Natural Language Understanding:
Deep Learning for Abstract Meaning Representation

by

William R. Foland, Jr.

B.S.E.E., University of Colorado, Boulder, 1980

M.S.C.S., University of Colorado, Boulder, 2014

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

2017

This thesis entitled:
Natural Language Understanding:
Deep Learning for Abstract Meaning Representation
written by William R. Foland, Jr.
has been approved for the Department of Computer Science

_____
Prof. James H. Martin

_____
Prof. Martha Palmer

_____
Prof. Wayne H. Ward

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Foland, Jr., William R. (Ph.D., Computer Science)

Natural Language Understanding:

Deep Learning for Abstract Meaning Representation

Thesis directed by Prof. James H. Martin

In the last few years there have been major improvements in the performance of hard natural language processing tasks due to the application of artificial neural network models. These models replace complex hand-engineered systems for extracting and representing the meaning of human language with systems which learn features based on processing examples of language. In this dissertation, I present deep neural networks for semantic role labeling, and then for Abstract Meaning Representation parsing, and a novel Distributed Abstract Meaning Representation, or DAMR. I then describe a model used to create fixed vector representations of sentence meaning from DAMR. Finally, I use natural language inference to test the quality of the meaning content of these fixed vectors.

Dedication

In loving memory of my mom and dad, and to my wonderful wife, Shelly, who encourages me in everything I try, no matter how crazy it seems at the time.

Acknowledgements

It was a great privilege to work closely with my advisor Jim Martin, I was happy to be guided by someone with such experience and knowledge in NLP. Jim met with me periodically to brainstorm and guide me along, and encouraged me to pursue my degree in the field Ive been interested in for so long.

Special thanks to Wayne Ward for sharing his excellent experience, it has been a pleasure to work with him and to discuss ideas for new algorithms for speech recognition.

I'm grateful to Martha Palmer for organizing the CU Computational Semantics Research group, and to all of my colleagues in that group. It continues to be a lively and collaborative environment in which to exchange ideas and listen to great NLP experts from around the world.

I've been lucky to have had the opportunity to interact with such a talented group of professors at CU Boulder.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

Natural Language Understanding is the subset of Natural Language Processing that deals with computer comprehension of human language. Computers can be programmed to understand human language by mapping it to structures, and then performing inference on those structures. This dissertation describes novel research for mapping natural language to symbolic and vector representation structures using neural networks, and presents results obtained using this approach.

1.1 Natural Language Understanding

The models described here produce computer-friendly meaning representations, which are primarily meant to be used as the input for further processing in order to accomplish some higher-order meaning associated task. This process is generally known as Natural Language Understanding (NLU). NLU is a critical part of many of the most actively researched Natural Language Processing (NLP) problems today, including question answering, translation, and dialog analysis.

NLU can be viewed as having various levels of intensity, ranging from shallow to deep. (note that this has nothing to do with the ”Deep” in ”Deep Learning”). Shallow NLU extracts specialized, simple meaning, such as determining the departure and destination cities in an airline reservation application. In that case, specific ”slots” are filled based on the specific domain of interest. Deep NLU attempts to understand language in a more general fashion, with less assumption about semantic context. Deep NLU might be used to understand detailed technical content in a professional journal, or to analyze the plot of a novel. In this dissertation we focus on representations

and methods which can be used for Deep NLU.

For any task that requires NLU, when it is possible to consider the semantic content of text to be limited to a narrow field, the accuracy of the NLU algorithm can be much higher than if a more general context must be considered, so its important to match the algorithms to desired outcome.

Current methods for Deep Natural Language Understanding attempt to first model natural language in a general way, capturing ideas that are expressed in text and mapping them into a representation, which represents some state of affairs of the world. The representations are then related to the ”world” we are interested in modeling. It is important to limit the ”world” we will relate to if it can be done, because we will then increase the accuracy of our analysis. Examples of the world we model for Deep NLU might be drug reactions for a medical text analytics application, or useful information specific to a human user in the case of a personal digital assistant. The data associated with the world is often stored in some sort of database, called a knowledge base.

The general system ability to draw valid conclusions from meaning representations and a knowledge base is called inference. A practical system should be able to make an assessment about the truth of propositions which are not explicitly expressed, but which can be logically derived from the ideas expressed in a meaning representation. The two most commonly used forms of meaning representations for Deep NLU are called Symbolic and Distributed. We now describe each of these, along with the general methods which can be used to relate representations of one of the two forms to each other.

1.2 Symbolic Meaning Representations

The symbols which are used in Symbolic Representations are used to represent objects, relations between objects, and attributes of objects.

Groups of words in a sentence, like *Rocky Mountain National Park*, can be mapped to a single object, in this case, a noun phrase, which can be identified by a subtask called shallow parsing, or chunking. Another example of a symbolic subtask, called semantic role labelling, identifies the roles of words in a sentence as related to a particular verb, or predicate.

1.2.1 Abstract Meaning Representation

A recent innovation in symbolic representations is the Abstract Meaning Representation (AMR) which is designed to incorporate these individual subtask productions into a single coherent meaning representation. AMR is meant to be a more general purpose, broad-coverage meaning representation, as opposed to a task-specific representation. While almost any isolated sentence can be interpreted in different ways based on other evidence, such as surrounding context, there is a notion of most probable meaning for a sentence, and a sentence can be disambiguated based on that notion. The AMR specification comes close to being able to represent any disambiguated english language sentence and development is continuing to improve its expressiveness.

Figure 1.1 shows an example of an AMR for the simple sentence *It definitely sounds interesting.*

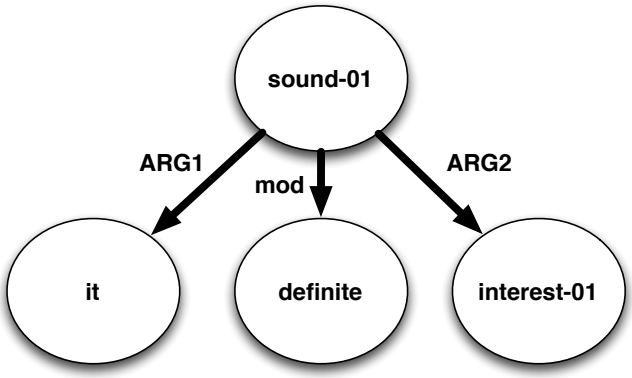


Figure 1.1: AMR Graph for the sentence: *It definitely sounds interesting.*

1.2.2 Truth-Conditional Reasoning

Truth-Conditional Reasoning requires some definition of truth, and a means for testing the truthfulness of a meaning representation. This in turn enables verifiability, which allows us to compare the state of affairs expressed in the language model to the state of affairs of our world. To this end, automated deduction methods can be used in order to relate meaning representations to a knowledge base [Montague, 1973, Blackburn and Bos, 2005].

One commonly used logical framework is called first order logic, which is a flexible, well-understood, and computationally tractable means of providing a truth-conditional mapping from symbolic meaning representation expressions to the world being modeled. A typical reasoning flow using symbolic meaning representations is shown in Figure 1.2. [1]

1.3 Distributed Meaning Representations

Vector representations for symbolic representations consist of one-hot, or integer valued vectors which are the length of the vocabulary, and they are therefore considered high dimensional and sparse. Distributed Meaning Representation vectors express meaning using a relatively small number (50 to 1000) of collections of real-valued parameters, and are described as low dimensional and dense. The advantage of representing meaning this way is that comparisons between representations become meaningful - cat vs. dog is more similar than cat vs. plant, for example.

1.3.1 Word Vectors

The use of distributed word representations generated from large text corpora is pervasive in modern NLP. When word meanings are represented as vectors, we can consider this as defining a semantic hyperspace, where each word is defined by its position. The idea of representing meaning in a geometrical space is not new, it dates back to [Osgood and Suci, 1957], who asked subjects to rate word meanings on a series of scales whose extremes were polar opposites, like happy - sad.

Vectors are the natural feature representation for neural networks, and much of todays NLP research starts with word representations as input to neural networks, as does the work described here.

Word Vectors can be trained ”from scratch” during task specific, supervised training. However, over the past decade or so, automated techniques which use the context of words in large

[1] A more restricted logic framework, called Simple Discrete Event Calculus (SDEC), was used by [Mitra and Baral, 2015] to accurately solve a set of text understanding and reasoning tasks posed by [Weston et al., 2015]. AMR representations were obtained using an early AMR parser [Flanigan et al., 2014]. They were then translated to SDEC, and processed by a reasoning engine, to answer basic questions. More details will be described in Chapter 2.

quantities of text to create vectors has revolutionized Natural Language Processing in many areas. The pre-trained word representations generated in this way group similar words together, and contain a remarkable amount of meaning structure. Furthermore, the process has become efficient and widely available. Pre-computed vectors of various sizes have been made available on the internet for research.

One of the earliest use of automatically generated word representations dates back to 1986 due to [Rumelhart et al., 1988]. The follow up work includes applications to automatic speech recognition and machine translation [Schwenk, 2007, Mikolov, 2012], and a wide range of NLP tasks part-of-speech tags, chunks, named entity tags, semantic roles, semantically similar words and the likelihood that the sentence makes sense: [Collobert and Weston, 2008] sentiment classification: [Glorot et al., 2011] parsing natural scenes: [Socher et al., 2011b] analogy and paraphrase: [Turney, 2013] image annotation: [Weston et al., 2011]

The word representations computed using neural networks are very interesting because these embeddings explicitly encode many linguistic regularities and patterns. Somewhat surprisingly, many of these patterns can be represented as linear translations. For example, the result of a vector calculation vec(Madrid) - vec(Spain) + vec(France) is closer to vec(Paris) than to any other word vector [Mikolov et al., 2013b, Mikolov et al., 2013a].

1.3.2 Sentence Vectors

The principle of compositionality describes how the semantic content of words and syntax compose the larger meaning expressed by sentences. A distributed representation for sentence level meaning is described in Chapter 5, and a means of converting this representation to a fixed sized vector is described in Chapter 6. Fixed sized sentence vectors are interesting because they can be used as the input for many different semantic processing tasks and applications.

One method of composing sentence meaning vectors without syntax is surprisingly effective: simply add the word representations together, illustrated by the composition example Vietnam + capitol = Hanoi in [Mikolov et al., 2013b]. We will explore more effective compositional meth-

ods such as convolutional and recurrent neural networks, which learn to model complex semantic phenomena that depend on word order, for example.

1.3.3 Relational Reasoning

Entailment and contradiction are considered to be key concepts in the linguistic field of semantics, necessary (but not sufficient) for proper understanding of language. Natural Language Inference (NLI) is the process of identifying and using these relations in computational systems [Fyodorov et al., 2000, Condoravdi et al., 2003, Bos and Markert, 2005, Dagan et al., 2006, MacCartney and Manning, 2009]

An NLI model considers two sentences, a premise and a hypothesis, and attempts to determine their relationship to each other. A common set of labels used to define this relationship is:

- entailment: the meaning of the hypothesis can be inferred (entailed) from the premise
- contradiction: the meaning of the hypothesis contradicts the premise
- neutrality: the two sentences are semantically independent.

For example, given the premise *Small boy wearing blue shorts sitting on bed*, the hypothesis *the boy's shorts are red* is a contradiction. Likewise, *The boy is 6 years old* is neutral, and *A boy is wearing shorts* is an entailment.

An inference-based reasoning flow using distributed meaning representations is shown in Figure 1.3. The task is simple to define, but demands handling complex phenomena such as lexical and syntactic ambiguity, lexical entailment, quantification, coreference, tense, belief, and modality. As such, NLI is considered to be a very good benchmark task for assessing reasoning capability.

1.4 Artificial Neural Networks

Artificial Neural networks are algorithms which can be run on a computer, and which loosely mimic the way that some believe that biological brains, including human brains, work. The theory

is over fifty years old [Rosenblatt, 1962], and enthusiasm for it ebbs and flows over the decades. Recently, new applications exhibiting state of the art performance, using advanced neural network architectures and much cheaper and more powerful computers, are making them very interesting once again.

In a biological neural network, like that shown in Figure 1.4[1] , axon terminals connect via synapses to dendrites on other neurons. The electrical signals from one neuron to another can have different synaptic strengths. If the sum of the input signals into one neuron surpasses a certain threshold, the neuron sends an action potential (AP) at the axon hillock and transmits this electrical signal along the axon to the next group of neurons.

In an artificial neural network composed of artificial neurons like (figure 1.5), signals from a simulated neuron are multiplied by a **weight** before being transmitted to other neurons. This weight is similar to synaptic strength in a biological network.

Biological axon firing based on a threshold is a form of **non-linearity**. The artificial equivalent is a quick but smooth mathematical transition, such as a hyperbolic tangent, a sigmoid function, or the currently popular rectified linear unit (RELU).

Like many natural phenomena, rather than the incredibly complex structure of trillions of independent connections, the brain is believed to be organized in a more organized, fundamental way which can be thought of as successive application of a common pattern of neurons, which some, such as [Kurzweil, 2012], call a pattern recognizer. The brain is remarkably adaptable. For example, it is known that when a specialized part of the brain is damaged, other parts can learn to compensate, suggesting that seemingly unlike functions such as sight or speech are composed of similar structures.

An artificial neural network structure is created with default connections and connection strengths, and an algorithm is applied to ”learn” the connections and weights of this general structure. The algorithm is called ”back-propagation” because it runs in the opposite direction from

[1] Source: ”Blausen 0657 MultipolarNeuron” by BruceBlaus - Own work. Licensed under Creative Commons Attribution 3.0 via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:Blausen_0657_MultipolarNeuron.png#mediaviewer/File:Blausen_0657_MultipolarNeuron.png.

which the network normally runs, as will be explained. The process of running this learning algorithm is called ”training”.

When the parameters of a network have been trained, the network is ready to perform its task, which is referred to as running the network, or model, forward.

1.5 This Dissertation

Experiments with deep learning for generating symbolic and distributed meaning representations constitute the primary mode of research in this dissertation. My goal is to understand and improve neural network based semantic extraction models that have been empirically shown to help solve hard natural language processing problems. After Chapter 2 introduces the key technical concepts which will be used, Chapter 3 presents a convolutional neural network based semantic role labelling system, which achieves state of the art performance for a dependency parse sourced system. The key finding was that separate role and sense inference models yielded the best results, and an ablation study shows the influence of various input features. Chapter 4 uses Long short-term memory networks to create an Abstract Meaning Representation parser, this time without using explicit syntax from a separate parsing task. The system learns to identify syntactic clues embedded within the sentence implicitly, and obtains state of the art results. In chapter 5, the AMR parser is modified and improved to create an intermediate representation, a Distributed Abstract Meaning Representation. The parser results are further improved with the new architecture, which confirms that the meaning of a sentence is adequately represented in the intermediate representation. In chapter 6, this intermediate representation is used as a feature source to a bridging network, which produces a fixed-sized vector representing sentence meaning. The sentence vectors produced are then tested using two natural language inference corpora, SNLI and MultiNLI. The results are promising, yielding competitive performance compared with other published sentence vector techniques. Finally, Chapter 7 concludes and lays out promising directions for future work.

