To Appear In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-16), Berlin, Germany 2016.

Improved Semantic Parsers For If-Then Statements

I. Beltagy
The University of Texas at Austin
beltagy@cs.utexas.edu

Chris Quirk
Microsoft Research
chrisq@microsoft.com

Abstract

Digital personal assistants are becoming both more common and more useful. The major NLP challenge for personal assistants is machine understanding: translating natural language user commands into an executable representation. This paper focuses on understanding rules written as If-Then statements, though the techniques should be portable to other semantic parsing tasks. We view understanding as structure prediction and show improved models using both conventional techniques and neural network models. We also discuss various ways to improve generalization and reduce overfitting: synthetic training data from paraphrase, grammar combinations, feature selection and ensembles of multiple systems. An ensemble of these techniques achieves a new state of the art result with 8% accuracy improvement.

1 Introduction

The ability to instruct computers using natural language clearly allows novice users to better use modern information technology. Work in semantic parsing has explored mapping natural language to some formal domain-specific programming languages such as database queries (Woods, 1977; Zelle and Mooney, 1996; Berant et al., 2013; Andreas et al., 2016; Yin et al., 2016), commands to robots (Kate et al., 2005), operating systems (Branavan et al., 2009), and spreadsheets (Gulwani and Marron, 2014). This paper explores the use of neural network models (NN) and conventional models for semantic parsing. Recently approaches using neural networks have shown great improvements in a number of areas such as parsing (Vinyals et al., 2015), machine translation (Devlin et al., 2014), and image captioning (Karpathy and Fei-Fei, 2015). We are among the first to apply neural network methods to semantic parsing tasks (Grefenstette et al., 2014; Dong and Lapata, 2016).

There are several benchmark datasets for semantic parsing, the most well known of which is Geoquery (Zelle and Mooney, 1996). We target an If-Then dataset (Quirk et al., 2015) for several reasons. First, it is both directly applicable to the end-user task of training personal digital assistants. Second, the training data, drawn from the site http://ifttt.com, is comparatively quite large, containing nearly 100,000 recipe-description pairs. That said, it is several orders of magnitude smaller than the data for other tasks where neural networks have been successful. Machine translation datasets, for instance, may contain billions of tokens. NN methods appear "data-hungry". They require larger datasets to outperform sparse linear approaches with careful feature engineering, as evidenced in work on syntactic parsing (Vinyals et al., 2015). This makes it interesting to compare NN models with conventional models on this dataset.

As in most prior semantic parsing attempts, we model natural language understanding as a structure prediction problem. Each modeling decision predicts some small component of the target structure, conditioned on the whole input and all prior decisions. Because this is a real-world task, the vocabulary is large and varied, with many words appearing only rarely. Overfitting is a clear danger. We explore several methods to improve generalization. A classic method is to apply feature selection. Synthetic data generated by paraphrasing helps augment the data available. Adjusting the conditional structure of our model also makes sense, as does creating ensembles of the best performing approaches.

An ensemble of the resulting systems achieves a new state-of-the-art result, with an absolute improvement of 8% in accuracy. We compare the performance of a neural network model with logistic regression, and explore in detail the contribution of each of them, and why the logistic regression is performing better than the neural network.

2 Related Work

2.1 Semantic Parsing

Semantic parsing is the task of translating natural language to a meaning representation language that the machine can execute. Various semantic parsing tasks have been proposed before, including querying a database (Zelle and Mooney, 1996), following navigation instructions (Chen, 2012), translating to Abstract Meaning Representation (AMR) (Artzi et al., 2015), as well as the If-Then task we explore. Meaning representation languages vary with the task. In database queries, the meaning representation language is either the native query language (e.g. SQL or Prolog), or some alternative that can be deterministically transformed into the native query language. To follow navigation instructions, the meaning representation language is comprised of sequences of valid actions: turn left, turn right, move forward, etc. For parsing If-Then rules, the meaning representation is an abstract syntax tree (AST) in a very simple language. Each root node expands into a "trigger" and "action" pair. These nodes in turn expand into a set of supported triggers and actions. We model these trees as an (almost) context free grammar[1] that generates valid If-Then tasks.

