Abstract Meaning Representation Parsing using LSTM Recurrent Neural Networks

William R. Foland Jr.
Department of Computer Science
University of Colorado
Boulder, CO 80309
William.Foland@colorado.edu

James H. Martin
Department of Computer Science and
Institute of Cognitive Science
University of Colorado
Boulder, CO 80309
James.Martin@colorado.edu

Abstract

We present a system which parses sentences into Abstract Meaning Representations, improving state-of-the-art results for this task by more than 5%. AMR graphs represent semantic content using linguistic properties such as semantic roles, coreference, negation, and more. The AMR parser does not rely on a syntactic preparse, or heavily engineered features, and uses five recurrent neural networks as the key architectural components for inferring AMR graphs.

1 Introduction

Semantic analysis is the process of extracting meaning from text, revealing key ideas such as "who did what to whom, when, how, and where?", and is considered to be one of the most complex tasks in natural language processing. Historically, an important consideration has been the definition of the output of the task - how can the concepts in a sentence be captured in a general, consistent and expressive manner that facilitates downstream semantic processing? Over the years many formalisms have been proposed as suitable target representations including variants of first order logic, semantic networks, and frame-based slot-filler notations. Such representations have found a place in many semantic applications but there is no clear consensus as to the best representation. However, with the rise of supervised machine learning techniques, a new requirement has come to the fore: the ability of human annotators to quickly and reliably generate semantic representations as training data.

Abstract Meaning Representation (AMR) (Banarescu et al., 2012)[1] was developed to provide a computationally useful and expressive representation that could be reliably generated by human annotators. Sentence meanings in AMR are represented in the form of graphs consisting of concepts (nodes) connected by labeled relations (edges). AMR graphs include a number of traditional NLP representations including named entities (Nadeau and Sekine, 2007), word senses (Banerjee and Pedersen, 2002), coreference relations, and predicate-argument structures (Kingsbury and Palmer, 2002; Palmer et al., 2005). More recent innovations include wikification of named entities and normalization of temporal expressions (Verhagen et al., 2010; Strötgen and Gertz, 2010). (2016) provides an insightful discussion of the relationship between AMR and other formal representations including first order logic.

The process of creating AMR's for sentences is called AMR Parsing and was first introduced in (Flanigan et al., 2014). A key factor driving the development of AMR systems has been the increasing availability of training resources in the form of corpora where each sentence is paired with a corresponding AMR representation [2]. A consistent framework for evaluating AMR parsers was defined by the Semeval-2016 Meaning Representation Parsing Task[3]. Standard training, development and test splits for the AMR Annotation Release 1 corpus are provided, as well as an additional out-of-domain test dataset, for system comparisons. [4]

Viewed as a structured prediction task, AMR parsing poses some difficult challenges not faced by other related language processing tasks including part of speech tagging, syntactic parsing or se-