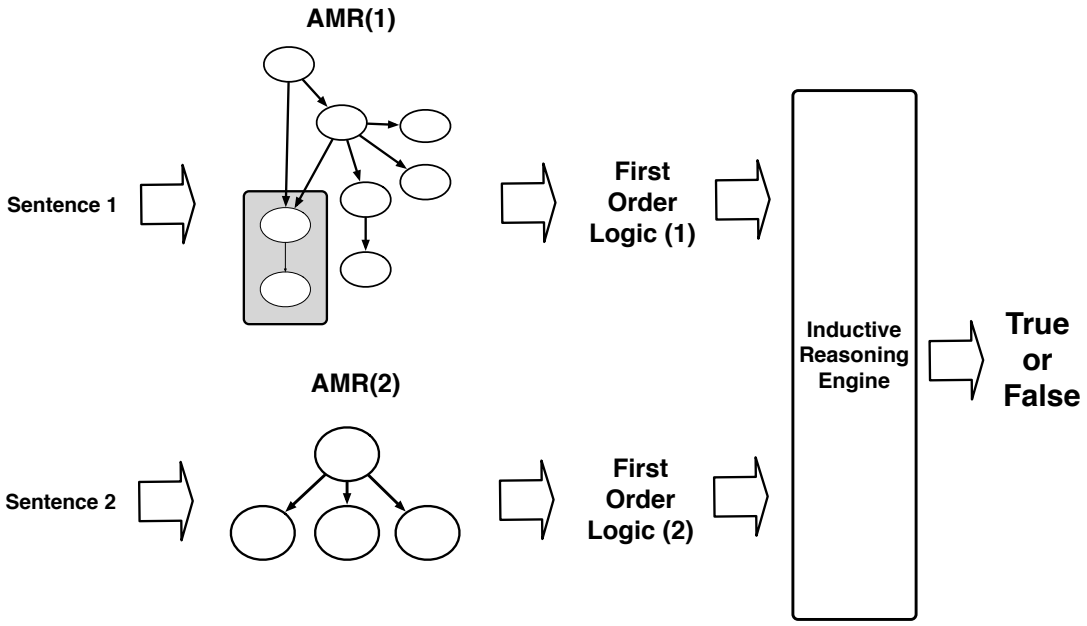


Figure 1.2: Symbolic Representations used in a Truth-Conditional Reasoning Flow

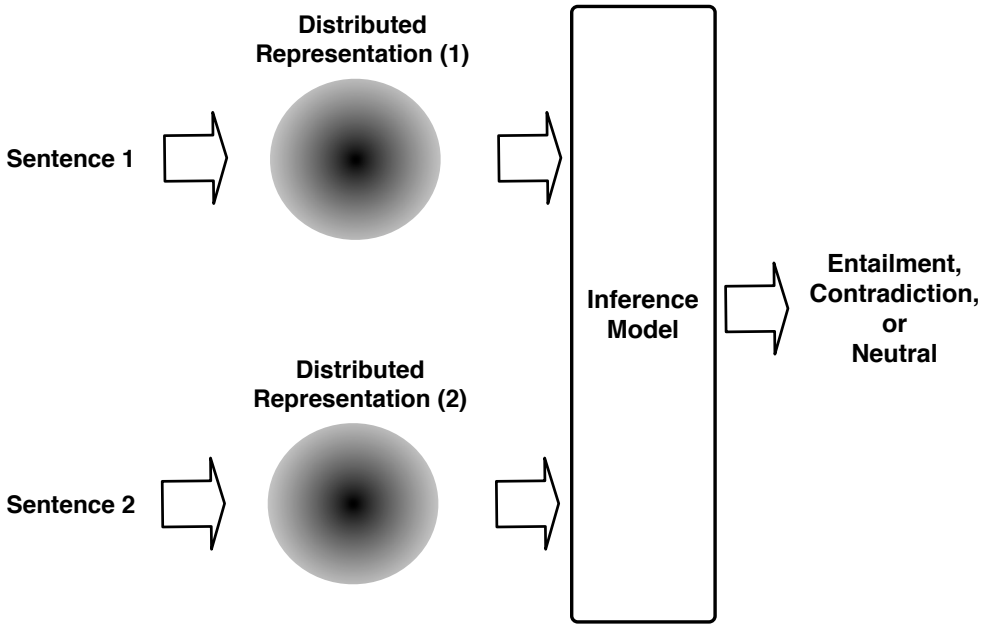


Figure 1.3: Distributed Representations used in a Natural Language Inference Reasoning Flow

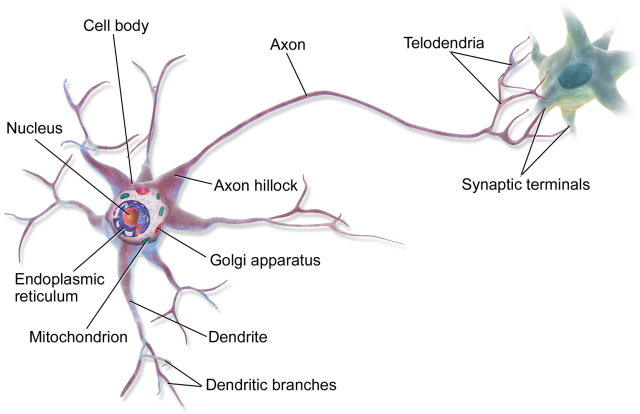
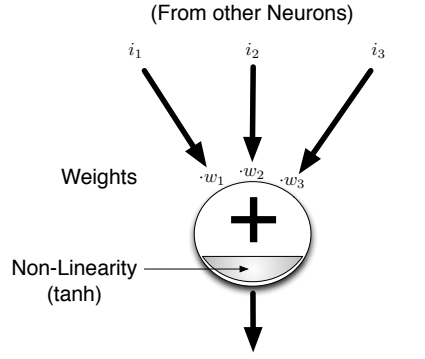


Figure 1.4: Biological Neuron.



$$out = tanh(w_1 i_1 + w_2 i_2 + w_3 i_3)$$

Figure 1.5: Simple ANN Cell.

Chapter 2

Background

2.1 Symbolic Meaning

By themselves, symbols allow us to express completely equivalent entities, but a reference system is necessary to gain further insight to compare two different symbols. For example, the symbols boy, dog, and rock are all recognized as being different from each other, but there is not an inherent property of the symbols that tells us that two are alive and the other inanimate. Similarly, hop and jump are different symbols, but their similarity is not directly represented.

The words in a sentence can be thought of as discrete symbols. A count of the number of word symbols in a document can be used to generally describe the document and compare it to others, and is the basis for basic search algorithms like Google. This simple form of representing a document is called a ”bag of words”, since we could cut the document into its constituent words and jumble them up before counting them.

But words are ambiguous. Consider the sentence:

I saw a man on a hill with a telescope.

Does the word *saw* mean that we viewed him with our eyes, or did we cut him with a saw? Even when words are disambiguated, sentences can almost always be interpreted in different ways. Did we use the telescope to see the man, or is he the one with the telescope? Questions like these are usually resolved using common sense, or the context of the sentence.

The importance of word order for understanding meaning, is illustrated in the example below, taken from [Landauer et al., 1997]. Sentences (1-a) and (1-b) contain exactly the same set of words

but their meanings are very different.

1-a: It was not the sales manager who hit the bottle that day, but the office worker with the serious drinking problem.

1-b: That day the office manager, who was drinking, hit the problem sales worker with a bottle, but it was not serious.

Even though word order can be very important, models which ignore word order can perform surprisingly well, as we will show in Chapter 6.

2.1.1 Lexical Resources

Lexical resources are used to group word symbols together based on meaning. Various lexical resources have been developed over the years, and four of the most important ones have been linked together in a project called SemLink [Palmer, 2009, Loper et al., 2007]. Semlink includes:

- PropBank: [Palmer et al., 2005a] A corpus of one million words of English text, annotated with argument role labels for verbs; and a lexicon defining those argument roles on a per-verb basis.
- VerbNet: [Schuler, 2005] A lexicon that groups verbs based on their semantic/syntactic linking behavior.
- FrameNet: [Baker et al., 1998] A lexicon based on frame semantics.
- WordNet: [Miller et al., 1985, Fellbaum, 1998] A lexicon that describes semantic relationships (such as synonymy and hyperonymy) between individual words.

2.2 Syntactic Parsing

Syntactic parsing of natural language sentences is a central task in natural language processing (NLP) which is an important step for composing language into meaning. The explicit use of syntax has been an important component for many tasks, such as relation extraction, semantic role

labeling (Gildea and Palmer, 2002) and paraphrase detection (Callison-Burch, 2008).

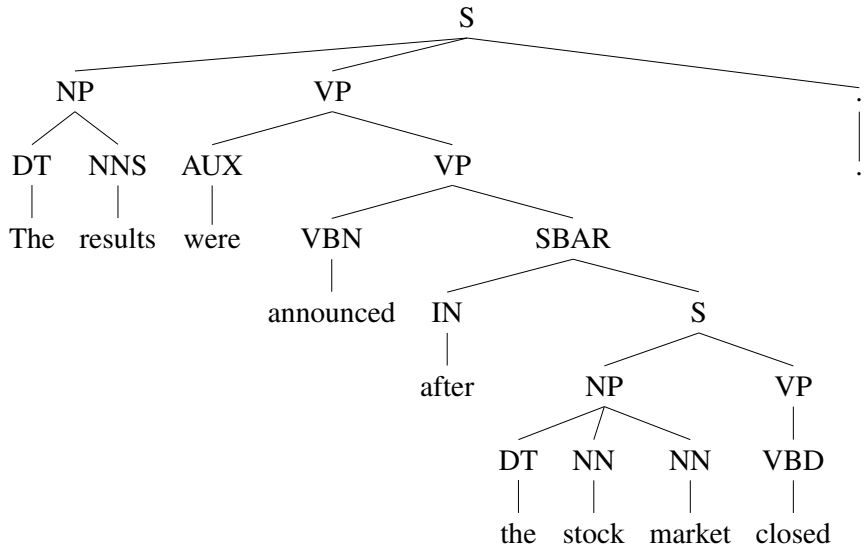


Figure 2.1: Tree for Charniak Parse Tree

There are two dominant ways to describe syntactic structure. The phrase-based, or constituency based parse, which represents how words are combined to create phrases is shown in figure 2.1. The dependency parse, which concentrates more on the relationship of words to each other in a sentence, is shown in figure 2.2. Neural networks can be used create either form [Chen and Manning, 2014, Pei et al., , Durrett and Klein, 2015, Nguyen et al., 2017], but more effort has been applied recently towards the dependency parse, probably because it is applicable to a larger group of languages and a dependency parser is easier to produce. We use a dependency parser to source the semantic role labeler in Chapter 3.

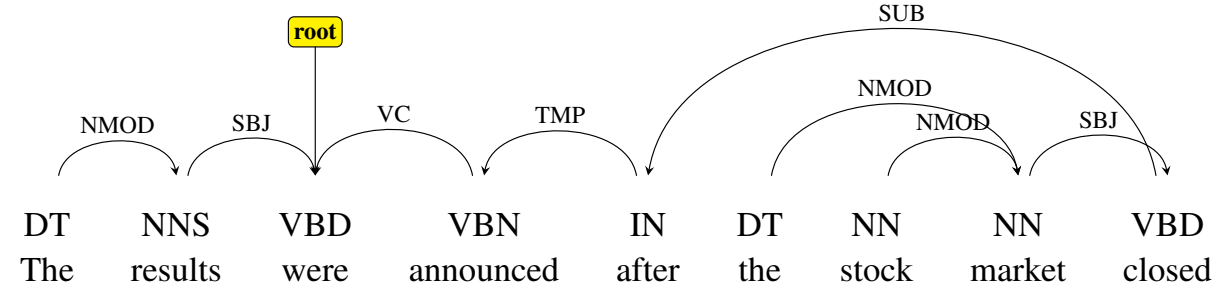


Figure 2.2: Example Dependency Parse, with POS Tags

2.3 Semantic Role Labelling

Semantic role labeling (SRL) is a form of shallow semantic parsing which identifies the predicate-argument structure in a sentence. The Proposition Bank Project [Palmer et al., 2005a] added predicate-argument information called semantic role labels to the syntactic structures of the Penn Treebank corpus, which spawned extensive machine learning activity meant to automatically discover these relationships. Semantic role labelling is useful as an intermediate step in a large number of semantic tasks, such as such as question-answering [Surdeanu et al., 2003, Moschitti et al., 2003, Shen and Lapata, 2007], text categorization [Persson et al., 2009], and inference [Emanuele et al., 2013].

2.4 Abstract Meaning Representation

Semantic parsing is the process of extracting meaning from text and expressing it in some sort of common semantic framework. An important consideration has been the definition of the framework - how can the ideas in a sentence be expressed in a general, consistent manner? Traditional representations have expressive shortcomings which have motivated the development of a general graph based representation, called Abstract Meaning Representation (AMR) [Banarescu et al., 2012].[1]

AMR graphs represent semantic information as a set of concepts (nodes) connected by relations (edges). AMR concepts incorporate a number of NLP tasks, including named entity recognition, [Nadeau and Sekine, 2007], word sense disambiguation [Banerjee and Pedersen, 2002] and lemmatization. AMR makes extensive use of PropBank framesets [Kingsbury and Palmer, 2002, Palmer et al., 2005a] to capture the relations between verbs and their arguments. In addition, AMR requires normalizing temporal expressions [Verhagen et al., 2010, Strötgen and Gertz, 2010].

Abstract meaning representation is a graphical semantic form which allows the expression

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

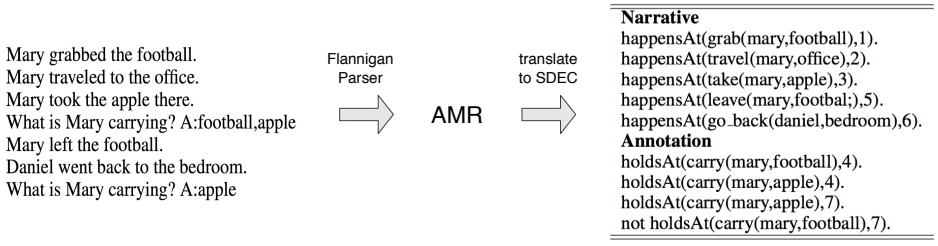
of the meaning of text as concepts and their relations. AMR graphs can be printed as regular text (see figure 2.3b), or drawn as graphs (figure 4.1), which are generally understandable by humans (although not as easily as the original text).

