vocabularies, the trellis is generally constructed dynamically as each of its components is needed. An example of a trellis is given in Figure 2.

The Decoder takes feature vectors from the FrontEnd and traverses the search graph generated by the Linguist in order to form a list of recognition hypotheses (Lamere, Kwok, Walker, et al., 2003). The search operation is done using the Bushderby algorithm (Singh, Warmuth, Raj, & Lamere, 2003), which subsumes the Viterbi and forward algorithms depending on the chosen hyper-parameter values.

[1]http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html

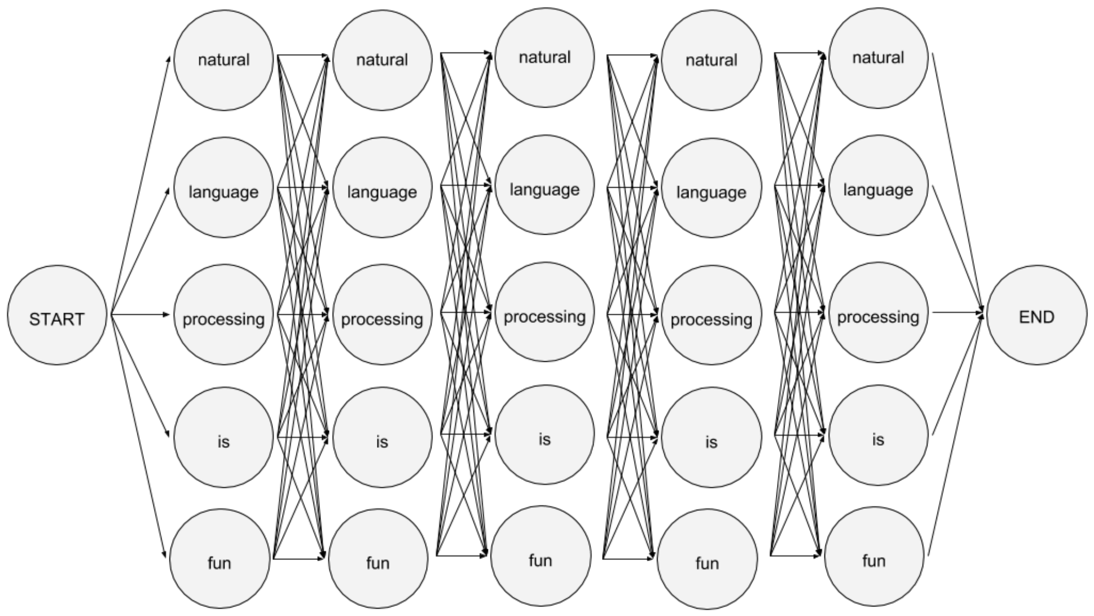


Figure 2: *An example of a trellis built over 5 time steps with a vocabulary of 5 words.*

A token tree is generated during the search, using a token passing algorithm (Young, Russell, & Thornton, 1989), which may be queried for a list of hypotheses. Each node in the tree contains a token which in turn consists of the overall acoustic and language scores as well as other scoring information for a given path in the search. During the search, the tree is pruned in order to conserve computational resources.

It must be noted that in recent years there have been great advances in speech recognition which use deep recurrent neural networks (Graves & Jaitly, 2014; Xiong et al., 2016), particularly Long Short Term Memory networks (LSTMs). These methods have achieved state of the art performance and have even achieved human parity on the NIST 2000 dataset[2]. Unfortunately, although there are a variety of libraries with which neural network models may be constructed[3][4][5], source code for the specific architectures used in these works has not been made

[2]http://www.itl.nist.gov/iad/mig/tests/ctr/2000/h5_2000_v1.3.html

[3]https://www.tensorflow.org/

[4]http://caffe.berkeleyvision.org/

[5]https://keras.io/

readily available. Sphinx-4 was chosen as our ASR framework in part due to its being open-source.

2.3 Semantic Parsing

2.3.1 Mapping Natural to Formal Language

Intuitively, semantic parsing may be described as the task of deriving the meaning of a natural language phrase in a form which is interpretable by a computer. More concretely, this generally entails finding a mapping from a given phrase to an unambiguous formal language representation, such as **first order logic** (FOL). For example, the phrase "Bring Dr. Smith the marker" could feasibly be mapped to the FOL representation $bring(smith, marker)$. In order to accomplish this, a semantic parser's knowledge is comprised of two main components, the **ontology** and the **lexicon** (Jurafsky & Martin, 2009).

The ontology of the parser is what denotes the set of atoms which are used to derive meaning from phrases. In other words, the ontology will contain the set of entities of the domain, the possible properties or relations that they may possess, as well as the actions which any entities might be able to take. An example of what an ontology could look like using FOL is shown in Figure 3.

Entities	Dr. Smith, John, Jane, marker
Properties	isProfessor(x), isStudent(x)
Relations	professorOf(x,y), studentOf(x,y)
Actions	bring(x,y)

Figure 3: *An ontology for a small domain.*

Whereas the ontology denotes the building blocks of a domain with which a system may form meaning, the lexicon ascribes pertinent linguistic information to certain words or phrases in the vocabulary for use by the system. For example,

Word or Phrase	**Mapping in Ontology**
Dr. Smith	Dr. Smith
The professor	Dr. Smith
John	John
Johny	John
Jane	Jane

Figure 4: *An example lexicon which might pertain to the domain of the ontology in Figure 3. This lexicon maps words or phrases which may be found in natural language input to grounded FOL atoms found in the ontology.*

a lexicon may map natural language words or phrases which the system might encounter to atoms in the ontology, as may be seen in Figure 4. Note the distinction between the names of FOL atoms or predicates, such as *John*, and actual words or phrases in the vocabulary, such as *Johny*, which may be found in the natural language input itself. As will be described in section 2.3.2, lexicons may also contain other types of information. For example, ASR systems, such as the CMU Sphinx system, make use of lexicons which map words in their vocabulary to a set of phones denoting their pronunciation.

2.3.2 Semantic Parsing with Combinatory Categorical Grammars

The semantic parser used in this work was developed within the UTCS department and is modeled after (Liang & Potts, 2015). The parser is analogous in function to the University of Washington Semantic Parsing Framework (Artzi & Zettlemoyer, 2013a), which uses **typed λ-calculus** in conjunction with **Combinatory Categorial Grammars** (CCGs) (Steedman & Baldridge, 2011) to compute meaning representations for given natural language inputs.

λ-calculus is a high-order logic formalism which subsumes FOL. Through the use of the λ term, a λ-calculus expression may exhibit behavior akin to that of a function. For example, the expression $\lambda x.bring(x, marker)$ may be applied to the atom $John$, which would evaluate to the expression $bring(John, marker)$. By extending the formalism with types, expression arguments may be constrained (Church, 1940). For example, the expression above could be extended to require arguments of a type which represent people:

$$\lambda x : person.bring(x, marker)$$

In our work, predicate and atom types are specified in the system's ontology.