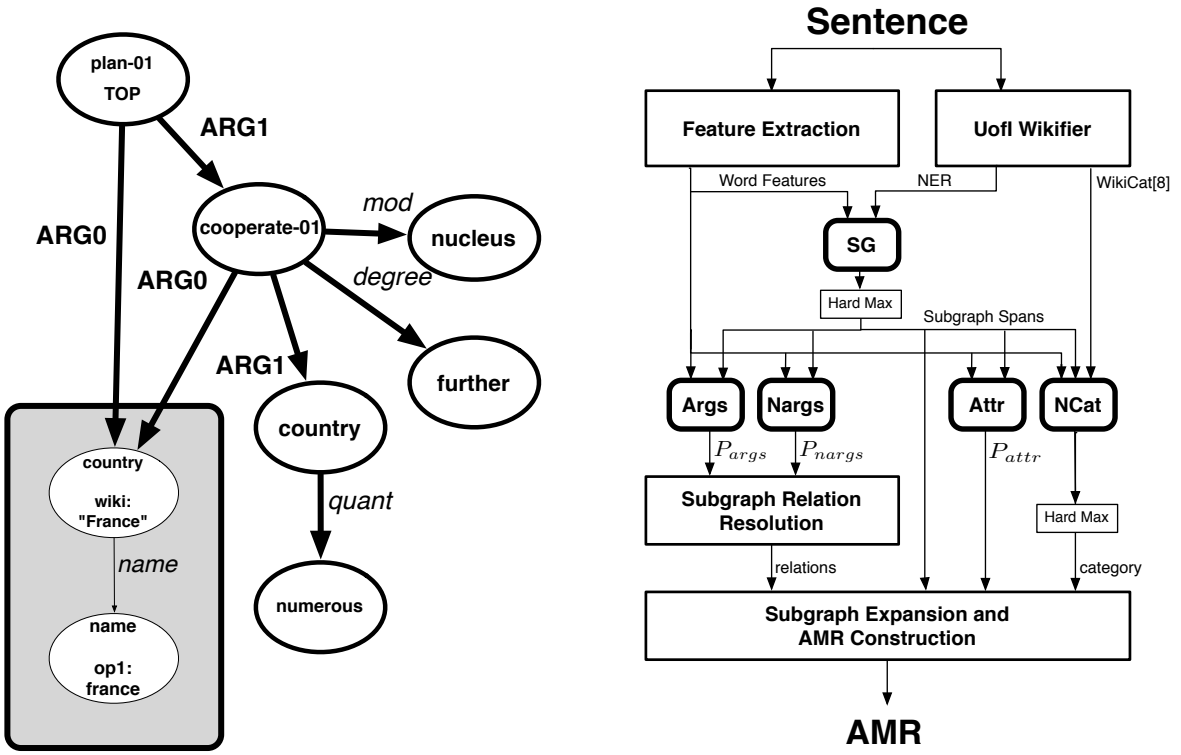
[1] http://amr.isi.edu/language.html

[2] See amr.isi.edu for information on currently available resources

[3] http://alt.qcri.org/semeval2016/task8/#

[4] Available from LDC as LDC2015E86_DEFT_Phase_2_AMR_Annotation_R1 dataset.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 463–472
Vancouver, Canada, July 30 - August 4, 2017. ©2017 Association for Computational Linguistics
https://doi.org/10.18653/v1/P17-1043



(a) An AMR graphical depiction of the meaning of the sentence ***France plans further nuclear cooperation with numerous countries .*** Concepts are represented as ovals, and relations are the directed connections between them. Predicate concepts are labelled with their PropBank sense, and semantic roles are indicated by "Arg" relations. Non-Arg relations like name or mod are called "Nargs" in this paper. Note the shaded section, which shows an example of a *subgraph*, containing related concepts and relations. In the example, the subgraph represents "France" which includes the category *country* and a shortened link to the France wiki page.

(b) General Architecture for the AMR Parser, which creates an AMR based on the words in a sentence. The 5 B-LSTM networks infer structures of the AMR. For example, the SG network infers subgraphs, which are mostly single concept, like "plan-01" or "further", but can also be like the more complex shaded "France" subgraph in the example. Other B-LSTM networks are used to infer predicate argument relations (Args), other relations (Nargs), attributes like "TOP" (Attr) and name categories like "country" for France (Ncat).

Figure 1: An example Abstract Meaning Representation and the architecture of the AMR parser, which produces an AMR from a sentence.

mantic role labeling. The prediction task in these settings can be cast as per-token labeling tasks (i.e. IOB tags) or as a sequence of discrete parser actions, as in transition-based (shift-reduce) approaches to dependency parsing.

The first challenge is that AMR representations are by design abstracted away from their associated surface forms. AMR corpora pair sentences with their corresponding representations, without providing an explicit annotation, or alignment, that links the parts of the representation to their corresponding elements of the sentence. Not surprisingly, this complicates training, decoding and evaluation.

The second challenge is the fact that, as noted earlier, the AMR parsing task is an amalgam of predicate identification and classification, entity recognition, co-reference, word sense disambiguation and semantic role labeling — each of which relies on the others for successful analysis. The architecture and system presented in the following sections is largely motivated by these two challenges.

2 Related Work

2.1 AMR Parsers

Most current AMR parsers are constructed using some form of supervised machine learning that exploits existing AMR corpora. In general, these systems make use of features derived from various forms of syntactic analysis, ranging from part-of-speech tagging to more complex dependency or phrase-structure analysis. Currently, most systems fall into two classes: (1) systems that incrementally transform a dependency parse into an AMR

graph using transition-based systems (Wang et al., 2015, 2016), and (2) graph-oriented approaches that use syntactic features to score edges between all concept pairs, and then use a maximum spanning connected subgraph (MSCG) algorithm to select edges that will constitute the graph (Flanigan et al., 2014; Werling et al., 2015).

As expected, there are exceptions to these general approaches. The largely rule-based approach of (2015) converts logical forms from an existing semantic analyzer into AMR graphs. They demonstrate the ability to use their existing system to generate AMRs in German, French, Spanish and Japanese without the need for a native AMR corpus.