Corpora have been constructed with sentence-AMR pairs, which can then be used to train an AMR parser. The parser described in section 4 uses recurrent neural networks in its underlying architecture to learn how to parse sentences into AMR's.

2.4.1 AMR as a source for semantic tasks

AMR is designed to contain sufficient semantic content for downstream processing, and a number of tasks have been developed.

AMR has successfully been used as the front end of system designed to test the natural language understanding ability of an intelligent agent by [Mitra and Baral, 2015]. [Weston et al., 2015] observed that many of the question answering tasks currently proposed for testing language understanding are too domain-specific, or require competence at so many subtasks (deduction, use of common-sense, abduction, coreference etc.), making it difficult to interpret the results from different approaches. They have introduced a set of 20 tasks for text understanding and reasoning which are less intertwined, called babi. Each task is noiseless and provides a set of 1000 training and 1000 test QA sets for problems which a human can potentially solve with 100% accuracy. They use the Flanigan parser to create AMRs, and then use a simple translation of AMR to Simple Discrete Event Calculus (SDEC). Here's an example from Task 8 (list/sets):



Mary grabbed the football.
Mary traveled to the office.
Mary took the apple there.
What is Mary carrying? A:football,apple
Mary left the football.
Daniel went back to the bedroom.
What is Mary carrying? A:apple

The system in [Mitra and Baral, 2015] uses the [Flanigan et al., 2014] parser to generate AMR:

```
(g / grab
    :ARG0 (p / person
        :name (m / name :op1 Mary ))
    :ARG1 (f / football ))
```

and then translates to Simple Discrete Event Calculus (SDEC):

Narrative
happensAt(grab(mary,football),1).
happensAt(travel(mary,office),2).
happensAt(take(mary,apple),3).
happensAt(leave(mary,footbal;),5).
happensAt(go_back(daniel,bedroom),6).
Annotation
holdsAt(carry(mary,football),4).
holdsAt(carry(mary,apple),4).
holdsAt(carry(mary,apple),7).
not holdsAt(carry(mary,football),7).

The system achieves nearly 100% on all 20 tasks, outperforming both Memory Network [Weston et al., 2014] and the Dynamic Memory Network [Kumar et al., 2015], two recent advanced neural network based models.

The graphical form of AMR can be useful and intuitive for composition and summarization. [Liu et al., 2015] present early results for their investigation into using AMR as part of an abstractive summarization tool. Abstractive summarization can use paraphrasing to describe a text, as opposed to extractive summarization which just selects small subsets from it. They envision a pipeline consisting of: i) parse sentences to AMR. ii) merge sentences into a single graph. iii) resolve common meanings to create a source graph. iv) select a portion of the source graph to generate a summary graph. v) use an AMR to text generator to create the summary. They assume the eventual availability of AMR to text generators, but since they don't exist yet, they heuristically generate a bag of words representation of their summary for comparison instead.

[Kai and Grishman, 2015] use a set of carefully crafted experiments to investigate the benefit of using information from AMR as features for event detection. They studied adding information

from AMR about parent and sibling nodes, and found that the relation connecting sibling nodes, and the combined word with relation of each sibling node, provide good evidence for word sense disambiguation of the trigger candidate. They added AMR features to a maxent system and showed a 2.1% improvement in F1 score for the ACE task.

2.5 Distributed Word Vectors

Prior to about 2012, most NLP systems treated words as atomic units, as indices into a vocabulary. As machine learning techniques progressed, it became possible to train more complex models on large data sets, and methods of creating dense word vectors, based on word context, emerged. Notable contributions to this progression include ([Hinton et al., 2006], [Bengio et al., 2007] and [Weston et al., 2012]).

Word vectors can be generated using unsupervised training, on large corpora, such as wikipedia. For example, the word vectors generated by [Collobert et al., 2011] were created using a pairwise ranking approach (([Schapire and Singer, 1998]). The goal of the neural network is to compute a higher score when given a correct phrase than when given an incorrect phrase. The network is first given the actual windowed words from the training corpus, then the same words with the center word replaced by a nonsensical word. This generates a score for the correct and incorrect phrases, which can be used as the ranking criteria:

$$\Theta \mapsto \sum_{x \in X} \sum_{w \in D} max\left\{0, 1 - f_\Theta(x) + f_\Theta(x^{(w)})\right\} \tag{2.1}$$

where X is the set of all possible text windows with d_{win} words coming from the training corpus, D is the dictionary of words, and $x^{(w)}$ denotes the text window obtained by replacing the central word in the text window by the word w.

These word vectors (also called word representations, or embeddings) are then used as input for further processing. A commonly used approach, described in [Collobert et al., 2011], is to use vectors which have been pre-trained, using unsupervised training, to initialize a network. The representations can then be fine-tuned using supervised training to execute a specific task.

There is an almost unlimited amount of free, untagged information available on the web. A method of training just the word representations from untagged databases has been very successfully applied to create a starting set of vectors that can be used to initialize a network, which is then fine-tuned with supervised training to execute a specific task. By ”pre-training” these word representations using large amounts of untagged text, very informative word relationships can be inexpensively extracted, and later used as the starting point for task specific application learning, see for example [Hinton et al., 2006], [Bengio et al., 2007] and [Weston et al., 2012].

The word representations generated by [Collobert et al., 2011] were created using a pairwise ranking approach (([Schapire and Singer, 1998]). The goal of the neural network is to compute a higher score when given a correct phrase than when given an incorrect phrase. The network is first given the actual windowed words from the training corpus, then the same words with the center word replaced by a nonsensical word. This generates a score for the correct and incorrect phrases, which can be used as the ranking criteria:

$$\Theta \mapsto \sum_{x \in X} \sum_{w \in D} max \left\{ 0, 1 - f_\Theta(x) + f_\Theta(x^{(w)}) \right\} \tag{2.2}$$

where X is the set of all possible text windows with dwin words coming from the training corpus, D is the dictionary of words, and $x^{(w)}$ denotes the text window obtained by replacing the central word in the text window by the word w.

The collection of 130K word representations from [Collobert et al., 2011] was pre-computed and published by the authors. Each representation is a vector of fifty real numbers. This model was created by running an unsupervised neural network on the entire English Wikipedia, which took seven weeks. The text was tokenized using the Penn Treebank tokenizer script, which resulted in a dataset containing about 631 million words. The most common words from a Wall Street Journal corpus were selected from the model, and another 30,000 of the most common words from a Reuters corpus were also selected.

2.6 Word Embedding Visualization

Unsupervised pre-trained word representations have been noted by [Mikolov et al., 2013b] and others to have very interesting linguistic structure. To visualize this structure we analyzed the embeddings using an approach based on Network Theory. Each word can be considered to be a vertex in a 50 dimensional hyperspace, and the Euclidean distance between words provides a similarity measure which is assigned to each edge in a network. This network can be *grown* by applying edges to the word vertices, starting from the closest words, and adding edges in order of increasing distance. The resulting structure is shown in Figure 2.4. The list of words in each cluster, shown in Table 2.1 contains a readily visible degree of semantic and syntactic similarity within each group.

The vectors which define the word representations (vertices) within these components are plotted together in Figure 2.5. Distinct bands of similar strength show the common traits between vectors which are responsible for the similarities which cause them to be grouped together within the network.

2.6.1 Sentence Vectors

A significant amount of research effort is currently being applied towards generating vectors to represent sentence meaning, and finding ways to use them as the basis for more sophisticated semantic applications. The emergence of sentence vectors as a common representation form benefits NLP research by allowing multi-task sentence representations to be produced independently from tasks which use them as input.

Sentence vector representations can be created for use in specific tasks, for example, by training an LSTM network using the sequence of word vectors representing the sentence as an input using a task specific training objective. Because the resulting vectors are influenced heavily by the training objective, they will be overfitted to the particular task.

Multi-task vectors on the other hand can be computed once, then and archived, and refer-

enced as needed. They project meaning into a common space, promoting data sharing and re-use. A straightforward method of creating multi-task sentence vectors is to average the word vectors for the sentence. Even though this ”bag of words” method discards the important information associated with word order, it can be surprisingly effective for some tasks, such as sentiment analysis.

Systems which create sentence vectors using supervised learning require large amounts of labelled training data, which can be expensive to create. Unsupervised learning techniques can make use of large and easily obtainable unlabelled datasets. Autoencoders which encode sentences into representations, and then try to decode them back into the original sentences have been successfully used[Dai and Le, 2015].

Skip-thought vectors [Kiros et al., 2015] are generated using a training objective which can be used for unsupervised training of sentence vectors. Similar to the word2vec skip-gram algorithm [Mikolov et al., 2013a], which predicts neighboring words, sentence representations are trained by having the system try to predict the previous or next sentences within book passages.

2.6.2 Sentence Vector Sourced Tasks

Some machine learning NLP systems can use a single vector representation as input. For example, sentiment analysis can be accomplished by first using a model to compose a distributed vector representation of the words in a sentence, then classifying the sentiment based on this representation.

Many other NLP tasks take the form of examining the semantic content of two sentences, and a common approach for these kinds of tasks is to first create a vector for each sentence and then process the two vectors.

Answer Selection involves examining a question sentence and comparing with various candidate answers to find the closest match. It can be accomplished by converting the two sentences into vectors which are then fed to an answer matching function to determine the degree of match [Tan et al., 2015].

Paraphrase Identification compares semantic similarity, which can be done by comparing

two sentence vectors as in [Socher et al., 2011a].

Semantic Inference determines the semantic relationship between a premise sentence and a hypothesis sentence. A classifier determines whether the premise (i) contradicts the hypothesis, (ii) is not related, or (iii) is entailed (implied) by the hypothesis [Bowman et al., 2015].

2.7 Neural Networks

2.7.1 Forward

Neural networks are composed of layers. The parameters for each layer are referred to as Θ, which includes a matrix of weights, W, and a vector of bias terms b. Each layer's output, prior to the activation function, can be calculated from the previous layer's activation output and parameters.

$$f_{\Theta}^{l} = W^{l-1} f_{\Theta}^{l-1} + b^{l-1} \tag{2.3}$$

2.7.2 Word Level Likelihood

An objective function is attached to the outputs of the network in order to provide a framework for what the network produces. A function which produces higher outputs, or scores, for good results, and low outputs for bad results is desired, and can be used to train the network to achieve that objective. One objective function is for example a mean squared error. A more commonly used, probabilistic objective is log-likelihood. To maximize a log-likelihood objective, the predictions of the network are converted to properly normalized log-probabilities using a softmax function ([Bridle, 1990]), which turns a linear regression into a logistic regression. This is sometimes referred as stacking a softmax function on top. This will coerce the network into producing normalized probabilities, which are perfect for classification problems where we are trying to figure out the most probable class to assign to the input.

Using the notation from [Collobert et al., 2011], the score of the system for tag_i, given a Θ and training example x, is $[f(\Theta, x)]_i$. The training example x is composed of the words of a sentence and some limited features extracted from the words, which are specific to the model. For Word Level Likelihood, a conditional tag probability given the training sample and the system parameters, $p(i|x, \Theta)$ can be computed as:

$$p(i|x, \Theta) = \frac{e^{|f(\Theta,x)|_i}}{\sum_k e^{|f(\Theta,x)|_k}} \tag{2.4}$$

Defining the log add operator as

$$\underset{i}{logadd}(z_i) = log(\sum_i e^{z_i}) \tag{2.5}$$

To simplify some of the following descriptions the reference to x will be omitted, so that the output of the network for a given tag i, $|f(\Theta, x)|_i$, will be shortened to $f[\Theta]_i$.

It's mathematically convenient to maximize the log of the probability, called log likelihood. The log-likelihood of one training example (x,y) can then be expressed as:

$$logp(y|x, \Theta) = f[\Theta]_y - \underset{j}{logadd}(f[\Theta]_j) \tag{2.6}$$

The softmax training criterion is also referred to as cross-entropy. It doesn't consider the often important relationships between words in the sentence, so a sequence detector (such as a Viterbi detector) is commonly added to the system to enforce dependencies between predicted tags.

2.7.3 Back Propagation

Back propagation is an important algorithm used to train the weights of the system. The objective is to choose parameters which will maximize the likelihood that the system produces

the desired output. Back propagation does this by first calculating a cost function (which is the log of the inverse of the probability discussed in section 2.7.2), then finding the partial derivatives of each parameter with respect to this cost function. By subtracting a fraction of this derivative, or gradient, from the parameters Θ while training, Θ are coaxed into a set of values which cause $f[\Theta]$ to produce values with minimum cost (maximum likelihood). Note that there are many such configurations, we are only looking for one during a single training session. This process is known as stochastic gradient descent, and is used to train the models described here.

2.7.4 Word Level Likelihood Gradients

Backpropagation can be based on the Word Level Log-Likelihood gradients such as described in section 2.7.2, or it can be based on Sentence Level Log-Likelihood gradients.

Backpropagation works by first computing the partial derivatives of the inputs of the neurons (after the sum is calculated, but before the activation is applied). Once these so called δ terms are computed, the gradients for parameters can be calculated from them. Calculation of the Cost function with respect to the output gradients of the network, will now be described.

If y is the true tag for a given word, maximizing 2.6 corresponds to minimizing:

$$C(f_\Theta) = \underset{i}{logadd}[f_\Theta]_j - [f_\Theta]_y \tag{2.7}$$

So the gradient w.r.t. f_Θ is

$$\frac{\partial C}{\partial [f_\Theta]_i} = \frac{e^{[f_\Theta]_i}}{\sum_k e^{[f_\Theta]_k}} - 1_{i=y} \; \forall i \tag{2.8}$$

Backpropagation then proceeds backwards, from output to input of the network, to calculate the rest of the necessary partial derivatives of the cost function with respect to inputs:

$$\frac{\partial C}{\partial f_\Theta^{l-1}} = \left[W^{l-1}\right]^T \frac{\partial C}{\partial f_\Theta^{l}} \tag{2.9}$$

Finally, the gradients of the Θ parameters, W and b, can be calculated.

$$\frac{\partial C}{\partial W^{l-1}} = \left[\frac{\partial C}{\partial f_{\Theta}^{l}}\right]\left[f_{\Theta}^{l-1}\right]^{T} \tag{2.10}$$

$$\frac{\partial C}{\partial b^{l-1}} = \left[\frac{\partial C}{\partial f_{\Theta}^{l}}\right] \tag{2.11}$$

2.8 Viterbi Forward

The Viterbi algorithm input is a matrix formed by joining column vectors created by the neural network. Each column vector consists of scores for all possible tags, where a score represents the unnormalized log probability that a tag corresponds to the word. The tags are considered to be hidden, or latent, states. The choice of log-likelihood cost functions for training the neural network coerces the network into producing unnormalized log probabilities which can be converted to normalized probabilities by using the softmax equation (equation 2.6).