CCGs are a categorial formalism which assigns syntactic categories to logical expressions:

$$ADJ : \lambda x.happy(x)$$

These category and expression pairs may be used in order to map natural language phrases to logical expressions. This is accomplished through the use of combinator categories which act as functions, for example:

$$S/NP : \lambda x.bring(x, marker)$$

These combinators are read from right to left, where the rightmost category is the parameter, while everything left of the slash is what is returned. The direction of the slash (forward or backward) determines whether the combinator consumes the token on its left or on its right. If we let S denote the complete sentence category, then the above example represents a combinator which consumes a noun phrase (such as a person) on its right and returns a full sentence. For the purposes of semantic parsing, words in the vocabulary may be mapped to such pairs within the lexicon:

$$happy \vdash ADJ : \lambda x.happy(x)$$

An example of what could be a feasible CCG parse tree for the phrase "John is happy" may be seen in Figure 5.

Because of the prohibitive time investment necessary to generate annotated CCG parse trees for phrases, generally only the final semantic form of each

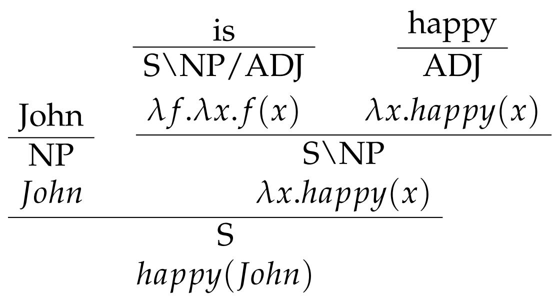


Figure 5: *A CCG parse of the phrase "John is happy". Note that the direction in which an argument is consumed depends on the direction of the slash specified by a combinator.*

phrase is specified in a training corpus (e.g. the training pair ["John is happy, $happy(John)$]). Due to this problem, it is necessary for the training algorithm used by the parser to generate and rank lists of candidate parses during training, something which is generally done through the use of variants of the CKY and perceptron algorithms (Artzi & Zettlemoyer, 2013b). Through this procedure, the parser incrementally adds entries to its lexicon and attempts to infer new rules for combining categories which will allow it to generate valid parses for the phrase and semantic form pairs given during training.

3 Related Work

Various different methods have been used in order to re-rank the n-best hypothesis list generated by ASR systems. Close to the phonetic level, Ananthakrishnan et al. (Ananthakrishnan & Narayanan, 2007) trains a prosody model which is used to score hypotheses based on their syllabic transcription. Similarly, Twiefel et al. (Twiefel, Baumann, Heinrich, & Wermter, 2014) takes Google ASR results and transforms them into sequences of phonemes. The generated list of phonemes is then post-processed by attempting to match it either at the sentence level to in-domain sentences, or at the word level to words in the given domain's vocabulary. Interestingly, their work also experimented with replacing the Sphinx-4 FrontEnd with their generated phonemes list and allowing Sphinx to perform the decoding using in-domain language and acoustic models.

Peng et al. (Peng, Roy, Shahshahani, & Beaufays, 2013) takes an intriguing approach by feeding each entry in the n-best list to a search engine, extracting a set of features from both the hypothesis and search results, and re-ranking the list by passing the feature vectors for each hypothesis through a maximum entropy (ME) model. Morbini et al. (Morbini et al., 2012) also uses a ME model in order to re-rank the list. The model is trained on features that are extracted from natural language understanding (NLU) categories that each hypothesis is given. Their work also explored re-ranking using a discriminative language model trained using the perceptron algorithm as well as by generating the hypothesis list from a combination of three different ASR systems' respective lists.

Syntactic and semantic parsing have also been used previously in different capacities for this problem. Zechner et al. (Zechner & Waibel, 1998) tags each hypothesis using a part of speech (POS) tagger. The generated tag sequence for each hypothesis is then parsed using a chunk based parser, with parses being scored based on their coverage of the sentence (i.e. number of words skipped during parsing). Each hypothesis is then passed through a neural network which takes as input the hypothesis recognition confidence score, chunk coverage score, and chunk language model score (which is assigned by a language model trained on chunk n-grams), and outputs a predicted WER rate which is used to re-rank the hypothesis. Basili et al. (Basili, Bastianelli, Castellucci, Nardi, & Perera, 2013) generates full syntactic parses and semantic representations for each hypothesis and re-ranks the list by using a kernel function. The syntactic parse for a hypothesis is generated using the Stanford Parser[6], while a semantic representation for it is derived through a linear combination of the distributional representation vectors of each of its constituent words (which are built through a

[6]http://nlp.stanford.edu/software/lex-parser.shtml

co-occurrence matrix).

Erdogan et al. (Erdogan, Sarikaya, Chen, Gao, & Picheny, 2005), which is closest to this work, uses semantics to re-score the hypothesis list. Similarly to Morbini et al., they use a concept sequence model as the first component of their re-scoring pipeline. Concept sequences are generated for a given input using a shallow semantic parser, which assigns each word a concept label. The generated concept label sequences are then themselves parsed which results in a hierarchical semantic parse being assigned to the entire inputted phrase. A ME model, which is trained jointly on sentences from the training data as well as features extracted from their parses, is used to re-rank the hypothesis list. The semantic parser used in our work, as described above, generates parses which grounds phrases to meaningful logical form representations using a lexicon and knowledge-base, something which may be differentiated from the more general labels given to phrases in Erdogan et al. Furthermore, our work only utilizes the confidence scores from the parser, which greatly simplifies the problem since additional statistical models (such as the ME models described above) are not needed. To the extent of our knowledge, no other works have studied the viability of inducing training examples from novel speech input.

4 Methodology

In this section, we describe the data set which was used as well as the approach which was taken to conduct our experiments.

4.1 Data Set

In order to be able to train the systems in our pipeline, a corpus was collected from 32 participants. The set of participants consisted of both males and females as well as people from a variety of nationalities and accents. Each participant was asked to read out a series of phrases on a screen at their own pace for 25 minutes. Participants contributed between 94 and 244 phrases each, with the average contribution being 150 phrases. Phrases consisted on average of 10 tokens.

Action	**Arguments**
$bring(x, y)$	Bring person y item x
$searchroom(x, y)$	Search room y for person x
$walk(x)$	Walk to location x
$walk_p(x)$	Walk to the office of person x

Figure 6: *The set of possible actions in our ontology. Template phrases for each of these actions were used in order to produce our corpus.* $walk_p$ *denotes the possessive walk action (i.e. possession of an office).*

Each phrase was randomly generated from a set of templates pertaining to a set of actions which could be expected of a robot in an office environment. The set of possible actions in our corpus may be seen in Figure 6, with examples for each action in Figure 7.