A number of semantic parsing approaches have been proposed, but most fit into the following broad divisions. First, approaches driven by Combinatory Categorical Grammar (CCG) have proven successful at several semantic parsing tasks. This approach is attractive in that it simultaneously provides syntactic and semantic parses of a natural language utterance. Syntactic structure helps constrain and guide semantic interpretation. CCG relies heavily on a *lexicon* that specifies both the syntactic category and formal semantics of each lexical item in the language. In many instantiations, the lexicon is learned from the training data (Zettlemoyer and Collins, 2005) and grounds directly in the meaning representation.

Another approach is to view the semantic parsing task as a machine translation task, where the source language is natural language commands and the target language is the meaning representation. Several approaches have applied standard machine translation techniques to semantic parsing (Wong and Mooney, 2006; Andreas et al., 2013; Ratnaparkhi, 1999) with successful results.

More recently, neural network approaches have been developed for semantic parsing, and especially for querying a database. A neural network is trained to translate the query and the database into some continuous representation then use it to answer the query (Andreas et al., 2016; Yin et al., 2016).

2.2 If-Then dataset

We use a semantic parsing dataset collected from `http://ifttt.com`, first introduced in Quirk et al. (2015). This website publishes a large set of *recipes* in the form of If-Then rules. Each recipe was authored by a website user to automate simple tasks. For instance, a recipe could send you a message every time you are tagged on a picture on Facebook. From a natural language standpoint, the most interesting part of this data is that alongside each recipe, there is a short natural language description intended to name or advertise the task. This provides a naturalistic albeit often noisy source of parallel data for training semantic parsing systems. Some of these descriptions faithfully represent the program. Others are underspecified or suggestive, with many details of the recipe are not uniquely specified or omitted altogether. The task is to predict the correct If-Then code given a natural language description.

As for the code, If-Then statements follow the format

```
If TriggerChannel.
       TriggerFunction (args)
Then ActionChannel.
       ActionFunction (args)
```

Every If-Then statement has exactly one *trigger* and one *action*. Each trigger and action consist of both a *channel* and a *function*. The channel represents a connection to a service, website, or device

[1] Information at the leaves of the action may use parameters drawn from the trigger. For instance, consider a rule that says "text me the daily weather report." The trigger is a new weather report, and the action is to send an SMS. The contents of that SMS are generated by the trigger, which is no longer context free.

(e.g., Facebook, Android, or ESPN) and provides a set of functions relevant to that channel. Finally, each of these functions may take a number of arguments: to receive a trigger when it becomes sunny, we need to specify the location to watch. The resulting dataset after cleaning and separation contains 77,495 training recipes, 5,171 development recipes and 4,294 testing recipes.

2.3 Semantic parsing for If-Then rules

Both CCG and MT-inspired approaches assume a fairly strong correspondence between the words in the natural language request and the concepts in the meaning representation. That is, most words in the description should correspond to some concept in the code, and most concepts in the code should correspond to some word in the description. However, prior work on this dataset (Quirk et al., 2015) found that this strong correspondence is often missing. The descriptions may mention only the most crucial or interesting concepts; the remainder of the meaning representation must be inferred from context. The best performing methods focused primarily on generating well-formed meaning representations, conditioning their decisions on the source language.

Quirk et al. (2015) proposed two models that rely on a grammar to generate all valid ASTs. The first model learns a simple classifier for each production in the grammar, treating the sentence as a bag of features. No alignment between the language and meaning representation is assumed. The second method attempts to learn a correspondence between the language and the code, jointly learning to select the correct productions in the meaning representation grammar. Although the latter approach is more appealing from a modeling standpoint, empirically it doesn't perform substantially better than the alignment-free model. Furthermore the alignment-free model is much simpler to implement and optimize. Therefore, we build upon the alignment-free approach.