The neural network output matrix which is passed on to the Viterbi detector is called f_{Θ}, and it contains elements $[f_{\Theta}]_{i,t}$ for every tag i and word t.

The Viterbi algorithm is initialized with a learned set of weights per tag (the I matrix), and computes the log-likelihood of transitioning from each state to the next by applying a learned set of weights from a square transition matrix A, with N^2 elements, where N is the number of tags.

By considering all possible state transitions for all words, the Viterbi algorithm evaluates all possible combinations of states for the sentence (a very large number, N^T, where T is the number of words in the sentence). It finds the most likely path by selecting the most likely path at each state along the way, and computes the most likely path in linear time.

Let $[j]_1^T$ be the set of all N^T possible paths which can describe a tag sequence. $[x]_1^T$ is the input sentence, composed of T words. The viterbi parameters for the transition matrix and the initialization matrix are grouped together with the other system parameters, Θ, and the entire group of parameters is called $\widetilde{\Theta}$.

The score of a path $[i]_1^T$ is the sum of the transition scores and the network scores, (adding logs instead of multiplying probabilities).

$$s([x]_1^T, [i]_1^T, \widetilde{\Theta}) = \sum_{t=1}^{T} ([A]_{[i]_{t-1},[i]_t} + [f_\Theta]_{[i]_t,t}) \tag{2.12}$$

The Viterbi algorithm finds the best sequence score s by computing the best path leading up to each state in the sequence and discarding paths which are suboptimal. By backtracking the states of the best path through the "trellis", the best path states (which result in the best score) can be computed:

$$\underset{\forall [j]_1^T}{argmax} \;\; s([x]_1^T, [j]_1^T, \widetilde{\Theta}) \tag{2.13}$$

2.9 Viterbi Cost Function (SLL)

The Viterbi cost function is based on Sentence Level Likelihood and is similar to equation 2.6, except the reference path score must be normalized by using the sum of the exponential of all path scores (the sum of unnormalized probabilities for **all** possible paths, instead of for all possible tags).

$$logp([y]_1^T | [x]_1^T, \widetilde{\Theta}) = s([x]_1^T, [y]_1^T, \widetilde{\Theta}) - \underset{\forall [j]_1^T}{logadd}(s([x]_1^T, [j]_1^T, \widetilde{\Theta})) \tag{2.14}$$

A recursion, described in [Rabiner, 1989] and specified in [Collobert et al., 2011], provides a method of computing the second term in equation 2.14. An intermediate vector, δ, is calculated, which will contain the unnormalized log probability that any path through the trellis will pass through a particular state k for the particular word t. The $delta$ vectors have a dimension of N, the number of tags, and they will be used for training viterbi and optionally network parameters as will be described later.

Initialize δ_0 for all states k, using the Viterbi initial State Log Likelihood matrix, I:

$$\delta_t(k) = f_\Theta[x]_{k,0} + [I]_k) \; \forall k \tag{2.15}$$

Recursively compute the δs for words 1 through T:

$$\delta_t(k) = f_\Theta[x]_{k,t} + \underset{i}{logadd}(\delta_{t-1}(i) + [A]_{i,k}) \; \forall k \tag{2.16}$$

Followed by the termination:

$$\underset{\forall [j]_1^T}{logadd}(s([x]_1^T, [j]_1^T)) = \underset{i}{logadd}(\delta_T(i)) \tag{2.17}$$

Which allows us to solve 2.14 for the log-likelihood in linear time.

2.10 Sentence Level Likelihood Gradients And Viterbi Training

If $[y]_1^T$ is the expected tag path for a sentence (from training data),

$$C(f_{\widetilde{\Theta}}) = \underset{\forall [j]_1^T}{logadd}(s([x]_1^T, [j]_1^T, \widetilde{\Theta})) - s([x]_1^T, [y]_1^T, \widetilde{\Theta}) \tag{2.18}$$

the second half of equation 2.18 is the Viterbi score of the expected path. From equation 2.12,

$$s([x]_1^T, [y]_1^T, \widetilde{\Theta}) = \sum_{t=1}^{T} ([A]_{[y]_{t-1},[y]_t} + [f_\Theta]_{[y]_t,t}) \tag{2.19}$$

(The first half of equation 2.18 is log of the sum of all possible tag paths.)

We want to calculate the gradients for the Viterbi transition matrix $\frac{\partial C}{\partial [A]_{i,j}}$ and the gradients of the inputs $\frac{\partial C}{\partial [f_\Theta]_{i,t}}$, which can be optionally used for Sentence Level Log-Likelihood calculations. From [Collobert et al., 2011], these can be calculated with a recursive procedure:

Initialize the gradients to zero.

$$\frac{\partial C}{\partial [A]_{i,j}} = 0, \forall i, j \text{ and } \frac{\partial C}{\partial [f_\Theta]_{i,t}} = 0, \forall i, t. \tag{2.20}$$

accumulate gradients over the second part of equation 2.18, $s([x]_1^T, [y]_1^T, \widetilde{\Theta})$, by traversing the expected path $[y]_1^T$,

$$\frac{\partial C}{\partial [A]_{[y]_{t-1},[y]_t}} += 1, \forall t \text{ and } \frac{\partial C}{\partial [f_\Theta]_{[y]_t,t}} += 1, \forall t. \tag{2.21}$$

Defining the first part of equation 2.18 as C_{logadd},

Use recursion to compute terms $\frac{\partial C_{logadd}}{\partial \delta_t(i)}$. First, initialize $\frac{\partial C_{logadd}}{\partial \delta_T(i)}$, using softmax:

$$\frac{\partial C_{logadd}}{\partial \delta_T(i)} = \frac{e^{\delta_T(i)}}{\sum_k e^{\delta_T(k)}} \forall i \tag{2.22}$$

Then traverse the trellis from T-1 to 1 (backwards) to iteratively compute the remaining $\frac{\partial C_{logadd}}{\partial \delta_t(i)}$:

$$\frac{\partial C_{logadd}}{\partial \delta_{t-1}(i)} = \sum_j \frac{\partial C_{logadd}}{\partial \delta_t(j)} \frac{e^{\delta_{t-1}(i)+[A]_{i,j}}}{\sum_k e^{\delta_{t-1}(k)+[A]_{k,j}}} \tag{2.23}$$

Using both the forward δ_t and the backward $\frac{\partial C_{logadd}}{\partial \delta_t(i)}$, at each step we can iteratively update both the $\frac{\partial C}{\partial [f_\Theta]_{[y]_t,t}}$ (which can optionally be used to back propagate the neural network):

$$\frac{\partial C}{\partial [f_\Theta]_{i,t}} += \frac{\partial C_{logadd}}{\partial \delta_t(i)} \tag{2.24}$$

and the gradient terms of the transition score matrix A:

$$\frac{\partial C}{\partial [A]_{i,j}} += \frac{\partial C_{logadd}}{\partial \delta_t(j)} \frac{e^{\delta_{t-1}(i)+[A]_{i,j}}}{\sum_k e^{\delta_{t-1}(k)+[A]_{k,j}}} \tag{2.25}$$

This is described in [Rabiner, 1989] as calculating the ξ_t, and the sum of these quantities can be interpreted to be the score for transitioning from state i to state j.

The gradient for the initial score matrix I can be computed with one last step of the algorithm (or by considering it to be the first vector of an N by N+1 sized gradient matrix).

2.10.1 Convolutional Neural Networks

Convolutional neural networks are commonly used for computer vision tasks and for the initial stages of speech recognition, but can also be used for the initial stages of natural language text processing tasks. They work by considering a window of data inputs, and create a set of local features surrounding each item of interest, in our case, words. By successively multiplying a convolutional kernel to a sentence, which amounts to multiplying by a matrix of weights, a series of such features surrounding words can be created, which are then further processed to achieve the desired task. Chapter 3 goes into detail about how such as system can be used for semantic role labelling.

2.10.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) have been used very successfully to process sequences of data, such as for speech recognition [Graves et al., 2013b, Hannun et al., 2014] or for natural language. For processing sentences, they work by creating a hidden representation for each word based on the current word and the hidden representation created so far. As such they maintain a memory of words as they are processed, and are capable of learning long term relationships, which is important for language processing.

RNNs maintain a hidden vector $\mathbf{h}$, which is updated at time step t as follows:

$$\mathbf{h}_t = \tanh(\mathbf{W} * \mathbf{h}_{t-1} + \mathbf{I} * \mathbf{x}_t) \tag{2.26}$$

where $\tanh$ is the hyperbolic tangent function, $\mathbf{W}$ is the recurrent weight matrix and I is a projection matrix.

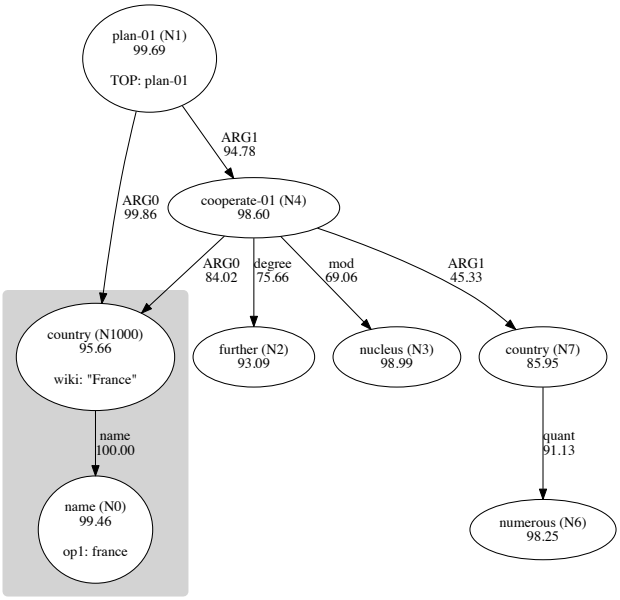
One problem with simple recurrent networks is that they are difficult to train because the gradients ”explode” or ”vanish” over long sequences. The currently accepted solution to this problem is to provide learnable information gates which help regulate the processing of memory over time, which leads us to Long Short Term Memory networks.

2.10.3 Long Short Term Memory Networks

Long Short Term Memory (LSTM) networks address the vanishing gradient problem by incorporating gating functions into their state dynamics [Hochreiter and Schmidhuber, 1997]. At each time step, an LSTM maintains both hidden vector $\mathbf{h}$ and a memory vector $\mathbf{m}$ responsible for controlling state updates and outputs. The computation at time step t is defined as follows [Kalchbrenner et al., 2015]:

$$\begin{aligned}
\mathbf{g}^u &= \sigma(\mathbf{W}^u * \mathbf{h}_{t-1} + \mathbf{I}^u * \mathbf{x}_t) \\
\mathbf{g}^f &= \sigma(\mathbf{W}^f * \mathbf{h}_{t-1} + \mathbf{I}^f * \mathbf{x}_t) \\
\mathbf{g}^o &= \sigma(\mathbf{W}^o * \mathbf{h}_{t-1} + \mathbf{I}^o * \mathbf{x}_t) \\
\mathbf{g}^c &= \tanh(\mathbf{W}^c * \mathbf{h}_{t-1} + \mathbf{I}^c * \mathbf{x}_t) \\
\mathbf{m}_t &= \mathbf{g}^f \odot + \mathbf{g}^u \odot \mathbf{g}^c \\
\mathbf{h}_t &= \tanh(\mathbf{g}^o \odot \mathbf{m}_{t-1})
\end{aligned} \tag{2.27}$$

here σ is the logistic sigmoid function, $\mathbf{W}^u, \mathbf{W}^f, \mathbf{W}^o, \mathbf{W}^c$ are recurrent weight matrices and $\mathbf{I}^u, \mathbf{I}^f, \mathbf{I}^o, \mathbf{I}^c$ are projection matrices.



(a) Graphical Representation (annotated with probabilities expressed as percentage.)

```
(p / plan-01
  :ARG0 (c / country :wiki "France"
    :name (n / name :op1 "France"))
  :ARG1 (c2 / cooperate-01
    :ARG0 c
    :ARG1 (c3 / country
      :quant (n2 / numerous))
    :mod (n3 / nucleus)
    :degree (f / further)))
```

(b) Textual representation of AMR (Penman/AMR).

Figure 2.3: Graphical and textual representations of the AMR for the example sentence: ***France plans further nuclear cooperation with numerous countries .***

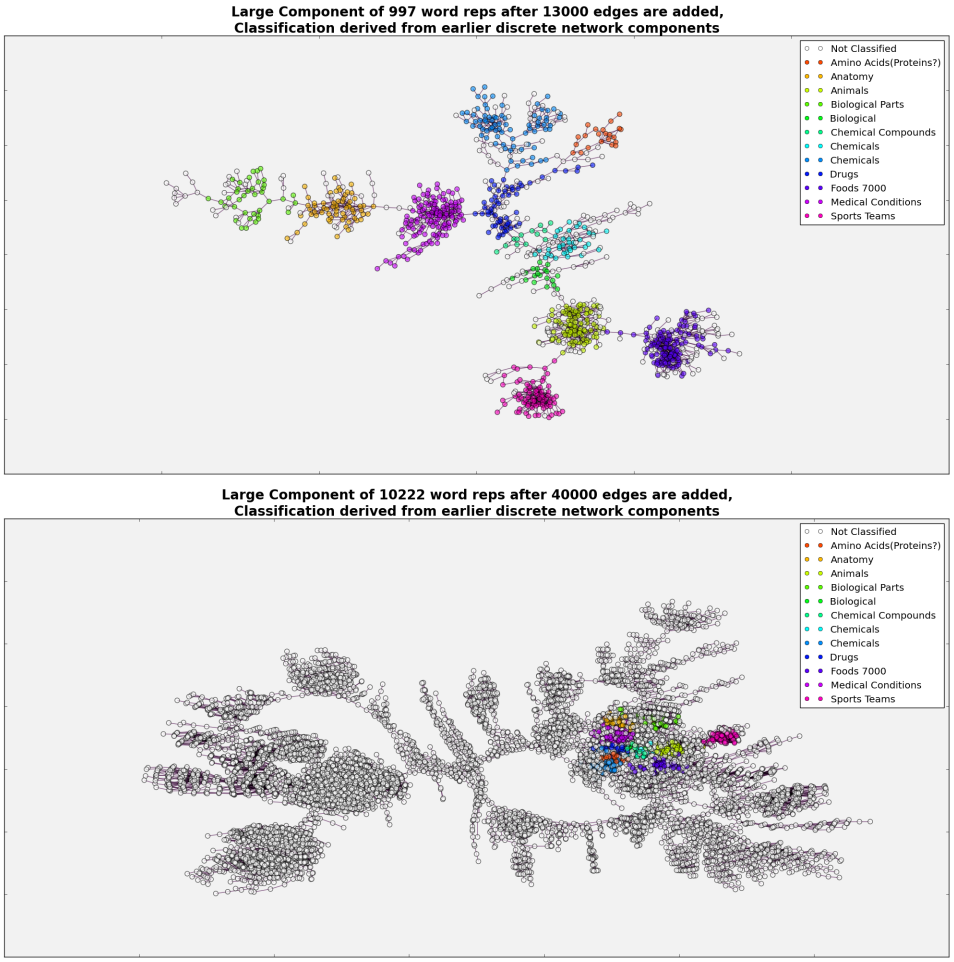


Figure 2.4: The large component as the network is grown for two network states during growth. The structure of the colored groups, corresponding to the selected early distinct components, is especially visible in the top diagram due to the networkx Spring-Layout Visualization algorithm.