The action arguments in each template were randomly pulled from our system's ontology during phrase generation. In total, our ontology contains 11 people, 12 location, and 30 item atoms. Additionally, there are 42 generic noun predicates and 70 adjective predicates. Each noun predicate may be combined with up to

Action	**Template Example**	**Number of Templates**
bring(x,y)	I would like you to please bring x to y.	74
searchroom(x,y)	Find out if x is in y.	43
walk(x)	Would you please go to x.	39
$walk_p(x)$	Hurry and walk to x's office.	39

Figure 7: *Example templates for each action in our domain as well as the total number of unique templates per action. The arguments in each template were randomly sampled from the atoms in our ontology.*

Phrase	Semantic Form
See if Bob is in room thirty-four one eight.	$searchroom(bob, l3\_418)$
Deliver a green cup to Jane	$bring(a(\lambda x : i.(and(green(x), cup(x)))), jane)$
Run over to room three five one six.	$walk(l3\_516)$
Go to John's office.	$walk(the(\lambda x : l.(and(office(x), possesses(x, john)))))$

Figure 8: *Example phrases from our corpus with their semantic forms. Note how lambdas in the semantic forms denote expressions which may eventually be grounded using a knowledge base.*

two adjective predicates when sampling an item for a template, amounting to an additional

$$42 \cdot \left(\binom{72}{0} + \binom{72}{1} + \binom{72}{2} \right) = 110,418$$

theoretically possible item groundings. The templates were also used in order to generate an appropriate semantic form for each phrase, examples of which may be seen in Figure 8. Additionally, the generic US English dictionary accompanying Sphinx was augmented to include correct pronunciations for words in our corpus which it did not originally contain.

At completion, the corpus was split by participant into 8 folds for 8-fold cross validation. The number was chosen due to the very even split it allowed for between the 32 participants. Specifically, this split allowed for data from 28 participants to be used for training, 2 for validation, and 2 for testing in each fold.

4.2 ASR

In order to use the Sphinx ASR system, an in-domain language model was trained using our corpus of phrases. Additionally, the default Sphinx acoustic model was adapted with our collected recordings and their transcriptions using a tool provided by the Sphinx framework. Initially, the trained ASR system was used in order to generate a list of 1000 hypotheses for each phrase in the validation set. In order to mitigate the computational cost of re-ranking, however, this number was eventually reduced to 10 for our experiments. This decision was motivated by an analysis of the hypothesis lists, from which it was found that out of all the lists which contained the correct hypothesis for their phrase, 92% held it within the first 10 entries on the list. The results of this analysis may be seen in Figure 9.

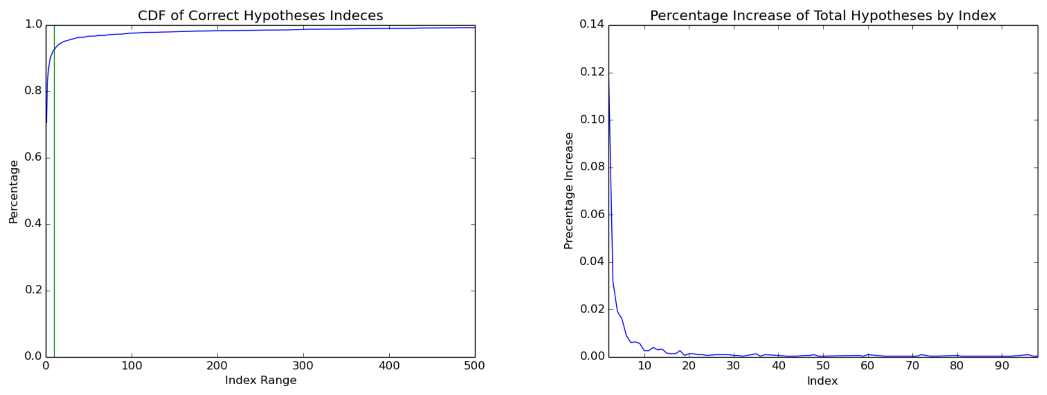


Figure 9: *Statistics of correct hypothesis index within hypothesis lists which contain the correct phrase. Left) The probability that the correct hypothesis lies within one of these lists as a function of the number of top results (out of 1000) kept (i.e. the cumulative distribution function). The green vertical line denotes an index of 10, which covers 92% of correct hypotheses. Right) The Percentage increase in number of correct hypotheses captured in the lists as the index of the last hypothesis maintained grows.*

4.3 Semantic Parser Re-ranking and Retraining

The parser was boot-strapped with an initial lexicon containing entries for one expression of each entity in our ontology. For example, there was one CCG entry in our lexicon for each person or office atom. Each entity in our ontology had multiple referents to them in our corpus (e.g. people atoms had both proper names and nicknames mapping to them), however, which necessitated that the parser learn the additional referents during training.

Having generated hypotheses lists for each phrase in the test set, the semantic parser was then used in order to re-rank the lists based on parse confidence scores produced by the top scoring parse of each hypothesis. It was found that using phrases from our corpus which contained more than 7 tokens results in prohibitive training and testing times for the parsing component of our system. Therefore, the experiments and results presented in this work were generated using only the phrases from our corpus which remained at or below this length. Pseudo code for the re-ranking procedure may be seen in algorithm 1. Additionally, figure 10 contains some examples of using the algorithm on our corpus.

Following the re-ranking step, the acoustic models were retrained in a variety of conditions in order to test the efficacy of training the acoustic models on new data. Specifically, the acoustic models were retrained with data from the validation set by taking the top hypothesis in each phrase's list and designating it as ground truth. This method was used for both the raw lists generated by the ASR system as well as for the re-ranked lists generated with semantic parsing, the results of which are discussed in section 5. Pseudo code for the re-training procedure may be seen in algorithm 2.

4.4 ASR and Parser Confidence Score Interpolation

In addition to simply re-ranking ASR hypothesis lists purely with the parser, we also experimented with interpolating the confidence scores produced by the ASR system for each hypothesis with the confidence score from its parse.

More formally, each ASR hypothesis in the n-best list is assigned a log-scale score s_a by the ASR system based on how close the hypothesis transcription matches the speech utterance given the acoustic model. Similarly, the parse produced for each hypothesis by the parser is given with a log-scale confidence score s_p.

In order to be able to interpolate the two scores, they are each first converted into probabilities by normalizing over the scores of all hypotheses in the n-best