(2015) proposes a synchronous hyperedge replacement grammar solution, (2015) uses syntax-based machine translation techniques to create tree structures similar to AMR, while (2015) creates logical form representations of sentences and then converts these to AMR.

An exception to the use of heavily engineered features is the deep learning approach of (2016), which, following (Collobert et al., 2011), relies on word embeddings and recurrent neural networks to generate AMR graphs.

2.2 Bidirectional LSTM Neural Networks

Unlike relatively simple sequence processing tasks like part-of-speech tagging and NER, semantic analysis requires the ability to keep track of relevant information that may be arbitrarily far away from the words currently under consideration. Recurrent neural networks (RNNs) are a class of neural architecture that use a form of short-term memory in order to solve this semantic distance problem. Basic RNN systems have been enhanced with the use of special memory cell units, referred to as Long Short-Term Memory neural networks, or LSTM's (Hochreiter and Schmidhuber, 1997). Such systems can effectively process information dispersed over hundreds of words (Schmidhuber et al., 2002; Gers et al., 2001).

Bidirectional LSTMs (B-LSTM) networks are LSTMs that are connected so that both future and past sequence context can be examined. (2015), successfully used a bidirectional LSTM network for semantic role labelling. We use the LSTM cell as described in (Graves et al., 2013), configured in a B-LSTM shown in Figure 2, as the core network architecture in the system. Five B-LSTM Neural Networks comprise the parser.

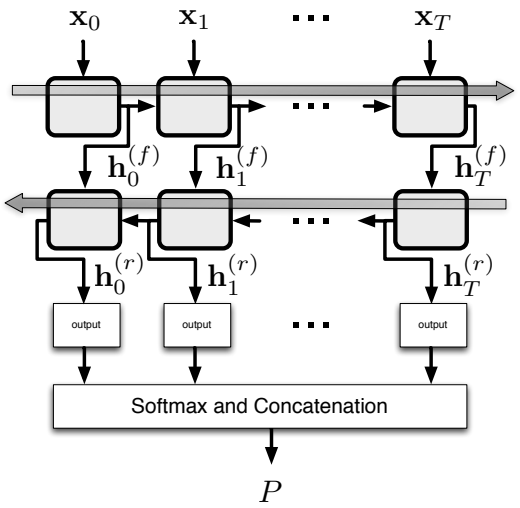


Figure 2: A general diagram of a B-LSTM network, showing the feature input vectors x_i, the forward layer (f) and the reverse layer (r). The network generates vectors of log likelihoods which are converted to probability vectors and then joined together to form an array of probabilities.

3 Parser Overview

Our parser[5] will be explained using this example sentence: ***France plans further nuclear cooperation with numerous countries .***

A graphical depiction of an AMR for this sentence is shown in Figure 1a.

Given an input sentence, the approach taken in our AMR parser is similar to (Flanigan et al., 2014) in that it consists of two subtasks: (1) discover the concepts (nodes and sub-graphs) present in the sentence, and (2) determine the relations (arcs) that connect the concepts (relations capture both traditional predicate-argument structures (ARGs), as well as additional modifier relations that capture notions including quantification, polarity, and cardinality.) Neither of these tasks is straightforward in the AMR context. Among the complications are the fact that individual words may contribute to more than one node (as in the case of France), parts of the graph may be "reentrant", participating in relations with multiple concepts, and predicate-argument and modifier relations can be introduced by arbitrary parts of the input.

At a high level, our system takes an input sentence in form of a vector of word embeddings

[5] source at https://github.com/BillFoland/daisyluAMR

and uses a series of recurrent neural networks to (1) discover the basic set of nodes and subgraphs that comprise the AMR, (2) discover the set of predicate-argument relations among those concepts, and (3) identifying any relevant modifier relations that are present.

A high level block diagram of the parser is shown in Figure 1b. The parser extracts features from the sentence which are processed by a bidirectional LSTM network (B-LSTM) to create a set of AMR subgraphs, which contain one or two concepts as well as their internal relations to each other. Features based on the sentence and these subgraphs are then processed by a pair of B-LSTM networks to compute the probabilities of relations between all subgraphs. All subgraphs are then connected using an iterative, greedy algorithm to compute a single component graph, with all subgraphs connected by relations. Separately, another two B-LSTM networks compute attribute and name categories, which are then appended to the graph. Finally, the subgraphs are expanded into the most probable AMR concept and relation primitives to create the final AMR.