Summary	Size	Edges	Words
Amino Acids	22	12200	adenine, aspartate, choline, cysteine, cytosine, dmso, glycerol, guanine, heme, histidine, lipid, lysine, pyridine, pyrimidine, ribose ...
Anatomy	62	9000	abdomen, anus, aorta, cecum, cerebellum, cerebrum, cervix, clavicle, clitoris, cochlea, conjunctiva, cornea, cytoplasm, cytoskeleton, dermis ...
Animals	91	7800	alligators, amphibians, aphids, arthropods, bivalves, boars, centipedes, cephalopods, cetaceans, clams, cockatoos, cockroaches, copepods, corals, cormorants ...
Basic Chemicals	26	8000	arsenide, azide, bromide, carbonate, chlorate, chloride, chlorine, deuterium, fluoride, fluorine, helium, hydrate, hydroxide, hypochlorite, iodide ...
...			
Foods	112	7000	almonds, apples, apricots, avocados, bananas, beans, beetroot, berries, blackberries, blueberries, breadfruit, breads, buckwheat, burgers, burritos ...
Medical Conditions	126	11000	abnormalities, adenocarcinoma, adenoma, anaemia, anemia, arrhythmias, arthritis, atherosclerosis, atresia, bloating, bradycardia, bronchitis, cancers, carcinoma, cholera ...
Sports Teams	74	11000	0ers, aeros, alouettes, astros, badgers, beavers, bengals, bisons, bobcats, braves, broncos, bruins, buccaneers, buckeyes, bulldogs ...

Table 2.1: Selected, representative components during network growth. The size and number of edges added are shown along with the first fifteen words from each component (in alphabetical order). These components were selected because they are eventually joined and become part of the large component examined after 13,000 edges have been added.

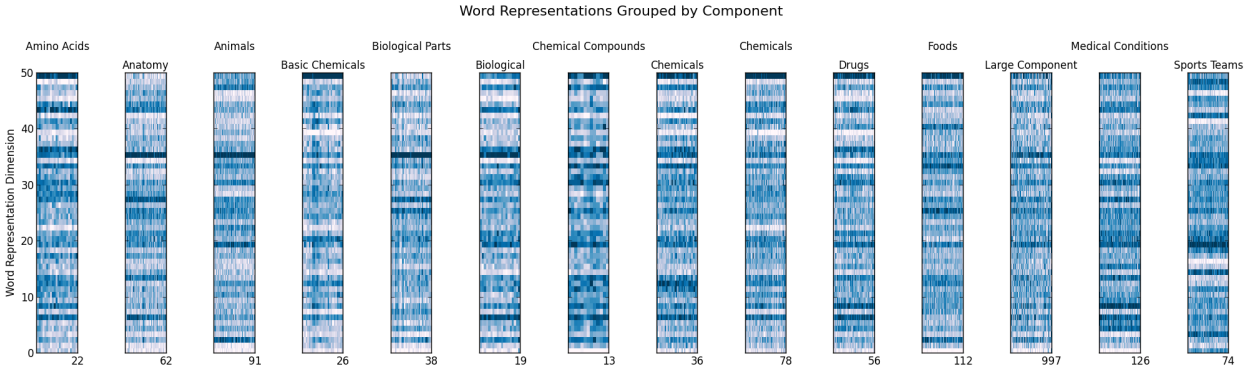


Figure 2.5: Word Representations grouped by Component

Word Representations grouped by Component for the selected, representative early components during network growth. Each representation is a 50 element real vector. Small euclidean distances, which represent the close similarities responsible for component formation, can be seen as light or dark bands across each group.

Chapter 3

Semantic Role Labelling with Dependency Parse Input

Semantic role labeling ([Gildea and Jurafsky, 2002]), the task of identifying and classifying the semantic arguments of verbal and nominal predicates in text, represents one of the most complex NLP tasks to be addressed by supervised machine learning techniques. In the standard supervised approach to building SRL systems, collections of multiway classifiers are trained using annotated corpora such as PropBank ([Palmer et al., 2005b]). In this approach, classifiers are trained using features derived directly from the original source text, as well as from subsequent syntactic and semantic processing.

As reported in several shared tasks ([Carreras and Màrquez, 2004],[Carreras and Màrquez, 2005],[Hajič et al., 2009]), SRL systems trained in this manner can achieve high performance. State-of-the-art systems employ classifiers such as support vector machines trained with large numbers of relatively complex combinations of features, often combined with re-ranking based on multiple syntactic analyses. Unfortunately, these approaches have a number of non-trivial limitations including the computational cost of the syntactic parsing and the sparse nature of the complex features on which they rely. This latter limitation is particularly critical since it leads to significant degradation in performance when the trained system is applied to texts from new domains.

However, recent results using multilayer neural networks and pre-trained word embeddings have demonstrated high performance using a much smaller number of minimalist features. The architecture described by [Collobert et al., 2011] combines time delay convolutional neural networks ([Waibel et al., 1989]) and pre-trained word representations for a number of NLP tasks. They de-

velop four components and compare their performance to previous benchmarks, one of which is an SRL system which uses features derived from a phrase-structure parse as input, based on the CoNLL 2005 shared task ([Carreras and Màrquez, 2005]).

The work described here adopts the basic architecture from [Collobert et al., 2011] and explores issues related to the use of this architecture in the context of the CoNLL 2009 shared task. In particular, we present Daisy, a system that (1) employs features derived from dependency parse as input, (2) assigns semantic roles to both verbal and nominal predicates, and (3) automatically assigns word senses to the predicates as described in the CoNLL 2009 shared task ([Hajič et al., 2009]).

The following sections will describe the architecture of the Daisy system, present state-of-the-art performance on the CoNLL 2009 shared task, and explore the utility of features derived from dependency parses, including a version of the traditional SRL syntactic path feature.

3.1 Experimental Setup

The CoNLL 2009 shared task consists of identifying the sense and semantic arguments for each given argument-bearing token (predicate). In addition to the words themselves, the training data provides the part of speech, syntactic head, and syntactic dependency relation to the head for each word in the sentence. Table 3.1 shows an example sentence and its representation in the dataset. The PDEPREL and PHEAD features are the head word and dependency relation predicted automatically by a dependency parser. In the example sentence, there are two predicates identified for labeling: *announce*, and *close*. The system should output two arguments for *announce*: *results*:A1 (Object), and *after*:AM-TEMP (Temporal Marker). Similarly, *market*:A1 should be output for the predicate *close*. In addition to role identification, the word sense for each predicate is output, in the example, the expected sense for *announce* is 01, and for *close* is 02.

The training, validation, and evaluation datasets are annotated sentences from the Wall Street Journal. An additional out of domain dataset mostly from the Brown corpus was also supplied. A comprehensive F1 score was generated for both role labels and sense predictions using the provided

eval09.pl perl script.

ID	FORM	LEMMA	PLEMMA	POS	PPOS	FEAT	PFEAT	HEAD	PHEAD	DEPREL	PDEPREL	FILLPRED	PRED	A[announce]	A[close]
1	The	the	the	DT	DT	_	_	2	2	NMOD	NMOD	_	_	_	_
2	results	result	result	NNS	NNS	_	_	3	3	SBJ	SBJ	_	_	A1	_
3	were	be	be	VBD	VBD	_	_	0	0	ROOT	ROOT	_	_	_	_
4	announced	announce	announce	VBN	VBN	_	_	3	3	VC	VC	Y	announce.01	_	_
5	after	after	after	IN	IN	_	_	4	4	TMP	TMP	_	_	AM-TMP	_
6	the	the	the	DT	DT	_	_	8	8	NMOD	NMOD	_	_	_	_
7	stock	stock	stock	NN	NN	_	_	8	8	NMOD	NMOD	_	_	_	_
8	market	market	market	NN	NN	_	_	9	9	SBJ	SBJ	_	_	_	A1
9	closed	close	close	VBD	VBD	_	_	5	5	SUB	SUB	Y	close.02	_	_
10	.	.	.	.	.	_	_	3	3	P	P	_	_	_	_

Table 3.1: CoNLL format SRL Dependency Parse Input Test Sentence Example

3.2 Semantic Role Labeling System

The general block diagrams for the Daisy SRL system are shown in Figures 3.1 and 3.2. The input to the system is a list of words w_i from w_1 to w_n, a list of predicate positions, and dependency parse tree information for the sentence. We treat role labeling and the sense identification as two separate tasks. For each predicate in a given sentence, the Role Subsystem outputs the list of predicted role tags for all words (SRL_i), and the Sense Subsystem outputs the sense tag of the predicate. The system is composed of five major components:

- Word Preprocessing and Word Derived Feature Convolution (Figure 3.3).
- Predicate Position Feature Convolution.
- Word Position Feature Convolution.
- Neural Network and Viterbi (Figure 3.5).
- Predicate Sense Neural Network (Figure 3.6).

3.3 Word Derived Feature Convolution Section

The Word Derived Features and Convolution section, shown in Figure 3.3, is sourced by five features which are derived on a word by word basis.

The upper portion of Figure 3.3 depicts the process of looking up features from the words and parse tree information. The numeric information from the features for each word is concatenated

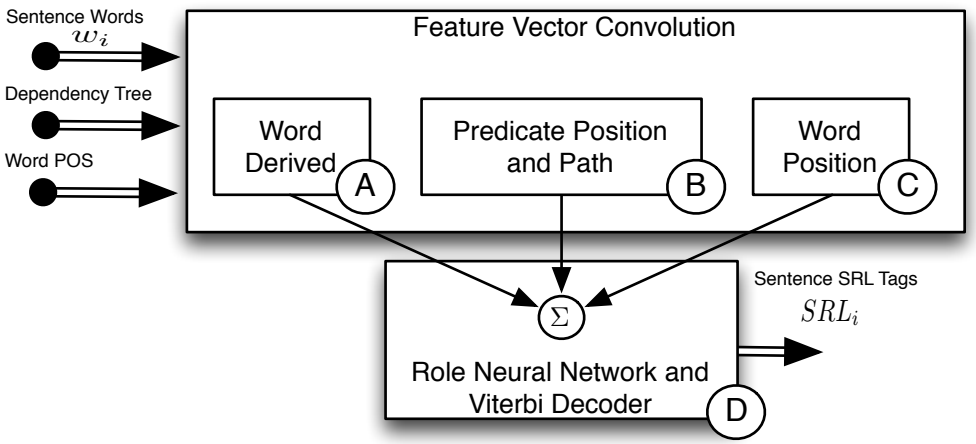


Figure 3.1: Role Subsystem

together to form one long feature vector, shown in the diagram as a multi-shaded set of rectangles. Three words of feature information (the word and its two neighbors) from the Word Derived Feature Vector are multiplied by the the weights and bias of Θ_4 and stored in the Convolved Word Derived Feature Vector, for each word in the sentence. For the default convolution width of 300, this results in a long vector of $300 \cdot n$, where n is the number of words in the sentence.

Each feature lookup table contains an entry for **PADDING**. In order to allow the window to extend beyond boundaries of the sentence for early and late words the Feature Vector is padded with the **PADDING** value from each lookup table. If a feature is in the table, the associated vector is output, otherwise the vector corresponding to the special token **UNKNOWN** is output. The **PADDING** and **UNKNOWN** vectors are trained during supervised training.

To train the word representations from scratch, all except the 0.02% least common words from the training set are added to the lookup table. The remaining words are therefore trained as the UNKNOWN word, which can then be used to represent any word encountered outside the trained word list. For other features, the representation for the most probable token is used as the UNKNOWN representation.

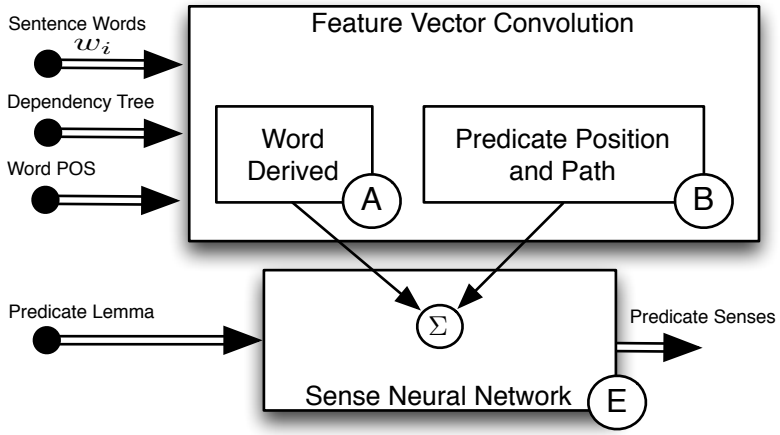


Figure 3.2: Sense Subsystem

The five types of word-derived features tested for the SRL Dependency Parse tagger are:

- Word Embeddings
- Capitalization
- POS tag of word
- Dependency Relation
- POS tag of head

3.4 Word Pre-processing

The input data provided for the CoNLL 2009 task has already gone through some initial tokenizing. This prevents tokenization differences of different systems from influencing the results, which are meant to allow comparison of the SRL tagging architecture itself. The Daisy pre-processor does not split hyphenated input words, so each input word will result in a single pre-processed word. Numeric values are collapsed to the single common **0** token, and words are lower-cased to create a word representation lookup word.