Hypothesis	Parse Semantic Form	Parse Score	ASR Score
please take the pizza dan	walk(the(lambda 1:l.(and(possesses(1,jane),office(1)))))	-62.30	-1242160
please take the tea to dan	***bring(tea,dan)***	**-32.18**	-1242202
please take the tea to dan	bring(tea,dan)	-32.18	-1242288
please take the tea to dan	bring(tea,dan)	-32.18	-1242311
please take the pizza dan	walk(the(lambda 1:l.(and(possesses(1,jane),office(1)))))	-62.32	-1242348
please take the tea to dan	bring(tea,dan)	-32.18	-1242424
please take the peter dan	walk(the(lambda 1:l.(and(possesses(1,jane),office(1)))))	-62.29	-1242431
please take the pizza dan a	bring(coke,ben)	-46.41	-1242495
please take the tea to dan	bring(tea,dan)	-32.18	-1242533
please take the tea to dan a	None	$-\infty$	-1242544
please walk to professor smith a coffee	walk(l3_516)	-46.54	-476184
please walk to professor smith a coffee	walk(l3_516)	-45.40	-476254
please walk to professor smith a coffee	walk(l3_516)	-46.54	-476258
please walk to professor smith a coffee	walk(l3_516)	-45.40	-476272
please walk to professor smith a coffee	walk(l3_516)	-46.54	-476276
please walk to professor smith a coffee	walk(l3_516)	-46.54	-476346
please walk to professor smith a coffee	walk(l3_516)	-46.54	-476350
please walk to professor smith's office	**walk(the(lambda 1:l.(and(possesses(1,tom),office(1)))))**	**-38.55**	-476359
please walk to professor smith the coffee	walk(l3_516)	-46.54	-476378
please walk to professor smith the coffee	walk(l3_516)	-45.40	-476382
roll over to sam's office	*walk(the(lambda 1:l.(and(possesses(1,sam),office(1)))))*	-38.95	-28518
roll over to sam's office	walk(the(lambda 1:l.(and(possesses(1,sam),office(1)))))	-38.95	-28653
roll over to sam office	None	$-\infty$	-28920
roll over to sam office	None	$-\infty$	-29086
roll over to sam office	None	$-\infty$	-29221
a roll over to sam's office	searchroom(tom,the(lambda 1:l.(and(possesses(1,sam),office(1)))))	-28.65	-29259
roll over to to sam's office	**searchroom(tom,the(lambda 1:l.(and(possesses(1,sam),office(1)))))**	**-27.25**	-29311
roll over to to sam's office	None	$-\infty$	-29359
a roll over to sam's office	searchroom(tom,the(lambda 1:l.(and(possesses(1,sam),office(1)))))	-28.65	-29425
roll over to sam's office you	bring(coke,john)	-45.08	-29476

Figure 10: *3 example re-rankings using our algorithm. Each hypothesis list is ordered according to the original ASR output. Correct hypotheses are in italics while top re-ranked hypotheses chosen by our algorithm are bolded. All scores are in log-likelihood form. Semantic forms denoted by "None" represent phrases which the parser was not able to find a valid parse for and were given a score of $-\infty$. Top) A hypothesis with incorrect ASR and semantic form being replaced with one with both fields correct. Middle) A hypothesis with incorrect ASR and incorrect semantic form being replaced by one with correct ASR and more partially correct semantic form. Bottom) A hypothesis with correct ASR and semantic form being replaced by one with both fields incorrect.*

Algorithm 1 The hypothesis re-ranking algorithm employed in this work.

Input: A set of ASR hypotheses H, a set of their ASR scores A, a set of their parse scores P, and an interpolation value β.

Output: The top scoring hypothesis.

```
$n \leftarrow |H|$

$S_a \leftarrow \sum_{i=1}^{n} A[i]$
$S_p \leftarrow \sum_{i=1}^{n} P[i]$

\\Sets to contain normalized ASR and parse scores
$P_p \leftarrow \emptyset$
$P_s \leftarrow \emptyset$

for $i \leftarrow 1$ to $n$
    $P_p[i] \leftarrow \frac{P[i]}{S_p}$
    $P_s[i] \leftarrow \frac{A[i]}{S_a}$
end for

\\Compute interpolated score for each hypothesis.
for $i \leftarrow 1$ to $n$
    $H[i].score \leftarrow \beta \cdot P_p[i] + (1 - \beta) \cdot P_s[i]$
end for

\\Sort $H$ in descending order by score and return top hypothesis.
$sort(H)$

return $H[1]$
```

Algorithm 2 The acoustic model re-training algorithm.

Input: An acoustic model A, a set of speech recordings R, and a set of re-ranked hypothesis lists L, where each list pertains to a recording in R.

Output: The adapted acoustic model.

```
$T \leftarrow \emptyset$ \\Will contain the set of transcriptions for the recordings in $R$.
$n \leftarrow |R|$

for $i \leftarrow 1$ to $n$
    \\Use top hypothesis in list as transcription for recording.
    $T[i] \leftarrow L[i][1]$
end for

\\Adapt acoustic model using Sphinx framework.
$A' \leftarrow SphinxAdaptACModel(A, R, T)$

return $A'$
```

list:

$$P_s(h_i) = \frac{s_{a_i}}{\sum_{j=1}^{n} s_{a_j}}$$

$$P_p(h_i) = \frac{s_{p_i}}{\sum_{j=1}^{n} s_{p_j}}$$

After normalizing, we now interpolate the scores using a $\beta \in [0,1]$ value and re-rank based on this new interpolated score S_{h_i}:

$$S_{h_i} = \beta \cdot P_p(h_i) + (1 - \beta) \cdot P_s(h_i)$$

A value of β may be determined by evaluating different values of it on the validation set. The implementation and results of this are discussed further in section 5.4.

5 Results and Discussion

5.1 Evaluation Metrics

As described in section 2.1, the recognition performance of our system was evaluated by measuring WER in each testing condition. Additionally, the Recall @ 1 and Recall @ 5 accuracy scores were measured (i.e. the percentage of phrases for which the correct result was found within the top 1 or 5 hypotheses).

Because we would eventually like to use our system for issuing commands in a robotics domain, the end-to-end performance of the system was evaluated through two metrics over the computed semantic forms (i.e. parses). The first metric simply tested for complete predicate matches against the ground truth (full semantic form evaluation), while the second metric computed **recall** and **precision** values over predicates in the semantic forms (partial semantic form evaluation). Recall may be computed as the number of correct predicates found over the total number of correct predicates:

$$R = \frac{\#Correct\ predicates\ in\ hypothesis}{\#Correct\ predicates}$$

While precision may be computed as the number of correct predicates found over the total number of predicates retrieved:

$$P = \frac{\#Correct\ predicates\ in\ hypothesis}{\#Predicates\ in\ hypothesis}$$

One may trivially maximize recall by including every possible predicate in the hypothesis or maximize precision by only including a small number of them. Therefore, these two scores are generally used in tandem to compute what is known as an **F1** score, which is computed as the harmonic mean of the recall and precision:

$$F1 = \frac{2}{\frac{1}{R} + \frac{1}{P}}$$

For brevity, the results in our figures are denoted by two letter acronyms, where the first letter stands for the system used in the experiment (B for the

ASR baseline and *R* for parser re-ranking) and the second letter stands for the re-training condition (*B* for retraining with ASR results, *R* for retraining using re-ranking results, and *N* for no re-training). Scores between different conditions were compared for statistical significance using a Student's paired t-test.

5.2 Main Experiment Results

The results on all metrics for the validation set may be seen in Figure 11. It may be seen that the best scores achieved through re-ranking performed worse than the baseline ASR in the WER, Recall @ 1, and Recall @ 5 metrics. The full semantic form performance was roughly the same for both conditions. It may also be observed that system performance for precision, recall, and F1 scores improved when using re-ranking.

	WER	***R@1***	***R@5***	***SF***	***F1***	***P***	***R***
ASR	**15.79**	**53.28**	**66.39**	0.237	0.401	0.423	0.408
SemP	18.66	35.81	60.67	**0.239**	**0.528**	**0.565**	**0.542**

Figure 11: *Evaluation results on the validation set. The best score for each metric is bolded. From left to right, the metric labels denote word error rate, Recall @ 1, Recall @ 5, full semantic form, F1, precision, and recall metric scores respectively. ASR denotes the baseline score while SemP denotes semantic parser re-ranking performance.*

The results from the test set may be seen in Figure 12. Similarly to the validation set, it may be observed that the system performed worse in the WER, Recall @ 1, Recall @ 5 metrics. All three of these results were statistically significant with p-values < 0.05.