2.4 Neural Networks

Neural network approaches have recently made great strides in several natural language processing tasks, including machine translation and dependency parsing. Partially these gains are due to better generalization ability. Until recently, the NLP community leaned heavily on feature-rich approaches that allow models to learn complex relationships from data. However, important features, such as indicator features for words and phrases, were often very sparse. Furthermore, the best systems often relied on manually-induced feature combinations (Bohnet, 2010). Multi-layer neural networks have several advantages. Words (or, more generally, features) are first embedded into a continuous space where similar features land in nearby locations; this helps lead to lexical generalization. The additional hidden layers can model feature interactions in complex ways, obviating the need for manual feature template induction. Feed-forward neural networks with relatively simple structure have shown great gains in both dependency parsing (Chen and Manning, 2014) and machine translation (Devlin et al., 2014) without the need for complex feature templates and large models. Our NN models here are inspired by these effective approaches.

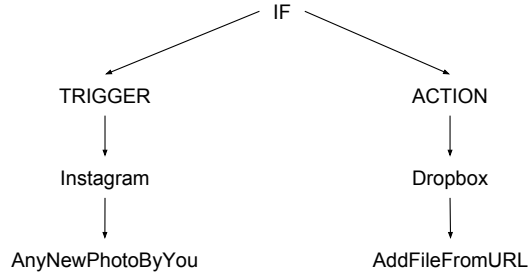

Figure 1: Derivation tree of If-Then statement of the recipe *Autosave your Instagram photos to Dropbox*. Arguments of the functions AnyNewPhotoByYou and AddFileFromURL are ignored.

3 Approach

We next describe the details of how If-Then recipes are constructed given natural language descriptions. As in prior work, we treat semantic parsing as a structure prediction task. First we describe the structure and features of the model, then expand on the details of inference.

3.1 Grammar

Along the lines of Quirk et al. (2015), we build a context-free grammar baseline. This grammar generates only well-formed meaning representations. In the case of this dataset, meaning representations always consist of a root production with two children: a trigger and an action. Both trigger and action first generate a channel, then a function matching that action. Optionally we may also generate the arguments of these functions; we do not

evaluate these selections as they are often idiosyncratic and specific to the user. For example, the recipe *Autosave your Instagram photos to Dropbox* has the following meaning representation:

```
IF Instagram.AnyNewPhotoByYou
THEN Dropbox.AddFileFromURL (
  FileURL={SourceUrl},
  FileName={Caption},
  DropboxFolderPath=IFTTT/Instagram
)
```

If we ignore the function arguments, the resulting meaning representation is:

```
IF Instagram.AnyNewPhotoByYou
THEN Dropbox.AddFileFromURL
```

This examples shows also that most of the function arguments are not crucial for the representation of the If-Then statement.[2]

The grammar we use has productions corresponding to every channel and every function. Figure 1 shows an example derivation tree D. This grammar consists of 892 productions: 128 trigger channels, 487 trigger functions, 99 action channels and 178 action functions.[3]

3.2 Model

Our goal is to learn a model of derivation trees D given a natural sentences S. To predict the derivation for a sentence, we seek the derivation D with maximum probability given the sentence $P(D|S)$.

For the purposes of modeling, we prefer to work with sequences rather than trees. Given a derivation tree D, we transform it into a sequence of productions $R(D) = r_1, \ldots, r_n$ by a top-down, left-to-right tree traversal: r_1 is the top-most production, and r_n is the bottom right production. The sentence S is represented as a set of features $f(S)$.

The derivation score $P(D|S)$ is a function of the productions of D and those features $f(S)$:

$$P(D|S) = \prod_{r_i \in R(D)} P(r_i|r_1, \ldots, r_{i-1}, f(S)) \quad (1)$$

The score of a derivation tree given the sentence is the product of probabilities of its productions.

[2]Arguments are still important for a few If-Then recipes. For instance, in *If there is snow tomorrow send a notification*, "snow" is an argument to the function Tomorrow'sForecastCallsFor. We are not handling such cases in this work.

[3]For this task, it is possible to model the programs as a 4-tuple, but using the grammar approach allows us to port the same technique to other semantic parsing tasks.