4 Detailed Parser Architecture

4.1 AMR Spans, Subgraphs, and Subgraph Decoding

Mapping the words in a sentence to AMR concepts is a critical first step in the parsing process, and can influence the performance of all subsequent processing. Although the most common mapping is one word to one concept, a series of consecutive words, or **span**, can also be associated with an AMR concept. Likewise, a span of words can be mapped to a small connected **subgraph**, such as the single word span *France* which is mapped to a subgraph composed of two concepts connected by a *name* relation. (see the shaded section of Figure 1a).

Training corpora provide sentences which are annotated by humans with AMR graphs, not necessarily including a reference span to subgraph mapping. An automatic AMR **aligner** can be used to predict relationships between words and gold AMR's. We use the alignments produced by the aligner of (2014), along with the words and reference AMR graphs, to identify a **subgraph type** to associate with each span. Each word in the sentence is then associated with an IOBES subgraph type tag. We call the algorithm which defines span to subgraph mapping the **Expert Span Identifier**, and use it to train the SG Network.

A convenient development detail stems from the fact that during the AMR creation process, the identified subgraphs must be expanded into individual concepts and relations. For example, the subgraph type "Named", along with the span *France*, must be expanded to create the concepts, relations, and attributes shown in Figure 1a. A **Subgraph Expander** algorithm implements this task, which is essentially the inverse of the Expert Span Identifier. The Expert Span Identifier and Subgraph Expander were developed by cascading the two in a test configuration as shown in Figure 3a.

4.2 Features

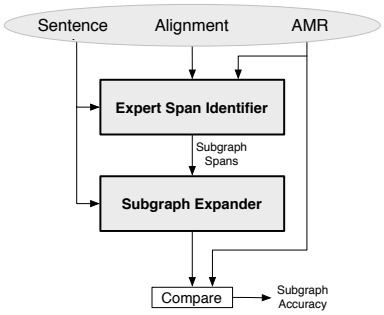
All input features for the five networks correspond to the sequence of words in the input sentence, and are presented to the networks as indices into lookup tables. With the exception of pre-trained word embeddings, these lookup tables are randomly initialized prior to training and representations are created during the training process.

4.2.1 Word Embeddings

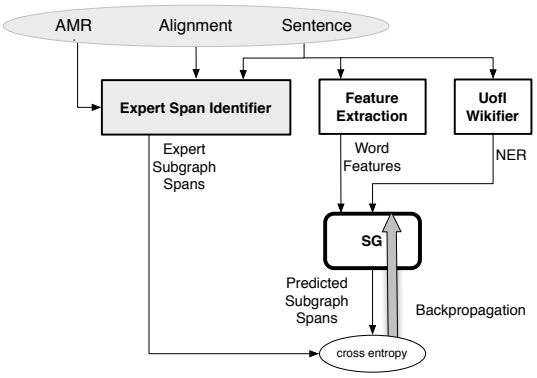
The use of distributed word representations generated from large text corpora is pervasive in modern NLP. We start with 300 dimension GloVe representations (Pennington et al., 2014) trained on the 840 billion word common crawl (Smith et al., 2013). We added two binary dimensions: one for out of vocabulary words, and one for padding, resulting in vectors with a width of 302. These embeddings are mapped from the words in the sentence, and are then trained using back propagation just like other parameters in the network.

4.2.2 Wikifier

The AMR standard was expanded to include the annotation of named entities with a canonical form, using Wikipedia as the standard (see *France* in Figure 1a). The wiki link associated with this "wikification" is expressed using the :wiki attribute, which requires some kind of global external knowledge of the Wikipedia ontology. We use the University of Illinois Wikifier (Ratinov et al., 2011; Cheng and Roth, 2013) to identify the :link directly, and use the possible categories output from the wikifier as feature inputs to the NCat Network.



(a) Expert System and Subgraph Expander Development. The alignment between the words in the sentence and elements of the AMR is provided by an automatic aligner. The expert system uses the sentence, reference AMR, and alignment to identify spans of words which are related to concepts within the AMR. These spans are also labelled with a subgraph type. A "subgraph expander" uses the words and subgraph type to expand into AMR subgraphs.



(b) SG Network Training. The SG Network uses just the words in the sentence as input, and is trained to imitate the output of the Expert System. This output defines spans of words and their subgraph types, which are the nodes of the AMR graph. Later stages of the system use this information to infer other aspects of the AMR, like relations (edges).