3.5 Word Embeddings

Words are transformed to numeric representations using a lookup table. Like all other feature lookup tables in the system, the word representation vectors can be initialized to small random values to start with, and then trained using the supervised training algorithm.

A method of training the word representations from untagged databases has been very successfully applied to create a starting set of vectors that can be used to initialize a network, which is then fine-tuned with supervised training to execute a specific task. By ”pre-training” these word representations using large amounts of untagged text, very informative word relationships can be inexpensively extracted, and later used as the starting point for task specific application learning, see for example [Hinton et al., 2006], [Bengio et al., 2007] and [Weston et al., 2012].

The word representations used as input to the Daisy SRL System for the experiments described here were generated by [Collobert et al., 2011] and were created using a pairwise ranking approach ([Schapire and Singer, 1998]).

3.6 Capitalization

Prior to lower casing, each word is checked for all capitals, initial capital, any capital, or no capitals, and this criteria is used to lookup a vector (default length 5) from the caps table.

3.7 Predicted Dependency Relation

The PDEPREL column from the training data, shown in table 3.1.

3.8 Predicted POS tag of word and of head

The Predicted Part-of-speech tag is provided in PPOS column of the training data. The head part of speech tag is found by following the PHEAD column and extracting the PPOS column. (see Table 3.1).

3.9 Predicate Position and Path Feature Convolution Section

Predicate Position and optional Path features are extracted on a per word basis and convolved, once per predicate (the outer loop of two).

3.10 Predicate Position Feature

The position of each word relative to the predicate being evaluated is represented by a vector (of length 25), based on distances of ± 12, and distances outside this range are saturated.

3.11 Dependency Path Feature

Information about the path from each word to a given predicate is provided in the Predicate Position Convolution section as a per word feature. Information about the paths is stored in a lookup table as usual. The results from experiments using different types of path information are described in the results section.

Generic Path: The sequence of up and down transitions to traverse the tree from a word to a given predicate is referred to here as the **Generic Path**. The dependency parse trees for each of the two predicates from the example training sentence shown in Table 3.1 are diagrammed in Figure 3.4. The Generic Path for each word is shown in the diagram, above the part of speech tag for the word.

Labeled Path: These are path descriptions which include both the arc direction (Generic Path) and the dependency relation of the arc within the dependency tree. After several rounds of experimentation, we chose to include paths which occur at least five times in the training data, which resulted in about 77K unique path types.

3.12 Word Position Feature Convolution Section

The position of every word with respect to the specific word being evaluated is extracted once per word, per predicate (the inner loop of two). In a similar fashion to the predicate position

feature, the position of each word relative to the word being evaluated is represented by a vector (of length 25), based on distances of ± 12, and distances outside this range are saturated.

3.13 Role Neural Network and Viterbi

Figure 3.5 shows the process of combining the Convolved Feature Vectors, processing with a neural network, and finding the most likely role sequence with a Viterbi detector. Both the Role and Sense neural networks are constructed with a single nonlinear layer followed by an output layer. The parameters for each layer are referred to here as Θ, which includes a matrix of weights, W, and a vector of bias terms b. Each layer's output, prior to the activation function, can be calculated from the previous layer's activation output and parameters.

$$f_{\Theta}^{l} = W^{l-1} f_{\Theta}^{l-1} + b^{l-1} \tag{3.1}$$

The tanh function is used as the nonlinear activation function.

The three Convolved Feature Vectors (diagrammed separately) are summed, then the maximum for each index within each group of 300 is determined. This results in a 300 element vector which will be the input to the Neural Network. A single layer neural network followed by a single output layer is used to create a "score" for each possible role "tag", for the word and predicate being analyzed. After running all words through the system for a single predicate, a matrix of SRL role scores of size $tags \times words$ is created, which will be used as the input to the Viterbi sequence decoder.

3.14 Sequence Decoder (Viterbi)

The Viterbi decoding algorithm input is a matrix which consists of a vector of SRL role scores for each word. The algorithm is initialized with a learned set of weights per tag, and computes the log-likelihood of transitioning from each state to the next by applying a learned set of weights from the transition matrix.

3.15 Predicate Sense Neural Network

Figure 3.6 shows the process of combining the Convolved Feature Vectors, processing with a neural network, and finding the most likely sense for a given predicate. The neural network parameters for the sense subsystem are managed with a lookup table holding parameters for each lemma in the training set that is mapped to multiple senses.

3.16 Sense Labeler Training and Forward Model Creation

Both the Role and Sense subsystems are trained using stochastic gradient descent. A forward pass is first run on the system, during which the indices of the maximum values of the sum of the convolution layer (word-derived and predicate) are saved.

Backpropagation of the Sense Neural Network is based on minimizing a log-likelihood objective:

$$\log p(y|x,\Theta) = f[x,\Theta]_y - \log(\sum_j e^{(f[x,\Theta]_j)}) \tag{3.2}$$

The two Sense and Role subsystems have the same convolution structures (See figures 3.1 and 3.2). Experiments run using a common structure for both tasks resulted in about 0.5% worse performance, so the the systems were kept independent.

A separate neural network was trained for each lemma found in the training data set, and the parameters for each network were stored in a lookup table. This results in very large memory requirements during training, especially since Adagrad ([Duchi et al., 2011] was used to decrease training time. To minimize memory requirements and training time, the sense for lemmas which always train to the same sense in the training data are stored in a dictionary. During forward processing, when a lemma is encountered that was not trained (and therefore is not in the parameter lookup table), the sense from the dictionary is output. If the lemma never occurred during training, it won't be in the dictionary, and the most commonly occurring sense of "01" is output by default.

3.17 Role Labeler Training and Forward Model Creation

During a forward pass, the activation layers and maxIndices are saved and reused during training.

3.18 Cost Calculation

The Viterbi parameters for initial score and transition probabilities are trained using the Sentence Level Log-Likelihood (SLL) cost function.

This cost function is based on Sentence Level Likelihood and is similar to equation 3.2, except the reference path score must be normalized by using the sum of the exponential of all path scores (the sum of unnormalized probabilities for **all** possible paths, instead of for all possible tags). A recursive method, developed in [Rabiner, 1989] and specified in [Collobert et al., 2011], provides an important and efficient means of computing the sum of the exponential of all path scores. An intermediate vector, δ, is calculated, which will contain the unnormalized log probability that any path through the trellis will pass through a particular state k for the particular word t. The δ vectors have a dimension of N, the number of tags, and they are re-used for the gradient calculation during backpropagation.

3.19 Backpropagation

The recursion described in [Collobert et al., 2011] is used to calculate Viterbi delta terms and gradients. The error is then back-propagated through the system in reverse, ending with the feature lookup tables. This is done for each word, for each predicate, requiring two nested loops for training a full sentence. The loop structure makes for long training times, roughly three days on a 2015 compute-optimized AWS core.

3.20 Results

3.21 Benchmark

The best ConLL 2009 English SRL F1 score (labeled "Nugues"), and described in [Björkelund et al., 2009], is used as a comparison benchmark. In this benchmark, 20 features were used for argument identification, including the Dependency Relation Path, and Part of Speech of Dependency Relation Path. A reranker was run on the output of multiple system outputs. The same SRL was later tested using improved dependency parsing in [Björkelund et al., 2010]. [Woodsend and Lapata, 2014] explores text rewriting and compares results with the benchmark, which they accept as the current state-of-the-art.

Table 3.2 compares the benchmark with a complete Daisy system using a labeled path, with a cutoff of 5, and two separate systems for sense and role labels. F1 scores are 0.41% higher for the WSJ Eval dataset, and 2.59% higher for the out of domain (OOD) Brown dataset.

System Description	WSJ F1	Brown F1
Benchmark (CoNLL2009)	85.63%	73.31%
Daisy	86.04%	75.90%

Table 3.2: SRL Dependency Parse Test F1

3.22 Metrics

In all experiments, we strictly followed the standard evaluation procedure of the CoNLL 2009 challenge. A simple validation procedure using the specified validation set was used to choose system hyper parameters, and the provided eval09.pl perl script was used to generate all system F1 scores. The system F1 score is the harmonic mean of precision and recall for both role and sense labels. Since we treated the predicate sense disambiguation and the predicate role assignment tasks as independent, it is interesting to view the performance of the two tasks separately. The predicate sense task requires a label for each given predicate, so a per predicate accuracy was

calculated (SenseAcc). Similarly we generated a role label F1 score (RoleF1) that is independent of the sense labels. These subsystem performance metrics were also calculated on the CoNLL 2009 benchmark results for comparison.

3.23 Incremental Experiments and Results

Feature abbreviations used in the descriptions are shown in Table 3.3.

Abbrev.	Feature Description
W	words, initialized randomly prior to training
C	capitalization
P	Part of Speech
HP	Part of Speech of head word
DR	Dependency Relation
GP	Generic path
TW	words, initialized with pre-trained word representations prior to training
LP5	Labeled paths that occur at least five times in the training data.

Table 3.3: Feature Abbreviations

Starting from a basic configuration of only words (randomly initialized) and capitalization (W,C), the system was incrementally modified to include different feature mixes. Next, simple per-token part of speech was added (W,C,P). Information from the dependency parser is added in two steps, first the head word part of speech and dependency relation (W,C,P,HP,DR), and next the generic path (W,C,P,HP,DR,GP). The word representations were then seeded with the pre-trained representations described in section 3.5 (TW,C,P,HP,DR,GP). Finally, the labeled path was used instead of the generic path, still seeding the words with pre-trained representations (TW,C,P,HP,DR,LP5).

For each system configuration, 12 role subsystems and 8 sense subsystems were trained and tested, using the WSJ development F1 score during training to determine the best model parameter

state. After model generation, the WSJ development scores for different systems don't correlate well with the WSJ eval or Brown scores. For example, models with high development scores don't necessarily correspond to best scoring models for the WSJ or Brown data tests.

The CoNLL2009 results used as benchmarks were given as single data points so statistics are not available.

Figure 3.7 shows the relationship between the development and Evaluation F1 scores, as well as the general performance improvement as features were added.

Tables 3.4 and 3.5 show the statistical performance of the system with WSJ and Brown test data.

For the WSJ (evaluation) dataset, The role subsystem F1 improves much more dramatically than the sense subsystem as POS (+1.52%) and dependency parser information (+1.68%) is added. The mean System F1 score is -0.25% under the benchmark without the pre-trained word representations. Adding the representations boosts performance such that even the lowest scoring systems beat the benchmark, and the mean F1 score is about 0.41% higher.

For the Brown (OOD) dataset, The role subsystem F1 improves significantly with POS and dependency parse information (+2.72%) while the sense subsystem benefits less (0.96%). The role subsystem dramatically improves when pre-trained words are added (2.59%), due in large part to a better ability to handle unseen words. The mean System F1 scores are higher than the benchmark as soon as dependency parser information is supplied, and the F1 is significantly better for the fully populated system (+2.59%).

3.24 Conclusions

We have presented a dependency-based semantic role labeler using neural networks, inspired by [Collobert et al., 2011] and others to reduce the use of hand-crafted features and make use of unsupervised techniques. Experimental evaluations show that our architecture improves the state of the art performance for this task significantly, for both in domain and out of domain test data. A key element of the system's performance is based on the use of features derived from

System Description	SystemF1				RoleF1		SenseAcc	
	Min	Mean	(Δ)	Max	Mean	(Δ)	Mean	(Δ)
Daisy(W,C)	82.86	83.03		83.24	77.47		94.92	
Daisy(W,C,**P**)	83.83	84.12	(+1.09)	84.43	79.00	(+1.52)	95.15	(+0.23)
Daisy(W,C,P,**HP,DR**)	84.46	84.79	(+0.67)	85.10	79.92	(+0.92)	95.29	(+0.13)
Daisy(W,C,P,HP,DR,**GP**)	85.05	85.38	(+0.58)	85.78	80.69	(+0.76)	95.46	(+0.17)
Benchmark (CoNLL2009)		85.63	(+0.25)		81.00	(+0.31)	95.59	(+0.13)
Daisy(**TW**,C,P,HP,DR,GP)	85.64	85.92	(+0.29)	86.17	81.40	(+0.40)	95.66	(+0.07)
Daisy(TW,C,P,HP,DR,**LP5**)	85.77	86.04	(+0.13)	86.31	81.53	(+0.13)	95.77	(+0.11)

Table 3.4: Performance on WSJ Eval Dataset for Various System Configurations

System Description	SystemF1				RoleF1		SenseAcc	
	Min	Mean	(Δ)	Max	Mean	(Δ)	Mean	(Δ)
Daisy(W,C)	70.50	71.70		72.43	65.49		85.08	
Daisy(W,C,**P**)	72.45	73.13	(+1.43)	73.78	67.38	(+1.89)	85.66	(+0.59)
Benchmark (CoNLL2009)		73.31	(+0.18)		67.78	(+0.40)	85.23	(-0.43)
Daisy(W,C,P,**HP,DR**)	72.47	73.48	(+0.17)	74.43	67.87	(+0.09)	85.71	(+0.48)
Daisy(W,C,P,HP,DR,**GP**)	73.17	73.83	(+0.36)	74.23	68.21	(+0.34)	86.04	(+0.33)
Daisy(**TW**,C,P,HP,DR,GP)	74.85	75.80	(+1.97)	76.46	70.80	(+2.59)	86.72	(+0.68)
Daisy(TW,C,P,HP,DR,**LP5**)	75.19	75.90	(+0.09)	76.93	70.62	(-0.18)	87.40	(+0.69)

Table 3.5: Performance on Brown Dataset (OOD) for Various System Configurations

syntactic dependency parses. The use of a dependency-based path feature, in particular, provides a significant boost in performance over simpler feature sets.

Promising future directions suggested by these results include whether proxies for the dependency based features can be derived from a similar architecture without the direct need for a full dependency analysis, thus eliminating the pre-processing parser cost. Another future direction involves the predicate disambiguation system. Although this sense disambiguation task is part of the CoNLL 2009 SRL evaluation, it is more properly a word sense disambiguation problem. A more thorough investigation of sense disambiguation in the context of an SRL system is warranted.