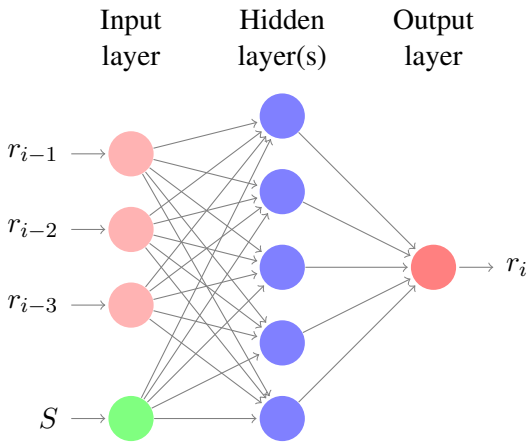

Figure 2: Architecture of the feed-forward neural networks used in this paper. When predicting rule r_i, the prior rules and the whole sentence are used as input. Separate parameters are learned for each position i.

The probability of selecting production r_i given the sentence S is dependent on the features of the sentence as well as the previous productions $r_1, \ldots, r_{i-1}$; namely, all those productions that are above and to the left of the current production. Conditioning on previous productions helps predicting the next one because it captures the conditional dependencies between the productions of the derivation tree, an improvement over prior work (Quirk et al., 2015). In particular, we can model which combinations of triggers and actions are more compatible, both function and channel.

3.3 Training

To learn the derivation score $P(D|S)$, we need to learn probability of productions $P(r_i|r_1, \ldots, r_{i-1}, f(S))$. We learn this probability using a multiclass classifier where the output classes are the possible productions in the grammar. The classifier is trained to predict the next production given previous productions and the sentence features.

Each sentence S is represented with a sparse feature vector $f(S)$. We used a simple set of features: word unigrams and bigrams, character trigrams, and Brown clusters (Liang, 2005). Each sentence is represented as a large sparse k-hot vector, where k is the number of features representing S, $|f(S)|$. We use a simple one-hot representation of prior rules.

For training, we explored two approaches: a standard logistic regression classifier, and a feed

forward neural network classifier. [4] As for network structure, we evaluated models with either one or two 200-dimensional hidden layers (with sigmoid activation function) followed by a softmax output layer to produce a probability for each production. We tried more than two hidden layers and larger hidden layer size, but the results were similar or worse likely because training becomes more difficult. Figure 2 shows the architecture of the network we use. For training, we used a variant of stochastic gradient descent called RMSprop (Dauphin et al., 2015) that adjusts the learning rate for each parameter adaptively, along with a global learning rate of 1^{-3}. The minibatch size was 100, with dropout regularization for hidden layers at 0.5 along with an L2 regularizer with weight 0.005. Each of these parameters were tuned on the validation set, though we found learning to be robust to minor variations in these parameters. All of the neural networks were implemented with Theanets (Johnson, 2015).

Note that history features $r_1, \ldots, r_{i-1}$ in classifier training are always correct. The model is akin to a MEMM, rather than a CRF. We make this simplifying assumption for tractability, like many neural network approaches (Devlin et al., 2014).

3.4 Inference

When, at test time, we are given a new sentence, we would like to infer its most probable derivation tree D. Classifiers trained as in the prior section give probability distributions over productions given the sentence and all prior productions $P(r_i|r_1, \ldots, r_{i-1}, f(S))$. Were the distribution to be context free, we could rely on algorithms similar to Earley parsing (Earley, 1970) to find the max derivation. However, the dependency on prior productions breaks the context free assumption. Therefore, we resort to approximate inference, namely beam search. Each partial hypothesis is grouped into a beam based on the number of productions it contains; we use a beam width of 8, and search for the highest scoring hypothesis.

4 Improving generalization

The data set we use for training and testing is primarily English but contains a broad vocabulary as well as many sentences from other languages such as Chinese, Arabic, and Russian. Thus, a seemingly large dataset of nearly eighty thousand examples is likely to suffer from overfitting. In this section, we discuss a few attempts to improve generalization in the sparse data setting.

4.1 Synthetic data using paraphrases

Arguably the best, though most expensive, way to reduce overfitting is to collect more training data. In our case, the training data available is limited and difficult to create. We propose to augment the training data in an automatic though potentially noisy way by generating synthetic training pairs.