Figure 3: SG Model Development Details.

Named Entity Recognition can be valuable input to a parser, and state-of-the-art NER systems can be created using convolutional neural networks (Collobert et al., 2011) or LSTM (Chiu and Nichols, 2015) aided by information from gazetteers. These gazetteers are large dictionaries containing well known named entities (e.g., (Florian et al., 2003)).

Rather than add gazetteer features to our system, we make use of the NER information already calculated and provided by the Univ. of Illinois Wikifier. We then encode the classified named entities output from the wikifier as feature embeddings, which are used by the SG Network.

4.2.3 AMR Subgraph (SG) Network

The features used as input to the SG network are:

- word: 45Kx302, the word embeddings
- suffix: 430x5, embeddings based on the final two letters of each word.
- caps: 5x5, embeddings based on the capitalization pattern of the word.
- NER: 5x5, embeddings indexed by NER from the Wikifier, 'O', 'LOC', 'ORG', 'PER' or 'MISC'.

The SG Network produces probabilities for 46 BIOES tagged subgraph types, and the highest probability tag is chosen for each word, as shown for the example sentence in Table 1.

4.2.4 Predicate Argument Relations (Args) Network

The AMR concepts (nodes) are connected by relations (arcs). We found it convenient to distinguish predicate argument relations, or "Args" from other relations, which we call "Nargs". For example, see ARG0 and ARG1 relations in Figure 1a are "Args", compared with the name, degree, mod, or quant relations which are "Nargs".

The Args Network is run once for each predicate subgraph, and produces a matrix P_{args} which defines the probability (prior to the identification of any relations[6]) of a type of predicate argument relation from a predicate subgraph to any other SG identified subgraph. (For example, see ARG0 and ARG1 relations in Figure 1a.) The matrix has dimensions 5 by s, where 5 is the number of predicate arg relations identified by the network, and s is the total number of subgraphs identified by the SG Network for the sentence.

The Args features, calculated for each source predicate subgraph, are:

- Word, Suffix and Caps as in the SG network.
- SG: 46x5, indexed by the SG network identified subgraph.
- PredWords[5], 45Kx302: The word embeddings of the word and surrounding 2 words associated with the source predicate subgraph.

[6]relation probabilities change as hard decisions are made, see section 4.3

words	BIOES	Prob	kind
France	S_Named	0.995	Named subgraph
plans	S_Pred-01	0.997	plan-01
further	S_NonPred	0.931	further
nuclear	S_NonPred	0.990	nucleus
cooperation	S_Pred-01	0.986	cooperate-01
with	O	1.000	O
numerous	S_NonPred	0.982	numerous
countries	S_NonPred	0.860	country
.	O	0.999	O

Table 1: SG Network Example Output

feature	width
Word[france]	302
Suffix[ce]	5
Caps[firstUp]	5
SG[S_Named]	10
Word[further]	302
Word[nuclear]	302
Word[cooperation]	302
Word[with]	302
Word[numerous]	302
SG[S_NonPred]	10
SG[S_NonPred]	10
SG[S_Pred-01]	10
SG[O]	10
SG[S_NonPred]	10
Distance[4]	5

Table 2: Args Network Features for the word *France* while evaluating outgoing args for the word *cooperation*, associated with predicate **cooperate-01**

- PredSG[5], 46x10: The SG embedding of the word and surrounding 2 words associated with the source predicate subgraph.
- regionMark: 21x5, indexed by the distance in words between the word and the word associated with the source predicate subgraph.

Table 2 shows an example feature set for one subgraph while evaluating a predicate subgraph.

4.2.5 Non-Predicate Relations (Nargs) Network

The Nargs Network uses features similar to the Args network. It is run once for each subgraph, and produces a matrix P_{nargs} which defines the probability of a type of relation from a subgraph to any other subgraph, prior to the identification of any relations.[7] The matrix has dimensions 43 by s, where 43 is the number of non-arg relations identified by the network, and s is the total number of subgraphs identified by the SG Network for the sentence.

4.2.6 Attributes (Attr) Network

The Attr Network determines a primary attribute for each subgraph, if any.[8] This network is simplified to detect only one attribute (there could be many) per subgraph, and only computes probabilities for the two most common attributes: TOP and polarity. Note that subgraph expansion also identifies many attributes, for example the words associated with named entities, or the normalized quantity and date representations. A known shortcoming of this network is that the TOP and polarity attributes are not mutually exclusive, but noting that the cooccurrence of the two does not occur in the training data, we chose to avoid adding a separate network to allow the prediction of both attributes for a single subgraph.