It should be noted, however, that although the system performed worse on average on the full semantic form evaluation, the difference in performance was not statistically significant (with a p-value of 0.12). Further, the system achieved higher precision, recall, and F1 performance when re-ranking without retraining

Re-Training	***Re-ranking***	***WER***	***R@1***	***R@5***	***SF***	***F1***	***R***	***P***
None	ASR	**14.55**	**55.31**	**72.47**	**0.334**	0.482	0.484	0.504
None	SemP	18.46	38.42	65.33	0.299	0.557	0.564	0.598
ASR	ASR	22.00	45.86	59.12	0.276	0.457	0.456	0.478
SemP	ASR	22.22	45.92	59.58	0.283	0.440	0.443	0.455
ASR	SemP	25.57	30.46	52.42	0.302	**0.569**	**0.581**	**0.604**
SemP	SemP	25.79	29.54	52.55	0.311	0.566	0.573	0.600

Figure 12: *Evaluation results on the test set. The best scoring system configuration score per metric is bolded. Re-training denotes the procedure through which re-training examples were generated (raw ASR hypothesis lists or SemP for semantic parser re-ranked lists) from the validation set, while Re-ranking denotes the hypothesis list ranking condition (baseline ASR or SemP parser re-ranking) used on the test set. From left to right, the metric labels denote word error rate, Recall @ 1, Recall @ 5, full semantic form, F1, recall, and precision metric scores respectively.*

all of which were statistically significant with p-values < 0.05.

The baseline ASR system without retraining achieved the highest scores over all other system conditions which used retraining on the WER, Recall @ 1, and Recall @ 5 metrics. This difference was statistically significant over all other system conditions. Although the baseline was also the best performing system in the full semantic form metric, the difference in scores was not statistically significant when compared to any method which used re-ranking.

It may also be observed that all conditions which used re-ranking performed better than all methods without it on the precision, recall, and F1 score metrics, all of which were statistically significant. There was no statistical significance, however, between the system which used re-ranking without retraining and those that did retrain.

These results show that our current system of re-ranking with semantic parsing has an adverse effect on metrics related to ASR transcription (WER, Recall @ 1, and Recall @ 5), but significantly improves on metrics related to semantic form (precision, recall, and F1). With this in mind, it is important to note that, in a robotics domain, it is of greater importance to maximize semantic understanding over speech transcription since the parsed semantic forms are what are ultimately translated to actions in a robotic system. Particularly, the improvement in F1 scores entails that our system would generally correctly infer more predicates than the baseline system without re-ranking. Any missing or incorrect predicates could then be disambiguated through dialogue as in (Thomason, Zhang, Mooney, & Stone, 2015). Under this framework, an increase in F1 score could result in a reduced number of exchanges being required during command disambiguation dialogue (since there would be less predicates to correct), improving the ease of use of the system. Therefore, although a decrease in transcription performance is not ideal, our system's significant improvement of partial semantic understanding is encouraging.

Our experiments have also shown that, currently, retraining has a significantly adverse effect on system performance. This is not surprising given that we re-train the acoustic model, whose performance is intimately dependent on the transcriptions fed to it. Because the WER currently goes up through our methods, it intuitively makes sense that re-training on this data decreases performance since the system is being re-trained on noisy data. It may be the case that once WER is improved through our system, then performance could be increased through re-training.

5.3 Error Analysis

The parser re-ranking algorithm's performance on the evaluated ASR transcription metrics could be due to a variety of reasons. It is possible that the parser could be overfitting the training data, leading it to generalize poorly to novel phrases. In order to investigate this, the parser training set performance was evaluated and compared to the performance on the ground truth ASR transcriptions from the validation and test sets, the results of which may be seen in Figure 13.

Training	**Validation**	**Test**
0.619	0.353	0.427

Figure 13: *Parsing performance using the full semantic form evaluation metric on the training set as well as on the ground truth transcriptions of the validation and test sets. These values show what parsing performance would be like given perfect speech recognition.*

From the figure, it may be observe that training set performance is much higher than that of the validation and test sets, which might indeed indicate some overfitting of the training data, something which could perhaps be alleviated through more training examples. This avenue could be very worthwhile to pursue given that roughly 86% of our corpus was left unused due to the 7 token limit imposed by computational cost, something which motivates the need for faster parsing methods.

Another possibility for why semantic parsing is not improving transcription performance might lie in the homogeneous quality of many of the n-best lists generated by the ASR system. It was found that many of the hypotheses in the computed lists (examples of which may be seen in Figure 10) generally only differ in a small fraction of tokens, with many hypotheses being identical. Although this characteristic can allow for corrections when the system makes minor mistakes, large mistakes can result in irrecoverable errors since the correct hypothesis will not be in the n-best list.

We also hypothesized that there could perhaps be greater room for improvements through re-ranking if longer phrases could be incorporated into our framework. In order to explore this, the ASR system was run on the entire corpus and the mean average WER was computed over all 8 folds by phrase length (in tokens), the results of which may be seen in Figure 14.

From looking at the figure, it may be inferred that WER does not in fact necessarily increase as a function of the number of tokens in the phrase. Therefore,

Phrase Length (in tokens)	Validation Set	Test Set
4	0.0	12.5
5	14.09	31.81
6	19.53	26.66
7	23.75	24.01
8	16.12	21.41
9	18.94	19.99
10	16.98	19.28
11	14.51	17.36
12	14.24	18.98
13	13.57	17.78
14	16.44	16.85
15	12.31	16.24
16	10.41	40.0
17	5.88	N/A

Figure 14: *The mean average WER across folds and over phrase lengths (in tokens). It may be observed that there is no significant trend (i.e. a monotonic increase) in WER performance as length increases. Note that there were no phrases with 17 tokens in the test set (denoted by N/A).*

it is not necessarily true that there would be more room for improvement through using the longer phrases in our corpus.

5.4 Confidence Score Interpolation

In order to test the efficacy of confidence score interpolation, system performance was evaluated over all metrics in the validation set starting at $\beta = 0$ and incrementing by 0.005 up to $\beta = 1$. The results of these experiments may be seen in Figures 15 and 17.