The main idea is that two semantically equivalent sentences should have the same meaning representation. Given an existing training pair, replacing the pair's linguistic description with a paraphrase leads to a new synthetic training pair. For example, a recipe like *Autosave your Instagram **photos** to Dropbox* can be paraphrased to *Autosave your Instagram **pictures** to Dropbox* while retaining the meaning representation:

```
IF Instagram.AnyNewPhotoByYou
THEN Dropbox.AddFileFromURL.
```

We first explore paraphrases using WordNet synonyms. Every word in the sentence can be replaced by one of its synonyms that is picked randomly (a word is a synonym of itself). For words with multiple senses, we group all synonyms of all senses, then retain only those synonyms already in the vocabulary of the training data. This has two advantages. First, we do not increase the vocabulary size and therefore avoid overfitting. Second, this acts as a simple form of word sense disambiguation. This adds around 50,000 additional training examples.

Next, we consider augmenting the data using the Paraphrase Database (Ganitkevitch et al., 2013). Each original description is converted into a lattice. The original word at each position is left in place with a constant score. For each word or phrase in the description found PPDB, we add one arc for each paraphrase, parameterized by the PPDB score of that phrase. The resulting lattice represents many possible paraphrases of the input. We select at most 10 diverse paths through this lattice using the method of Gimpel et al. (2013).[5] This adds around 470,000 training examples.

[4] We tried the sequence-to-sequence model with LSTMs (Sutskever et al., 2014) to map word sequence to the derivation tree productions, but the results were always lower than the feed forward network. This is probably because of the lack of enough training data.

[5] We use a trigram language model, and a weight of 4.

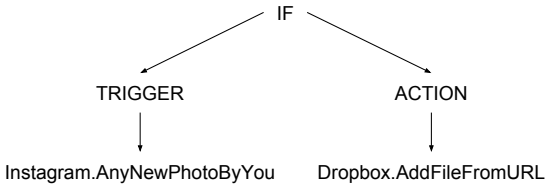

Figure 3: Derivation tree of IFTTT statement of the recipe *Autosave your Instagram photos to Dropbox* using the second grammar.

4.2 Alternative grammar formulation

We rely on a grammar to generate all valid meaning representations and learn models over the productions of this grammar. Different factorizations of the grammar lead to different model distributions. Our primary grammar is described in Section 3.1. A second, alternate grammar formulation has fewer levels but more productions: it combines the channel and function into a single production, in both the trigger and the action. Figure 3 shows an example derivation tree using this grammar. The size of this grammar is 780 productions (552 triggers + 228 actions).

An advantage of this grammar is that it cannot assign probability mass to invalid ASTs, where the function is not applicaable to the channel. On the other hand, this grammar likely does not generalize as well as the first grammar. The first grammar effectively has much more data about each channel, which likely improves accuracy. Function predictions can condition on hopefully accurate channel predictions. It can also benefit from the fact that some function names are shared among channels. From that perspective, the second grammar has fewer training instances for each outcome.

4.3 Feature selection

The training set contains approximately 77K training examples, yet the number of distinct features types (word unigrams and bigrams, character trigrams, Brown clusters) is approximately 230K. Only 80K features occur in the training set more than once. This ratio suggests overfitting may be a major issue.Feature selection likely can improve these issues. We used only simple count cutoffs, including only features that occur in the training set more than once and more than twice. Including features that occur more than once led to improvements in practice.

4.4 Ensemble

Finally, we explore improving generalization by building ensembles of multiple systems. Even if systems overfit, they likely overfit in different ways. When systems agree, they are likely to agree on the correct answer. Combining their results will suffer less from overfitting. We use simple majority voting as an ensemble strategy, resolving ties in an arbitrary but deterministic way.

5 Evaluation

We evaluate the performance of the systems by providing the model with descriptions unseen during training. Free parameters of the models were tuned using the development set. The separation of data into training, development, and test follows Quirk et al. (2015). Two evaluation metrics are used: accuracy on just channel selection and accuracy of both channel and function.

Two major families of approaches are considered: a baseline logistic regression classifier from scikit-learn (Pedregosa et al., 2011), as well as a feed-forward neural network. We explore a number of variations, including feature selection and grammar formulation.