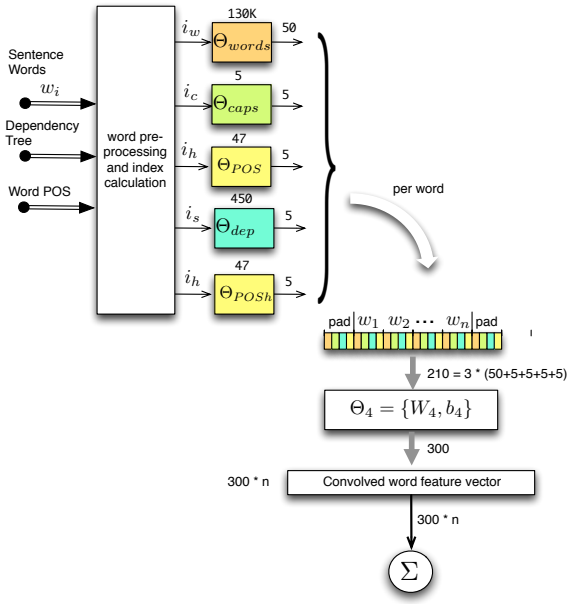


Figure 3.3: Word Preprocessing, Word Derived Features, and Word Derived Feature Convolution. (A) in figures 3.1 and 3.2.

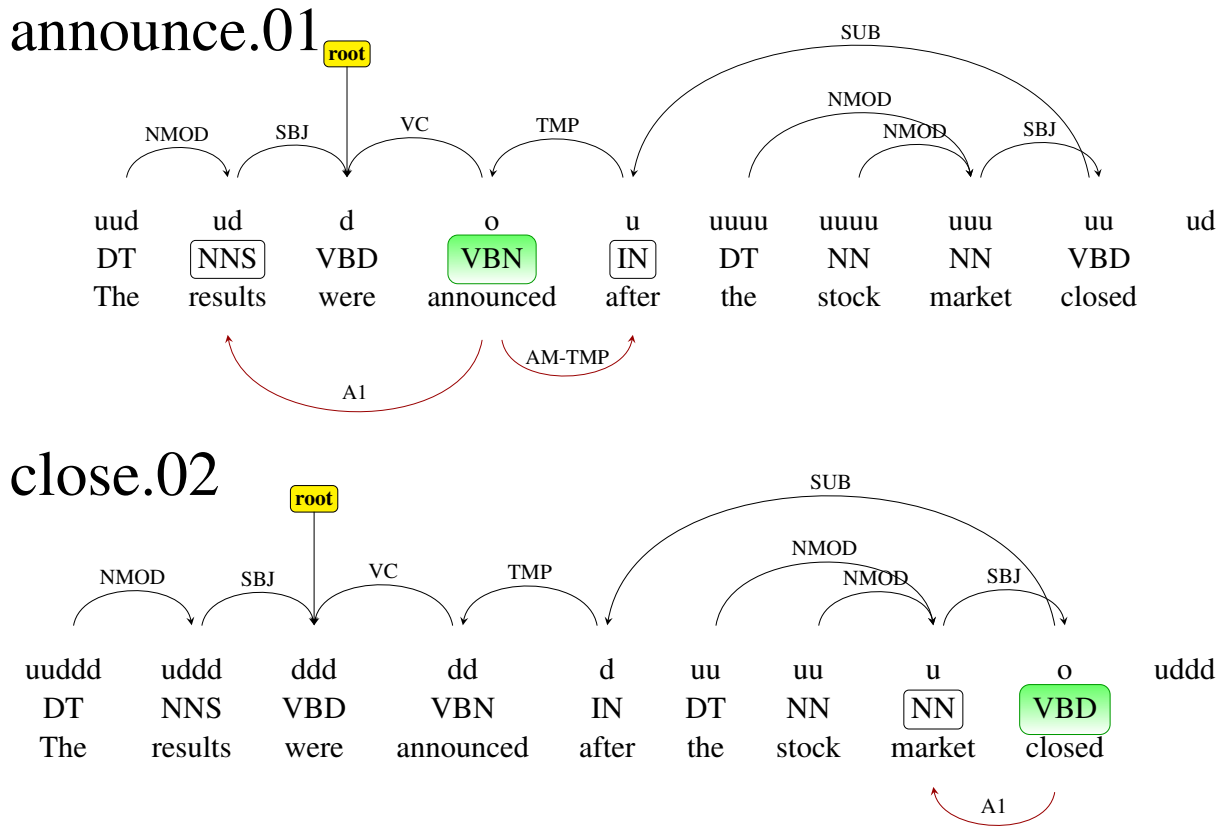


Figure 3.4: Dependency Parse and Generic Paths

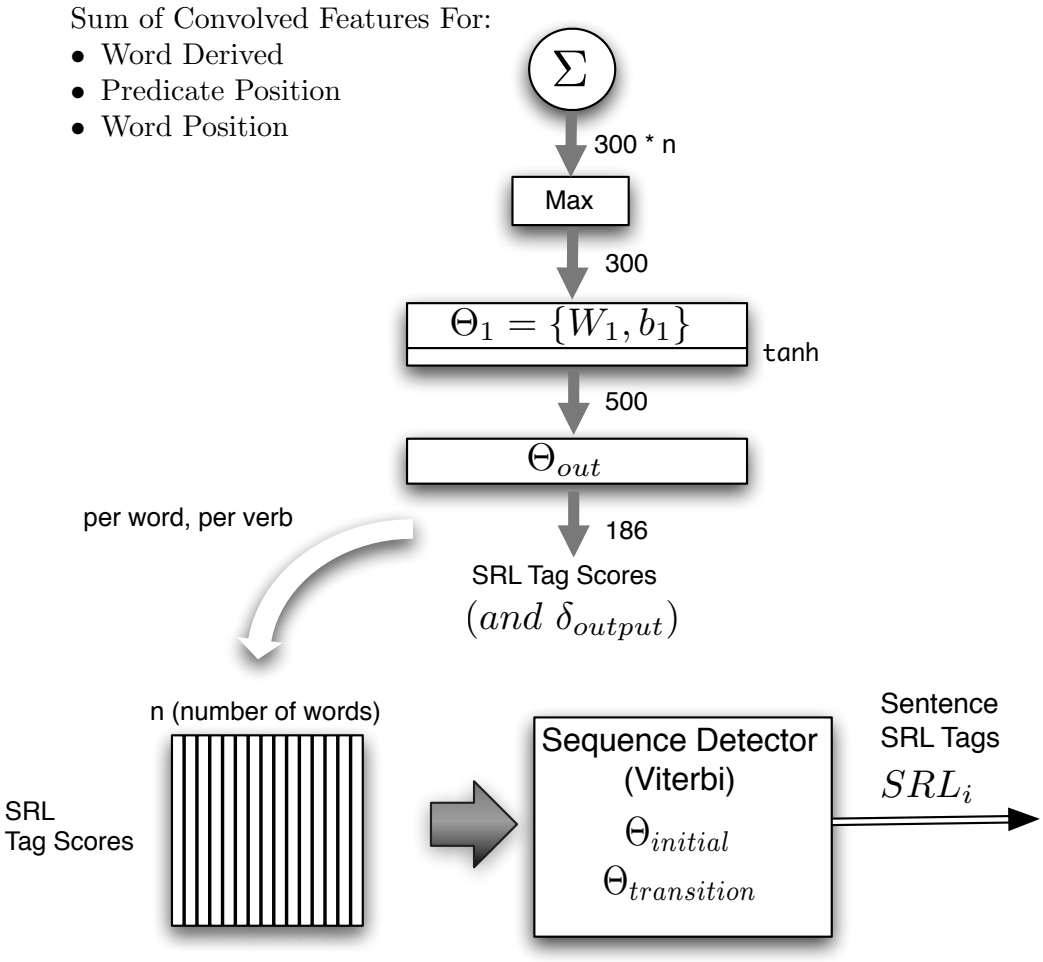


Figure 3.5: SRL Neural Network and Viterbi. (D) in figure 3.1.

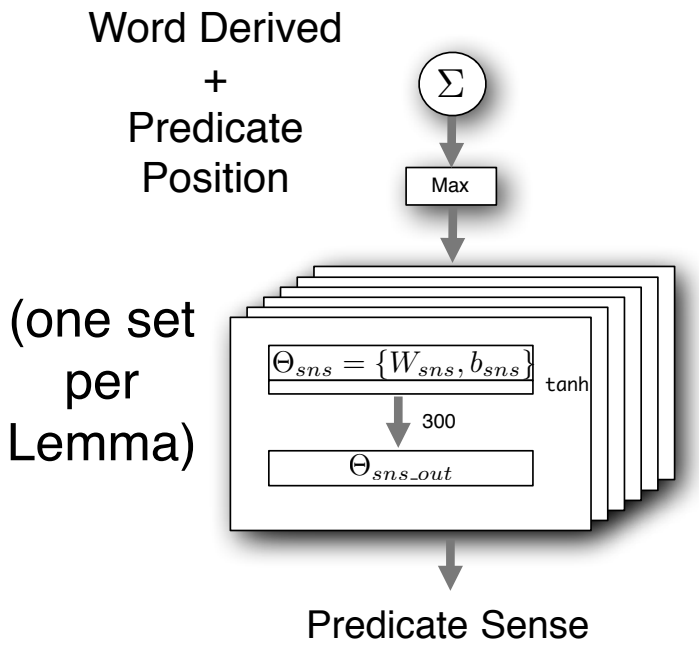


Figure 3.6: SRL Neural Network for Predicate Sense. (E) in figure3.2.

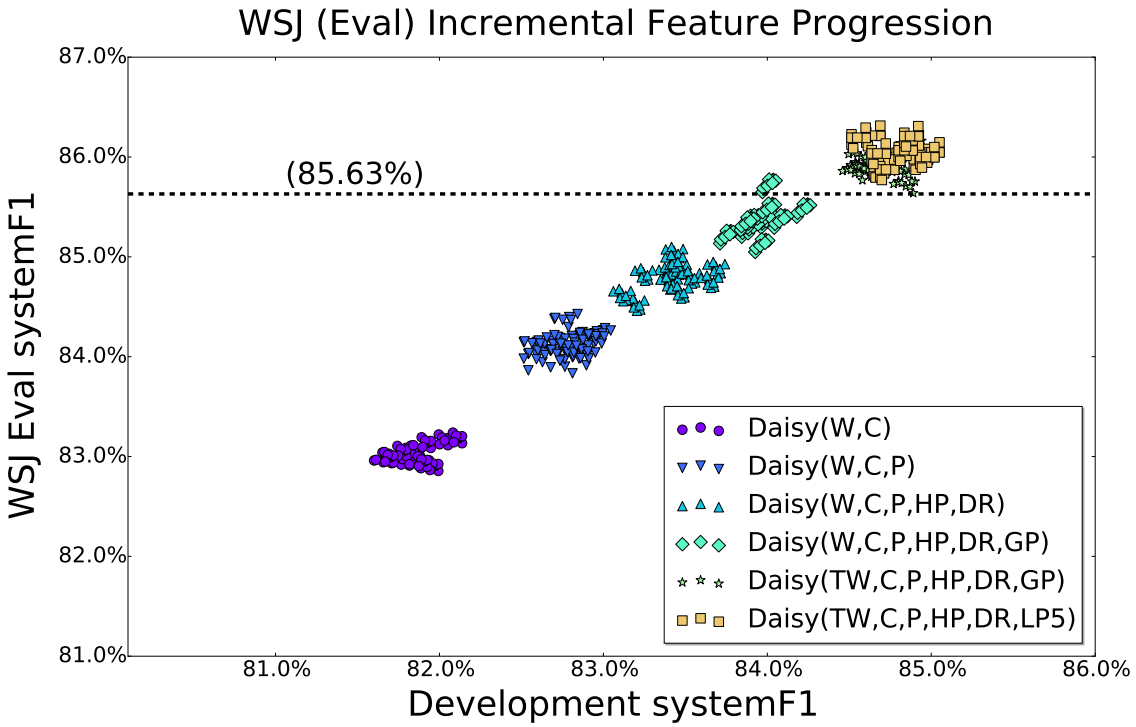


Figure 3.7: Scatter Plot of Dev F1 vs. Eval F1 for Various Feature Configurations (See also Table 3.4)

Chapter 4

Abstract Meaning Representation Parser

4.1 Introduction

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 pre-parse, or heavily engineered features, and uses five recurrent neural networks as the key architectural components for inferring AMR graphs.

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 pro-

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

vide 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., 2005a]. More recent innovations include wikification of named entities and normalization of temporal expressions [Verhagen et al., 2010, Strötgen and Gertz, 2010]. [Bos, 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 semantic 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

[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.

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.

4.2 Related Work

4.2.1 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, Wang et al., 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 [Vanderwende et al., 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.

[Peng et al., 2015] proposes a synchronous hyperedge replacement grammar solution, [Pust et al., 2015] uses syntax-based machine translation techniques to create tree structures similar to AMR, while [Artzi et al., 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 [Foland Jr and Martin, 2016], which, following [Collobert et al., 2011], relies on word embeddings and recurrent neural networks to generate AMR graphs.

4.2.2 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. [Zhou and Xu, 2015], successfully used a bidirectional LSTM network for semantic role labelling. We use the LSTM cell as described in [Graves et al., 2013a], configured in a B-LSTM shown in Figure 4.3, as the core network architecture in the system. Five B-LSTM Neural Networks comprise the parser.

4.3 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 4.1.

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

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

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 the form of a vector of word embeddings 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 4.2. 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.

4.4 Detailed Parser Architecture

4.4.1 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 4.1).

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 [Pourdamghani et al., 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 4.1. 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 4.4.

4.4.2 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.

4.4.2.1 Word Embeddings

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 em-

beddings are mapped from the words in the sentence, and are then trained using back propagation just like other parameters in the network.

4.4.2.2 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 4.1). 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.

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.

4.4.2.3 AMR Subgraph (SG) Network

The most frequent 25 subgraphs of 46 total which are identified by the SG network are shown in Table 4.1. Subgraphs are composed of either a single concept, as in NonPred (non-predicate), or two concepts, as in Named. The parent and child concepts and the relation connecting them within the subgraph are shown for each subgraph in the table.

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 4.2.

4.4.2.4 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 4.1 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 4.1.) 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.
- PredSG[5], 46x10: The SG embedding of the word and surrounding 2 words associated with the source predicate subgraph.

[6] relation probabilities change as hard decisions are made, see section 4.4.3

- regionMark: 21x5, indexed by the distance in words between the word and the word associated with the source predicate subgraph.

Table 4.3 shows an example feature set for one subgraph while evaluating a predicate subgraph.

4.4.2.5 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.

4.4.2.6 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.

4.4.2.7 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

[7] Degree, mod, or quant are examples of Narg relations in Figure 4.1.
[8] (TOP: plan-01) and (op1: france) are attribute examples shown in Figure 4.1.