Through statistical analysis it was found that a β value of 0.505 decreased

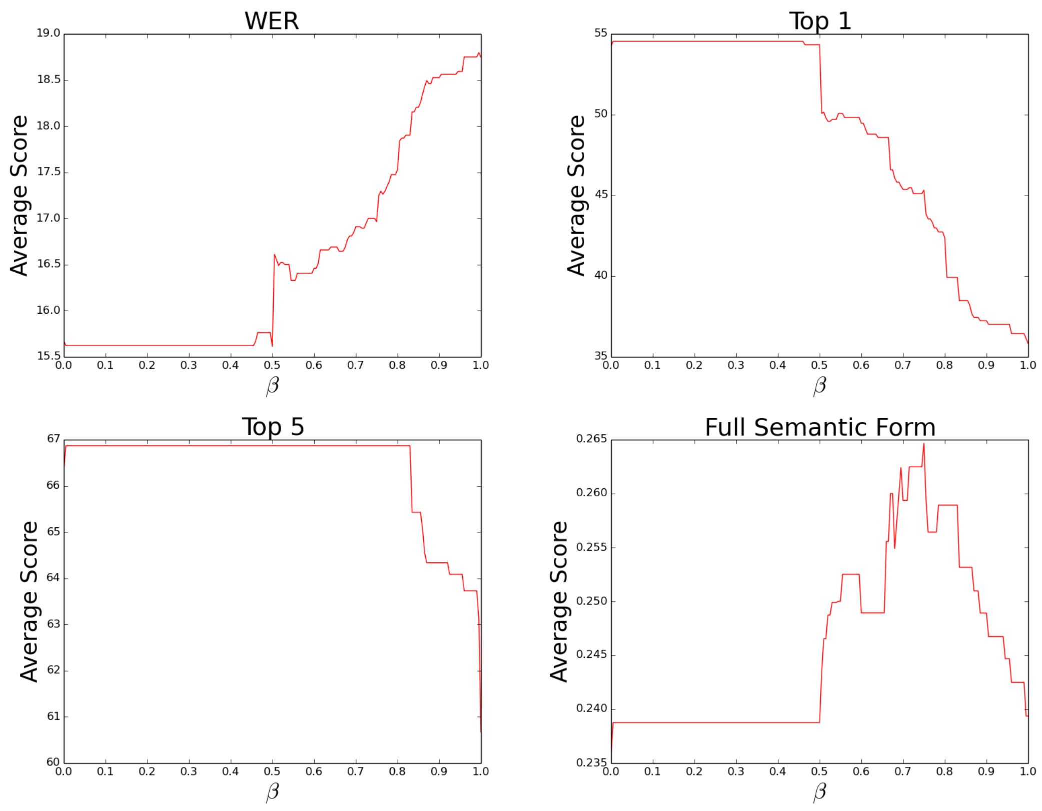


Figure 15: *WER, Recall @ 1, Recall @ 5, and full semantic form performance from interpolating ASR and parse confidence scores on the validation set. Note that $\beta = 0$ implies exclusive use of ASR confidence while $\beta = 1$ implies exclusive use of parser confidence.*

β	**F1**
0.0	0.401
0.865	0.537
1.0	0.528

Figure 16: *F1 performance on the validation set without re-ranking ($\beta = 0.0$), with the best performing interpolation value ($\beta = 0.865$), and with re-ranking without interpolation ($\beta = 1.0$). It may be observed that the best performance was achieved through interpolation, although the difference between $\beta = 0.865$ and $\beta = 1.0$ was not statistically significant ($p = 0.103$).*

WER performance by an insignificant amount (p-value of 0.16) while significantly increasing F1 performance (p-value of 0.004), something which motivated analyzing its performance on the test set. Additionally, it was found that F1 performance peaked at a value of $\beta = 0.865$. A comparison between F1 scores on the validation set at β values of 0.0, 0.865, and 1.0 may be seen in figure 16.

Through looking at the figures, it may be observed that the metrics associated with correctness in the text transcription (WER, Recall @ 1, and Recall @ 5) decrease with β while those associated with semantic form predicate correctness (precision, recall, and F1) increase. It should be noted that the full semantic form performance is not significantly affected by this increase.

The results from the same interpolation experiments on the test set may be seen in Figures 18 and 19. Although $\beta = 0.505$ produced a good middle ground between WER and F1 scores in the validation set, it was found that the decrease in WER performance in the test set under this value was significant (p-value of 0.03). This was also the case for $\beta = 0.865$ (p-value of .0.014). Further, it was found that all metrics peak at β values below 1.0, which suggests that retaining some signal from the ASR confidence is important for all metrics. Because these results are identical in statistical significance to the re-ranking case without interpolation (i.e. $\beta = 1.0$), the retraining experiments were not performed using interpolation and will likely be pursued in future work.

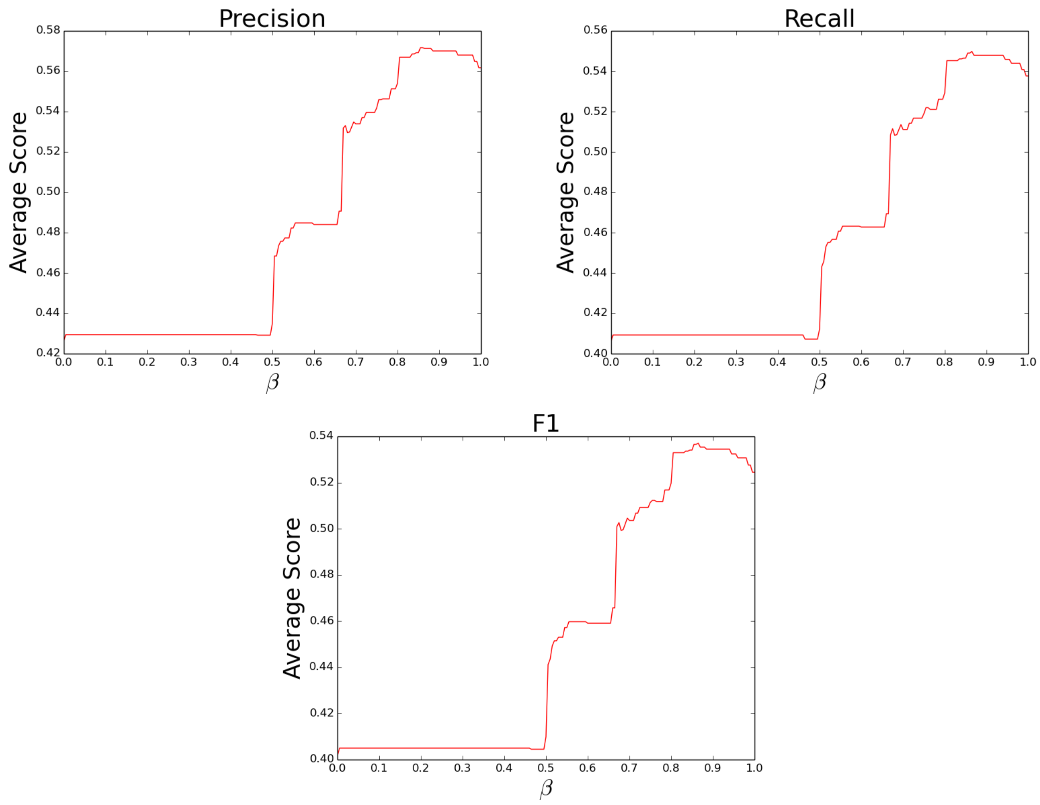


Figure 17: *Precision, recall, and F1 performance from interpolating ASR and parse confidence scores on the validation set. It may be observed that performance on these metrics begins to increase at $\beta = 0.5$, the point at which more weight is given to parse confidence scores over ASR scores.*