5.1 Comparison systems

Our default system was described in section 3, not including improvements from section 4 unless otherwise noted. The grammar uses the primary formulation from section 3.1. Neural network models use a single hidden layer by default; we also explore two hidden layers.

We evaluate two approaches for generating synthetic data. The first approach, leaning primarily on WordNet to generate up to one paraphrase for each instance, is labeled WN. The second approach using Paraphrase Database to generate up to ten paraphrases is labeled PPDB.

The Alternate grammar line uses the section 4.2 grammar, and otherwise default configurations (no synthetic data, single hidden layer for NN).

Feature selection again uses the default configuration, but uses only those features that occurred more than once in the training data.

Finally we explore ensembles of all approaches. First, we combine all variations within the same model family; next, we bring all systems together. To evaluate the impact of individual systems, we also present results with specific systems removed.

System	Channel accuracy		Full tree accuracy	
	NN	LR	NN	LR
Quirk et al. (2015) w/o alignment	-	46.30	-	33.00
Quirk et al. (2015) with alignment	-	47.40	-	34.50
Default configurations	52.93	53.73	39.66	41.87
Two hidden layers	46.81	-	32.77	-
No hidden layers	50.05	-	38.47	-
Synthetic data (WN)	52.45	53.68	38.64	41.55
Synthetic data (PPDB)	51.86	52.96	38.86	40.63
Alternate grammar	50.09	52.42	39.10	41.15
Feature selection	52.91	53.31	39.29	41.34
Ensemble of systems above	53.98	53.73	41.06	41.85
Ensemble NN + LR	54.31		**42.55**	
Ensemble NN + LR (w/o alternate grammar)	**54.38**		41.90	
Ensemble NN + LR (w/o synthetic data)	53.98		42.41	

Table 1: Accuracy of the Neural Network (NN) and Logistic Regression (LR) implementations of our system with various configurations. Channel-only and full tree (channel+function) accuracies are listed.

5.2 Results

Table 1 shows the accuracy of each evaluated system, and Table 2 explores system performance on important subsets of the data. The first columns present accuracy of just the channel, and the last columns present the channel and the function together (the full derivation). We achieve new state-of-the-art results, showing a 7% absolute improvement on the channel-only accuracy and 8% absolute improvement on the full derivation tree in the most difficult condition.

5.3 Discussion

Partly these improved results are driven by better features. Adding more robust representations of the input (e.g. Brown clusters) and conditioning on prior structure of the tree leads to more consistent and coherent trees.

One key observation is that the logistic regression classifier consistently outperforms the neural network, though by a small margin. We suspect two main causes: optimization difficulties and training size. To compare the optimization algorithms, Table 1 shows the result of a neural network with no hidden layers, which is effectively identical to a logistic regression model. Stochastic gradient descent used to train the neural network did not perform as well as the LIBLINEAR (Fan et al., 2008) solver used to train the logistic regression, because the loss function was not optimized as well. Optimization problems are even more likely with hidden layers, since the objective is no longer convex.

Second, the training data is small by neural network standards. Prior attempts to use neural networks for parsing required larger amounts of training data to exceed the state-of-the-art. Non-linear models are able to capture regularities that linear models cannot, but may require more training data to do so. Table 1 shows that a network with a single hidden layer outperforms a one with two hidden layers. This additional hidden layer seems to make learning harder (even with layer-wise pretraining). We also ran an additional experiment, limiting both NN and LR to use word unigram features, and varying the vocabulary size by frequency thresholding; the results are in table 3. LR models were more effective when all features were present, likely due to their convex objective and simple regularization. NN models, on the other hand, actually outperform LR models when limited to more common vocabulary items. Given more data, NN could likely find representations that outperformed manual feature engineering.