4.2.7 Named Category (NCat) Network

The NCat Network uses features similar to the SG Network, along with the suggested categories (up to eight) from the Wikifier, and produces probabilities for each of 68 :instance roles, or categories, for named entities identified in the training set AMR's.

- Word, Suffix and Caps as in the SG network.
- WikiCat[8]: 108 x 5, indexed by suggested categories from the Wikifier.

4.3 Relation Resolution

The generated P_{args} and P_{nargs} for each SG identified subgraph are processed to determine the most likely relation connections, using the constraints:

[7] Degree, mod, or quant are examples of Narg relations in Figure 1a.

[8] (TOP: plan-01) and (op1: france) are attribute examples shown in Figure 1a.

1. AMR's are single component graphs without cycles.
2. AMR's are simple directed graphs, a max of one relation between any two subgraphs is allowed.
3. Outgoing predicate relations are limited to one of each kind (i.e. can't have two ARG0's)

We initialize a graph description with all the subgraphs identified by the SG network. Probabilities for all possible edges are represented in the P_{args} and P_{nargs} matrices. The Subgraphs are connected to one another by applying a greedy algorithm, which repeatedly selects the most probable edge from the P_{args} and P_{nargs} matrices and adds the edge to the graph description. After an edge is selected to be added to the graph, we adjust P_{args} and P_{nargs} based on the constraints (hard decisions change the probabilities), and repeat adding edges until all remaining edge probabilities are below a threshold. (The optimum value of this threshold, 0.55, was found by experimenting with the development data set). From then on, only the most probable edges which span graph components are chosen, until the graph contains a single component.

Expressed as a step by step procedure, we first define $p_{connect}$ as the probability threshold at which to require graph component spanning, and we repeat the following, until any two subgraphs in the graph are connected by at least one path.

1. Select the most probable outgoing relation from any of the identified subgraph probability matrices. Denote this probability as p_r.
2. If $p_r < p_{connect}$, keep selecting most probable relations until a component spanning connection is found.
3. Add the selected relation to the graph. If a cycle is created, reverse the relation direction and label.
4. Eliminate impossible relations based on the constraints and re-normalize the affected P_{args} and P_{nargs} matrices.

4.4 AMR Construction

AMR Construction converts the connected subgraph AMR into the final AMR graph form, with proper concepts, relations, and root, as follows:

1. The TOP attribute occurs exactly once in each AMR, so the subgraph with highest TOP probability produced by the Attr network is identified. The AMR graph is adjusted so that it is rooted with the most probable TOP subgraph. After graph adjustment, new cycles are sometimes created, which are removed by using *-of* relation reversal.
2. The subgraphs identified by the SG network, which were considered to be single nodes during relation resolution, are expanded to basic AMR concepts and relations to form a concept/relation AMR graph representation, using the Subgraph Expander component developed as shown in Figure 3b. When a subgraph contains two concepts, the choice of connecting to parent or child within the subgraph is made based on training data statistics of each relation type (Arg or Narg) for each subgraph type.
3. Nationalities are normalized (e.g. French to France).
4. A very basic coreference resolution is performed by merging all concepts representing "I" into a single concept. Coreference resolution was otherwise ignored due to development time constraints.

5 Experimental Setup

Semantic graph comparison can be tricky because direct graph alignment fails in the presence of just a few miscompares. A practical graph comparison program called Smatch (Cai and Knight, 2013) is used to consistently evaluate AMR parsers. The smatch python script provides an F1 evaluation metric for whole-sentence semantic graph analysis by comparing sets of triples which describe portions of the graphs, and uses a hill climbing algorithm for efficiency.

All networks, including SG, were trained using stochastic gradient descent (SGD) with a fixed learning rate. We tried sentence level log-likelihood, which trains a viterbi decoder, as a training objective, but found no improvement over word-level likelihood (cross entropy). After all LSTM and linear layers, we added dropout to minimize overfitting (Hinton et al., 2012) and batch normalization to reduce sensitivity to learning rates and initialization (Ioffe and Szegedy, 2015).

For each of the five networks, we used the LDC2015E86 training split to train parameters, and periodically interrupted training to run the dev split (forward) in order to monitor performance.