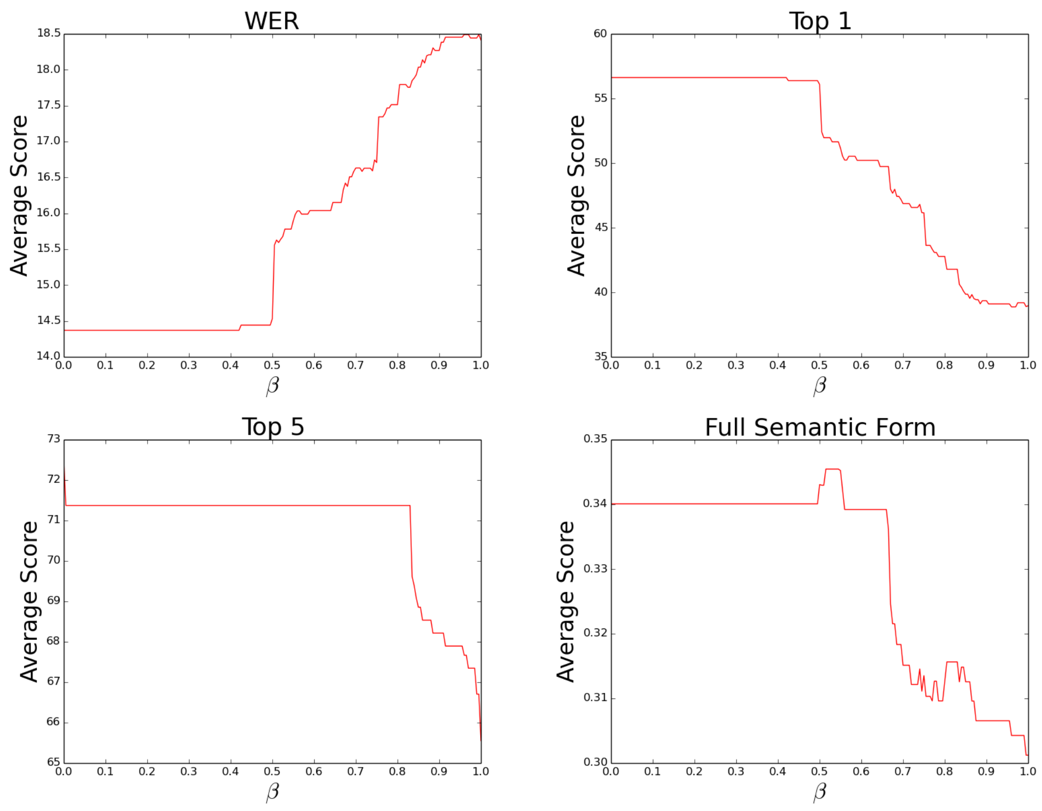
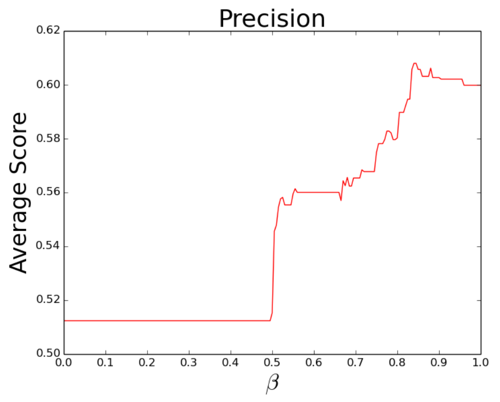
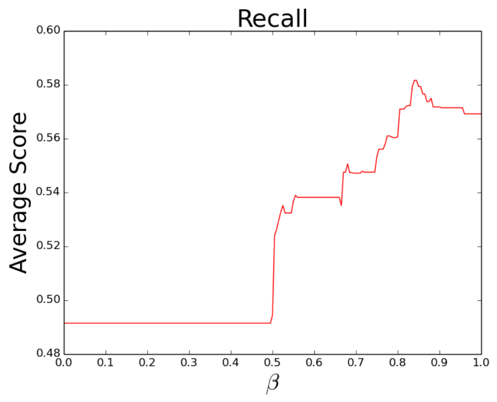


Figure 18: *WER, Recall @ 1, Recall @ 5, and full semantic form performance from interpolating ASR and parse confidence scores on the test set. Note that all 4 metrics begin to experience a drop in performance after a high enough β value.*





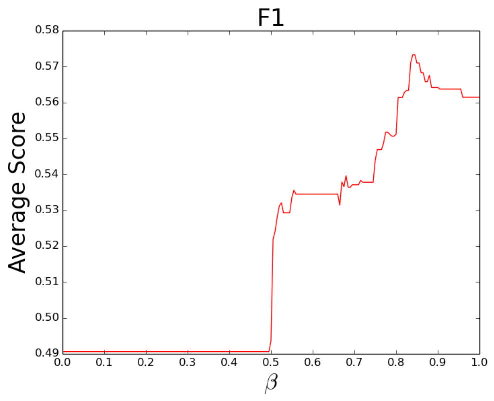


Figure 19: *Precision, recall, and F1 performance from interpolating ASR and parse confidence scores on the test set. Similarly to the validation set, all three metrics begin to experience an increase in performance after a value of* $\beta = 0.5$.

6 Future Work

In this section we briefly describe a few possible avenues that may be pursued for future work.

6.1 Deep Learning Approaches

Because we would eventually want to use an ASR system for issuing commands in robotic tasks, it is not strictly necessary that the text transcription of a speech utterance be generated. With this in mind, one possible avenue of future work could be to train an end to end system which can take speech utterances as input and output semantic forms pertaining to them.

There has been work done in end to end speech recognition in which an LSTM neural network was trained to transcribe speech to text without needing to use an intermediate phonetic representation (i.e. an explicit acoustic model) (Graves & Jaitly, 2014; Xiong et al., 2016). Perhaps such a system could be extended and trained to generate semantic forms rather than text transcriptions.

As mentioned in section 5.3, our current system is unable to use phrases in our corpus which are longer than 7 tokens in length due to computational cost. Therefore, it would be highly desirable to find methods of performing effective semantic parsing that incur lower computational cost. Recent works such as (Misra & Artzi, 2016) have shown promising findings using neural neural networks to produce results comparable to CKY parsers with a lower number of operations.

In future work, we may pursue the use of similar neural network methods for the components of our pipeline. Particularly, if we replaced both the ASR and semantic parser with neural models, then the entire system could perhaps be trained end-to-end (i.e. training on pairs of speech utterance input and semantic form output). In order to properly work with the relatively small size of our data set (compared to traditional deep learning data sets), we could use pre-trained models which we would then fine-tune with our data.

6.2 Discriminative Re-ranking

Our current system re-ranks recognition hypotheses using parse confidence scores. One possible approach that may be tried in order to improve the system could be to train a discriminative re-ranker for the re-ranking task. Similarly to (Collins & Koo, 2005), such a model could take some feature representation from candidate hypotheses' ASR transcriptions and parses along with the input speech utterances and produce a new set of rankings.