Although we only considered feed-forward nerual networks, results on recurrent architectures Dong and Lapata (2016) are in accordance with our findings. Their LSTM-based approach does not achieve great gains on this data set because: "*user curated descriptions are often of low quality, and thus align very loosely to their corresponding ASTs*". Even though this training set is larger than other semantic parsing datasets, the vocabulary, sentence structures, and even languages here are much more diverse, which make it difficult for the NN to learn useful representations. Dong and Lapata (2016) tried to reduce the impact of this problem by evaluating only on the English sub-

	Channel	Full tree
All: 4,294 recipes		
posclass	47.4	34.5
D&L	—	—
NN	52.9	39.7
LR	53.7	41.9
Ensemble	**54.3**	**42.6**
oracleturk	48.8	37.8
Omit non-English: 3,744 recipes		
posclass	50.0	36.9
D&L	54.3	39.2
NN	55.1	41.2
LR	56.0	44.3
Ensemble	**56.8**	**44.5**
oracleturk	56.0	43.5
Omit non-English, unintelligible: 2,433 recipes		
posclass	67.2	50.4
D&L	68.8	50.5
NN	71.3	53.7
LR	71.9	56.6
Ensemble	**72.7**	**57.1**
oracleturk	86.2	59.4
≥3 agree with gold: 760 recipes		
posclass	81.4	71.0
D&L	87.8	75.2
NN	88.0	74.3
LR	88.8	**82.5**
Ensemble	**89.1**	82.2
oracleturk	100.0	100.0

Table 2: System comparisons on various subsets of the data. Following Quirk et al. (2015), we also evaluation on illustrative subsets. "posclass" represents the best system from prior work. D&L is the best-performing system from Dong and Lapata (2016). NN and LR are the single best neural network, logistic regression models, and Ensemble is the combination of all systems. "oracleturk" represents cases where at least one turker agreed with the gold standard.

set of the data. Interestingly, our carefully built feed-forward networks outperform their approach in almost every subset.

Although the neural network with one hidden layer does not outperform logistic regression in a feature rich setting, it makes substantially different predictions. An ensemble of their outputs achieves better accuracy than either system individually.

Our techniques for improving generalization do not improve individual systems. Yet when all techniques are combined in an ensemble, the resulting predictions are better. Furthermore, an ensemble without the synthetic data or without the alternate grammar has lower accuracy: each technique contributes to the final result.

System	Full tree accuracy	
	NN	LR
All words	35.79	37.03
Count ≥ 2	37.01	36.91
Count ≥ 3	37.07	36.59

Table 3: Accuracy of NN and LR limited to word unigram features, with three vocabulary sizes: all words, words occurring at least twice in the training data (13,971 words), and those occurring at least three times in the training data (8,974 words).

5.4 Comparison of logistic regression and neural network approaches

We performed a detailed exploration of the cases where either the LR model was correct and the NN model was wrong, or vice versa. Table 4 breaks these errors into a number of cases:

- **Swapped trigger and action**. Here the system misinterpreted a rule, swapping the trigger for the action. An example NN swap was "Backup Pinboard entries to diigo"; an example LR swap was "Like a photo on tumblr and upload it to your flickr photostream ."
- **Duplicated**. In this case, the system used the same channel for both trigger and action, despite clear evidence in the language. For instance, the LR model incorrectly used Facebook as both the trigger and channel in this recipe: "New photo on Facebook addec to my Pryv". The NN model correctly identified Pryv as the target channel, despite the typo in the recipe.
- **Missed word cue**. In many cases there was a clear "cue word" in the language that should have forced a correct channel, but the model picked the wrong one. For instance, in "tweet # stared youtube video", the trigger should be starred YouTube videos, but the NN model incorrectly selected feeds.
- **Missed multi-word cue**. Sometimes the cue was a multi-word phrase, such as "One Drive". The NN model tended to miss these cues.
- **Missed inference**. In certain cases the cue was more of a loose inference. Words such as "payment" and "refund" should tend to

Error type	NN errors	LR errors
Swapped trigger and action	4	4
Duplicated	3	4
Missed word cue	8	8
Missed multi-word cue	2	0
Missed inference	8	0
Related channel	5	8
Grand Total	30	24

Table 4: Count of error cases by type for NN and LR models, in their default configurations. This table only counts those instances in the most clean set (where three or more turkers agree with the gold program) where exactly one system made an error.

refer to triggers from the Square payment provider; the NN seemed to struggle on these cases.

- **Related channel**. Often the true channel is very difficult to pick: should the system use iOS location or Android location? NN models seemed to do better on these cases, perhaps picking up on some latent cues in the data that were not immediately evident to the authors.