The model parameters which resulted in best dev performance were saved as the final model. The test split was used as the "in domain" data set to assess the fully assembled parser. The inferred AMR's were then evaluated using the smatch program to produce an F1 score.

An evaluation dataset was provided for Semeval 2016 task 8, which is significantly different from the LDC2015E86 split dataset. ((2016) describes the eval dataset as "quite difficult to parse, particularly due to creative approaches to word representation in the web forum portion").

6 Results

We report the statistics for smatch results of the "test" and "eval" datasets for 12 trained systems in Table 3. The top five scores for Semeval 2016 task 8, representing the previous state-of-the-art, are shown for context. With a smatch score of between 0.651 and 0.654, and a mean of 0.652, our system improves the state-of-the-art AMR parser performance by between 5.07% and 5.55%, and by a mean of 5.22%. The best performing systems for in-domain (dev and test) data correlated well with the best ones for the out-of-domain (eval) data, although the scores for the eval dataset were lower overall.

6.1 Individual Network Results

The word spans tagged by the SG network are used to determine the features for the other networks. In particular, every span identified as a predicate will trigger the system to evaluate the Args network in order to determine the probabilities of outgoing predicate ARG relations. Likewise, all spans identified as subgraphs (other than named subgraphs) will lead to a Nargs network evaluation to determine outgoing non-Arg relations. The SG network identifies predicates with 0.93 F1, named subgraphs with 0.91 F1, and all other subgraphs with 0.94 F1.

The Args network identifies ARG0 and ARG1 relations with 0.73 F1, but identification of ARG2, ARG3, and ARG4 drops down to (0.53, 0.20, and 0.43). It is difficult for the system to generalize among these relation tags because they differ significantly between predicates.

7 Conclusion and Future Work

We have shown that B-LSTM neural networks can be used as the basis for a graph based semantic parser. Our AMR parser effectively exploits the ability of B-LSTM networks to learn to selectively extract information from words separated by long distances in a sentence, and to build up higher level representations by rejecting or remembering important information during sequence processing. There are changes which could be made to eliminate all pre-processing and to further improve parser performance.

Eliminating the need for syntactic pre-parsing is valuable since a syntactic parser takes up significant time and computational resources, and errors in the generated syntax will propagate into an AMR parser. Our approach avoids both of these problems, while generating high quality results.

Wikification tasks are generally independent from parsing, but wiki links are a requirement for the latest AMR specification. Since our preferred wikifier application generates NER information, we used the generated NER tags as input to the SG network. But it would also be fairly easy to add gazetteer information to the network features in order to remove the need for NER preprocessing. Therefore, the wikification subtask is the only portion of the parser which requires any pre-processing at all. Incorporating wikification gazetteers as B-LSTM features might allow a performant, fully self contained parser to be created.

Sense disambiguation is not a very generalizable task, senses other than 01 and 02 for different predicates may differ from each other in ways which are very difficult to discern. A better approach to disambiguation is to consider predicates separately, solving for a set of coefficients for each verb found in the training set. A general set of model parameters could then be used to handle unseen examples. Likewise, high level ARGs like ARG2 and ARG3 don't generalize very well among different predicates, and ARG inference accuracy could be improved with predicate-specific network parameters for the most common cases.

The alignment between concepts and words is not a reliable, direct mapping: some concepts cannot be grounded to words, some are ambiguous, and automatic aligners tend to have high error rates relative to human aligning judgements. Improvements in the quality of the alignment in training data would improve parsing results.

System	Description	Test F1	Eval (OOD) F1
Our Parser (summary of 12 trained systems)	**mean**	**0.707**	**0.652**
	min	**0.706**	**0.651**
	max	**0.709**	**0.654**
RIGA (Barzdins and Gosko, 2016)		0.6720	0.6196
Brandeis/cemantix.org/RPI (Wang et al., 2016)		0.6670	0.6195
CU-NLP (Foland Jr and Martin, 2016)		0.6610	0.6060
ICL-HD (Brandt et al., 2016)		0.6200	0.6005
UCL+Sheffield (Goodman et al., 2016)		0.6370	0.5983

Table 3: Smatch F1 results for our parser and top 5 parsers from semeval 2016 task 8.

References

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. Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štepánek, Pavel Stranák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2012. Abstract meaning representation (amr) 1.0 specification. In *Parsing on Freebase from Question-Answer Pairs. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Seattle: ACL*. pages 1533–1544.