6.3 Incremental Learning Through Human-Robot Interaction

One of the desired applications for our system is to be able to deploy it in a robotics environment where the agent can incrementally learn to participate in dialogue with users. Similarly to (Thomason et al., 2015), the system would be boot-strapped with an initial lexicon for its parser and speech recognition system. With this initial knowledge, the agent would iteratively learn more robust models through dialogue interactions with users. As mentioned in section 5.2, this avenue of future work could prove particularly fruitful given the significant increase in partial semantic form performance that our system experiences through parser re-ranking. Particularly, if there is an increase in the correct number of predicates in the computed semantic form (i.e. F1 score), then the average dialogue length could be reduced since the system would have less parameters to disambiguate in given commands.

Additionally, our work did not explore the use of context or reasoning for biasing semantic parsing re-ranking. There are certain cases where semantic forms with the same predicates but different arguments (e.g. $bring(marker, tom)$ and $bring(marker, dan)$) can have very similar or identical parser confidence scores. If a dialogue system were to be implemented, then the incorporation of common sense reasoning, similar to what is used in (Zhang & Stone, 2015), could prove useful for disambiguating between different arguments or predicates.

7 Conclusion

Although our methods have been shown to have an adverse effect on WER, which is the main evaluation metric for speech recognition, we have shown that our system can significantly improve precision, recall, and F1 scores in the predicate lists of the final semantic forms of hypotheses. We believe that these findings are promising since we are ultimately interested in the end to end performance of the system (i.e. final semantic forms). Although the full semantic form performance is not significantly affected, we believe that our findings can serve as a proof of concept for the potential improvements which semantic parsing could provide a pipeline which uses speech as input, particularly one in which dialogue could disambiguate the parameters of issued commands.

Acknowledgements

I would like to thank Dr. Raymond Mooney for taking me on as his student and for advising me through this first research experience. I am incredibly grateful for his guidance and patience as I have gone through this process. Dr. Peter Stone for introducing me to the field of AI and for serving as my second reader. I would also like to thank Dr. Vijaya Ramachandran for serving on my committee. Additionally, I would like to thank Jesse Thomason for his mentorship, his help and input have been invaluable to this work. Finally, I'd like to thank Aishwarya Padmakumar for her continued input and assistance throughout.

References

Ananthakrishnan, S., & Narayanan, S. (2007). Improved Speech Recognition Using Acoustic and Lexical Correlates of Pitch Accent in a N-best Rescoring Framework. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on* (Vol. 4, pp. IV–873).

Artzi, Y., & Zettlemoyer, L. (2013a). UW SPF: The University of Washington Semantic Parsing Framework. *arXiv preprint arXiv:1311.3011*.

Artzi, Y., & Zettlemoyer, L. (2013b). Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions. *Transactions of the Association for Computational Linguistics*, *1*, 49–62.

Basili, R., Bastianelli, E., Castellucci, G., Nardi, D., & Perera, V. (2013). Kernel-Based Discriminative Re-ranking for Spoken Command Understanding in HRI. In *AI* IA 2013: Advances in Artificial Intelligence* (pp. 169–180). Springer.

Church, A. (1940). A Formulation of the Simple Theory of Types. *The Journal of Symbolic Logic*, *5*(02), 56–68.

Collins, M., & Koo, T. (2005). Discriminative Reranking for Natural Language Parsing. *Computational Linguistics*, *31*(1), 25–70.

Erdogan, H., Sarikaya, R., Chen, S. F., Gao, Y., & Picheny, M. (2005). Using Semantic Analysis to Improve Speech Recognition Performance. *Computer Speech & Language*, *19*(3), 321–343.

Forney, G. D. (1973). The Viterbi Algorithm. *Proceedings of the IEEE*, *61*(3), 268–278.

Graves, A., & Jaitly, N. (2014). Towards End-To-End Speech Recognition with Recurrent Neural Networks. In *ICML* (Vol. 14, pp. 1764–1772).

Jurafsky, D., & Martin, J. H. (2009). *Speech and Language Processing (2Nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Lamere, P., Kwok, P., Gouvea, E., Raj, B., Singh, R., Walker, W., … Wolf, P. (2003). The CMU SPHINX-4 Speech Recognition System. In *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2003), Hong Kong* (Vol. 1, pp.

2–5).

Lamere, P., Kwok, P., Walker, W., Gouvêa, E. B., Singh, R., Raj, B., & Wolf, P. (2003). Design of the CMU Sphinx-4 Decoder. In *INTERSPEECH.*

Liang, P., & Potts, C. (2015). Bringing Machine Learning and Compositional Semantics Together. *Annu. Rev. Linguist.*, *1*(1), 355–376.

Misra, D. K., & Artzi, Y. (2016). Neural Shift-Reduce CCG Semantic Parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing.*

Morbini, F., Audhkhasi, K., Artstein, R., Van Segbroeck, M., Sagae, K., Georgiou, P., … Narayanan, S. (2012). A Reranking Approach for Recognition and Classification of Speech Input in Conversational Dialogue Systems. In *Spoken Language Technology Workshop (SLT), 2012 IEEE* (pp. 49–54).

Peng, F., Roy, S., Shahshahani, B., & Beaufays, F. (2013). Search Results Based N-best Hypothesis Rescoring with Maximum Entropy Classification. In *ASRU* (pp. 422–427).

Rabiner, L. R. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, *77*(2), 257–286.

Rosenfeld, R. (2000). Two Decades of Statistical Language Modeling: Where Do We Go from Here?

Singh, R., Warmuth, M. K., Raj, B., & Lamere, P. (2003). Classification with Free Energy at Raised Temperatures. In *INTERSPEECH.*

Steedman, M., & Baldridge, J. (2011). Combinatory Categorial Grammar. *Non-Transformational Syntax: Formal and Explicit Models of Grammar. Wiley-Blackwell*.

Thomason, J., Zhang, S., Mooney, R., & Stone, P. (2015). Learning to Interpret Natural Language Commands Through Human-Robot Dialog. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI).*

Twiefel, J., Baumann, T., Heinrich, S., & Wermter, S. (2014). Improving Domain-independent Cloud-Based Speech Recognition with Domain-Dependent Phonetic Post-Processing. In *AAAI* (pp. 1529–1536).

Walker, W., Lamere, P., Kwok, P., Raj, B., Singh, R., Gouvea, E., … Woelfel, J. (2004). Sphinx-4: A Flexible Open Source Framework for Speech Recognition.

Xiong, W., Droppo, J., Huang, X., Seide, F., Seltzer, M., Stolcke, A., … Zweig, G. (2016). Achieving Human Parity in Conversational Speech Recognition. *arXiv preprint arXiv:1610.05256*.

Young, S. J., Russell, N., & Thornton, J. (1989). *Token Passing: A Simple Conceptual Model for Connected Speech Recognition Systems*. Cambridge University Engineering Department Cambridge, UK.

Zechner, K., & Waibel, A. (1998). Using Chunk Based Partial Parsing of Spontaneous Speech in Unrestricted Domains for Reducing Word Error Rate in Speech Recognition. In *Proceedings of the 17th International Conference on Computational Linguistics-Volume 2* (pp. 1453–1459).

Zhang, S., & Stone, P. (2015, January). CORPP: Commonsense Reasoning and Probabilistic Planning, as Applied to Dialog with a Mobile Robot. In *Proceedings of the 29th Conference on Artificial Intelligence (AAAI).*