In general, a slightly more powerful NN model with access to more relevant data might overcome some of the issues above.

We also explored correlations with errors and a number of other criteria, such as text length and frequency of the channels and functions, but found no substantial differences. In general, the remaining errors are often plausible given the noisy input.

6 Future Work

We have achieved a new state-of-the-art on this dataset, though derivation tree accuracy remains low, around 42%. While some errors are caused by training data noise and others are due to noisy test instances, there is still room for improvement.

We believe synthetic data is a promising direction. Initial attempts show small improvements; better results may be within reach given more tuning. This may enable gains with recurrent architectures (e.g., LSTMs).

The networks here rely primarily on word-based features. Character-based models have resulted in improved syntactic parsing results (Ballesteros et al., 2015). We believe that noisy data such as the If-Then corpus would benefit from character modelings, since the models could be more robust to spelling errors and variations.

Another important future work direction is to model the arguments of the If-Then statements. However, that requires segmenting the arguments into those that are general across all users, and those that are specific to the recipe's author. Likely this would require further annotation of the data.

7 Conclusion

In this paper, we address a semantic parsing task, namely translating sentences to If-Then statements. We model the task as structure prediction, and show improved models using both neural networks and logistic regression. We also discussed various ways to improve generalization and reduce overfitting, including adding synthetic training data by paraphrasing sentences, using multiple grammars, applying feature selection and ensembling multiple systems. We achieve a new state-of-the-art with 8% absolute accuracy improvement.

References

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 47–52, Sofia, Bulgaria, August. Association for Computational Linguistics.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. In *NAACL 2016*.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710, Lisbon, Portugal, September. Association for Computational Linguistics.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal, September. Association for Computational Linguistics.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-13)*.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97, Beijing, China, August. Coling 2010 Organizing Committee.

S.R.K. Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*, Singapore.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-14)*.

David L Chen. 2012. Fast online lexicon learning for grounded language acquisition. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 430–439. Association for Computational Linguistics.

Yann N Dauphin, Harm de Vries, Junyoung Chung, and Yoshua Bengio. 2015. RMSProp and equilibrated adaptive learning rates for non-convex optimization. *arXiv preprint arXiv:1502.04390*.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380, Baltimore, Maryland, June. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *arXiv:1601.01280*.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Commun. ACM*, 13(2):94–102, February.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, Atlanta, Georgia, June. Association for Computational Linguistics.

Kevin Gimpel, Dhruv Batra, Chris Dyer, and Gregory Shakhnarovich. 2013. A systematic exploration of diversity in machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1100–1111, Seattle, Washington, USA, October. Association for Computational Linguistics.

Edward Grefenstette, Phil Blunsom, Nando de Freitas, and Karl Moritz Hermann. 2014. A deep architecture for semantic parsing. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 22–27, Baltimore, MD, June. Association for Computational Linguistics.

Sumit Gulwani and Mark Marron. 2014. Nlyze: Interactive programming by natural language for spreadsheet data analysis and manipulation. In *SIGMOD*.

Leif Johnson. 2015. Theanets. `https://github.com/lmjohns3/theanets`.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June.

R. J. Kate, Y. W. Wong, and R. J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, pages 1062–1068, Pittsburgh, PA, July.

Percy Liang. 2005. *Semi-supervised learning for natural language*. Ph.D. thesis, Massachusetts Institute of Technology.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine learning*, 34(1-3):151–175.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, R. Garnett, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2755–2763. Curran Associates, Inc.

Yuk Wah Wong and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 439–446, New York City, USA, June. Association for Computational Linguistics.

William A. Woods. 1977. Lunar rocks in natural English: Explorations in natural language question answering. In Antonio Zampoli, editor, *Linguistic Structures Processing*. Elsevier North-Holland, New York.

Pengcheng Yin, Zhengdong Lu, and Ben Kao Hang Li. 2016. Neural enquirer: Learning to query tables with natural language. In *ICLR 2016*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1050–1055, Portland, OR, August.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *In Proceedings of the 21st Conference on Uncertainty in AI*, pages 658–666.