Satanjeev Banerjee and Ted Pedersen. 2002. An adapted lesk algorithm for word sense disambiguation using wordnet. In *Computational linguistics and intelligent text processing*, Springer, pages 136–145.

Guntis Barzdins and Didzis Gosko. 2016. Riga at semeval-2016 task 8: Impact of smatch extensions and character-level neural translation on amr parsing accuracy. *arXiv preprint arXiv:1604.01278* .

Johan Bos. 2016. Expressive power of abstract meaning representations. *Computational Linguistics* 42(3):527–535.

Lauritz Brandt, David Grimm, Mengfei Zhou, and Yannick Versley. 2016. Icl-hd at semeval-2016 task 8: Meaning representation parsing-augmenting amr parsing with a preposition semantic role labeling neural network. *Proceedings of SemEval* pages 1160–1166.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *ACL (2)*. pages 748–752.

X. Cheng and D. Roth. 2013. Relational inference for wikification. In *EMNLP*. http://cogcomp.cs.illinois.edu/papers/ChengRo13.pdf.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308* .

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research* 12:2493–2537.

Jeffrey Flanigan, Sam Thomson, Jaime G Carbonell, Chris Dyer, and Noah A Smith. 2014. A discriminative graph-based parser for the abstract meaning representation .

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. Association for Computational Linguistics, Stroudsburg, PA, USA, CONLL '03, pages 168–171. https://doi.org/10.3115/1119176.1119201.

William R Foland Jr and James H Martin. 2016. Cu-nlp at semeval-2016 task 8: Amr parsing using lstm-based recurrent neural networks. *Proceedings of SemEval* pages 1197–1201.

Felix A Gers, Douglas Eck, and Jürgen Schmidhuber. 2001. Applying lstm to time series predictable through time-window approaches. In *Artificial Neural NetworksICANN 2001*, Springer, pages 669–676.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Ucl+ sheffield at semeval-2016 task 8: Imitation learning for amr parsing with an α-bound. *Proceedings of SemEval* pages 1167–1172.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. 2013. Speech recognition with deep recurrent neural networks. *CoRR* abs/1303.5778. http://arxiv.org/abs/1303.5778.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* abs/1207.0580. http://arxiv.org/abs/1207.0580.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR* abs/1502.03167. http://arxiv.org/abs/1502.03167.

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *LREC*. Citeseer.

Jonathan May. 2016. Semeval-2016 task 8: Meaning representation parsing. *Proceedings of SemEval* pages 1063–1073.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes* 30(1):3–26.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics* 31(1):71–106.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for amr parsing. *CoNLL 2015* page 32.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)* 12.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning english strings with abstract meaning representation graphs. In *EMNLP*. pages 425–429.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing english into abstract meaning representation using syntax-based machine translation. *Training* 10:218–021.

L. Ratinov, D. Roth, D. Downey, and M. Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *ACL*. http://cogcomp.cs.illinois.edu/papers/RRDA11.pdf.

Jürgen Schmidhuber, F Gers, and Douglas Eck. 2002. Learning nonregular languages: A comparison of simple recurrent networks and lstm. *Neural Computation* 14(9):2039–2041.

Jason R Smith, Herve Saint-Amand, Magdalena Plamada, Philipp Koehn, Chris Callison-Burch, and Adam Lopez. 2013. Dirt cheap web-scale parallel text from the common crawl. In *ACL (1)*. pages 1374–1383.

Jannik Strötgen and Michael Gertz. 2010. Heideltime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pages 321–324.

Lucy Vanderwende, Arul Menezes, and Chris Quirk. 2015. An amr parser for english, french, german, spanish and japanese and a new amr-annotated corpus. In *Proceedings of NAACL-HLT*. pages 26–30.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. Semeval-2010 task 13: Tempeval-2. In *Proceedings of the 5th international workshop on semantic evaluation*. Association for Computational Linguistics, pages 57–62.

Chuan Wang, Sameer Pradhan, Nianwen Xue, Xiaoman Pan, and Heng Ji. 2016. Camr at semeval-2016 task 8: An extended transition-based amr parser. *Proceedings of SemEval* pages 1173–1178.

Chuan Wang, Nianwen Xue, Sameer Pradhan, and Sameer Pradhan. 2015. A transition-based algorithm for amr parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 366–375.

Keenon Werling, Gabor Angeli, and Christopher Manning. 2015. Robust subgraph generation improves abstract meaning representation parsing. *arXiv preprint arXiv:1506.03139* .

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.