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.

4.4.3 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:

(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.

4.4.4 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 4.5. 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.

4.5 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, 2015a].

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. ([May, 2016] describes the eval dataset as ”quite difficult to parse, particularly due to creative approaches to word representation in the web forum portion”).

4.6 Results

We report the statistics for smatch results of the ”test” and ”eval” datasets for 12 trained systems in Table 4.4. 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.

4.6.1 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.

4.7 Conclusions

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 infor-

mation, 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 pre-processing. 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.

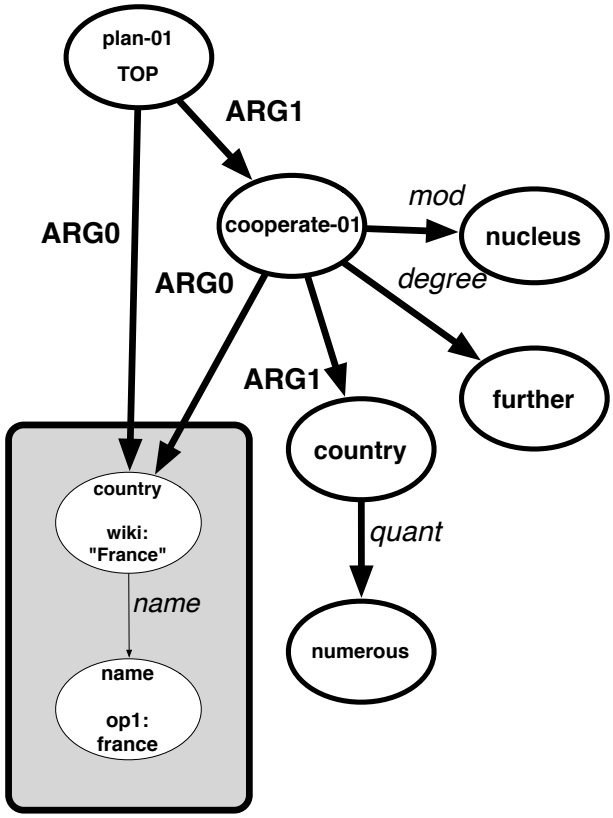


Figure 4.1: An AMR graphical depiction of the meaning of the sentence ***France plans further nuclear cooperation with numerous countries .*** Concepts are represented as ovals, and relations are the directed connections between them. Predicate concepts are labelled with their PropBank sense, and semantic 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.

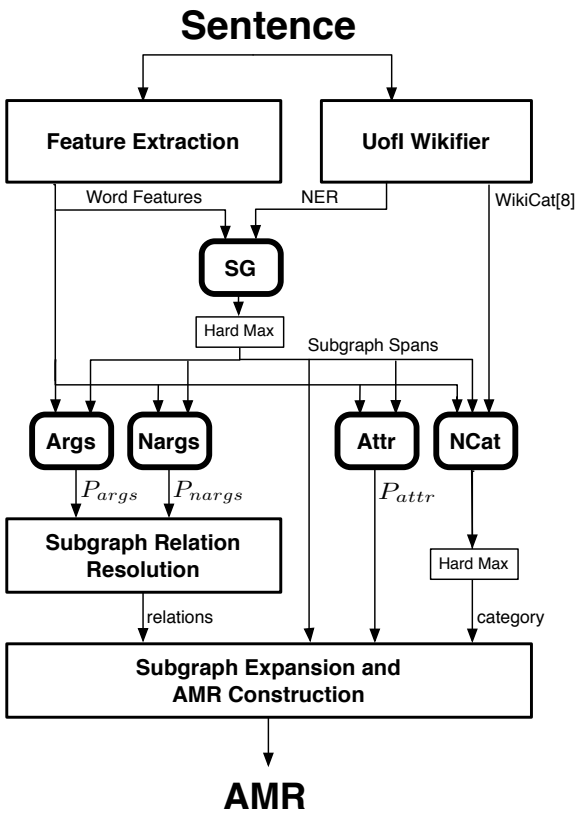


Figure 4.2: General Architecture for the AMR Parser. The parser creates an AMR based on the words in a sentence. The 5 B-LSTM networks infer structures of the AMR. For example, the SG network infers subgraphs, which are mostly single concept, like ”plan-01” or ”further”, but can also be like the more 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).

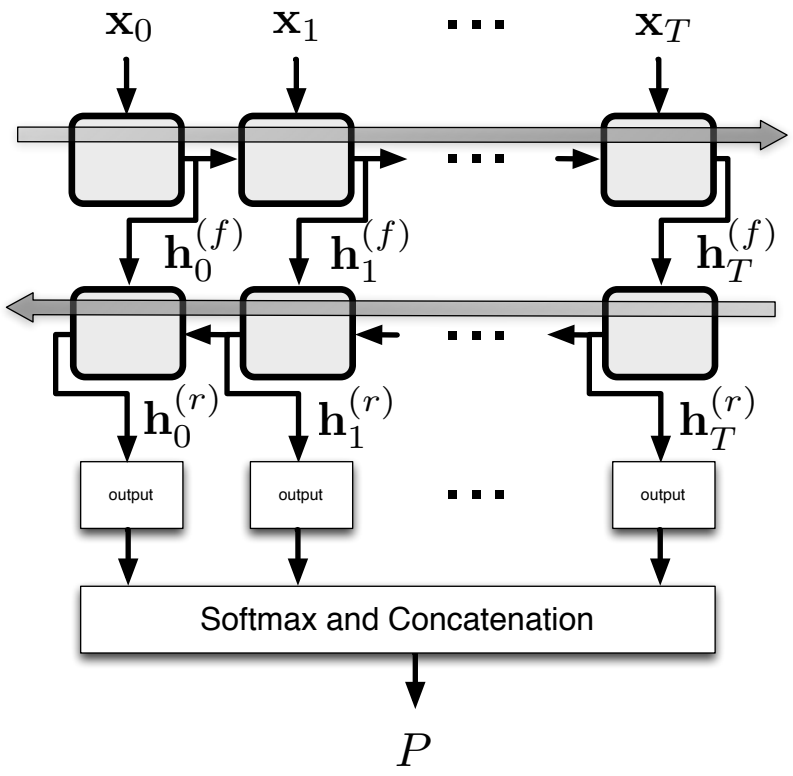


Figure 4.3: A general diagram of a B-LSTM network. Diagram shows the feature input vectors x_i, the forward layer (f) and the reverse layer (r). The network generates vectors of log likelihoods which are converted to probability vectors and then joined together to form an array of probabilities.

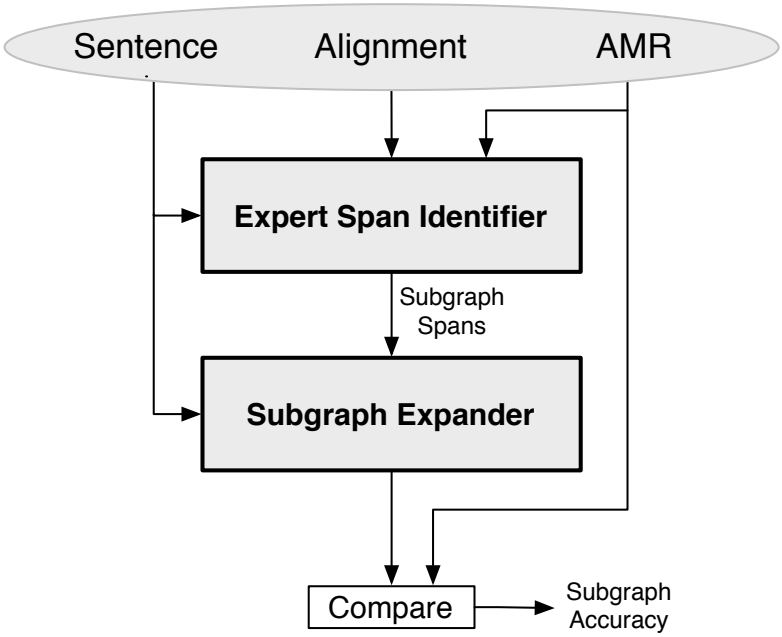


Figure 4.4: Expert System and Subgraph Expander Development. The alignment between the words in the sentence and elements of the AMR is provided by an automatic aligner. The expert system uses the sentence, reference AMR, and alignment to identify spans of words which are related to concepts within the AMR. These spans are also labelled with a subgraph type. A ”subgraph expander” uses the words and subgraph type to expand into AMR subgraphs.

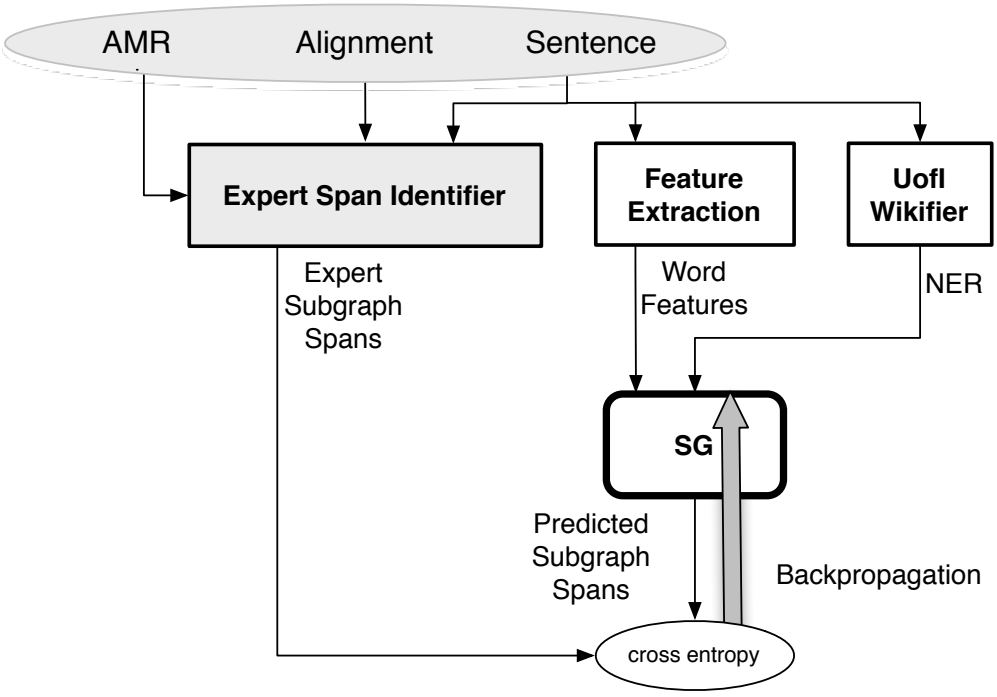


Figure 4.5: SG Network Training. The SG Network uses just the words in the sentence as input, and is trained to imitate the output of the Expert System. This output defines spans of words and their subgraph types, which are the nodes of the AMR graph. Later stages of the system use this information to infer other aspects of the AMR, like relations (edges).

Span Tag	Count	Accuracy	Parent	Child	Relation
NonPred	60112	0.984	lemma(N)		
Pred	45016	0.933	lemma(V)-sense		
Named	17295	0.951	category/wiki-link	name/words	name
DateEntity	2050	0.982	date-entity/norm'd date		
HaveOrg	1739	0.983	have-org-role	lemma	ARG2
Person0	1634	0.952	lemma(V)	person	ARG0
FrequentDouble	803	1.000	govern-01	government-organization	ARG0
TemporalQuantity	695	0.993	temporal-quantity/quant	(unit-type)	unit
Person1	599	0.922	lemma(V)	person	ARG1
Thing1	515	0.935	lemma	thing	ARG1
More	406	0.972	lookup(word)	more	degree
HaveRel	400	0.984	have-rel-role-91	word	ARG2
Include	317	0.885	include-91	word	ARG2
Most	276	0.979	lookup(word)	most	degree
Thing2	242	0.914	lemma	thing	ARG2
HaveConcession	240	0.995	have-concession-91	lookup(word)	ARG2
MonetaryQuantity	220	0.948	monetary-quantity/quant	(quant-type)	unit
Person2	159	0.895	lemma(V)	person	ARG2
BeLocatedAt	155	0.946	be-located-at-91	word	ARG2
MassQuantity	152	0.995	mass-quantity	(unit-type)	unit
HaveCondition	136	1.000	have-condition-91	lookup(word)	ARG2
Why	108	1.000	cause-01	amr-unknown	ARG0
Thing0	107	0.934	lemma	thing	ARG0
DistanceQuantity	83	0.981	distance-quantity	(unit-type)	unit
Ago	73	1.000	before	now	op1

Table 4.1: Top 25 of 46 subgraph categories identified by the SG Network. Accuracies are measured during system development, using the cascaded system shown in Figure 4.4. Parent and child nodes, and their connecting relation, are used by the expert to identify spans, and by the expander to create final AMR concepts and relations. NLTK lemmatizer specified for Nouns(n) of Verbs(v), lookup is the most common translation for the word found in training data.

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 4.2: 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 4.3: Args Network Features for the word *France* while evaluating outgoing args for the word *cooperation*, associated with predicate **cooperate-01**

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

Chapter 5

Distributed AMR Parser

The AMR parser described in Chapter 4 is constructed from multiple recurrent neural networks, cascaded together to produce log-likelihoods related to AMR concepts and relations. These resulting log-likelihoods are then used to choose and construct the most likely AMR corresponding to the input sentence.

This AMR Parser was modified so that probabilities are preserved until AMR construction time. Instead of deciding the subgraph type for each word early in the process, we instead maintain the distributed representation produced by the SG neural network and use it as the feature for the Args, Nargs, Attr, and Cat networks.

The modified AMR parser can be considered as having two separate stages: a) the generation of distributed concept and relation representations b) converting these distributed representation into discrete symbols and connections which make up the AMR. An advantage of this architectural view is that an intermediate result is produced, which we can investigate as the source for semantic processing besides AMR construction. We call this intermediate representation Distributed Abstract Meaning Representation, or DAMR.

The system shown in figure 4.2 was modified and split into two discrete systems so that it produces DAMR as a byproduct. The resulting systems, which can now be called a DAMR Parser and AMR Constructor, are shown in Figures 5.1 and 5.2.

The DAMR Parser is composed of many layers, each of which can be considered to be a distributed representation of features for that layer. This gives us a number of possible levels to

choose for defining the information to be used to construct DAMR. In the diagrams, we show the option of selecting the layer prior to the softmax output layer. For most of the experiments which follow, however, we chose to use the log-likelihoods produced by the softmax layer. The log-likelihoods can be interpreted as a probability distribution over tags, and thus should be more suitable for modification based on disambiguating evidence.

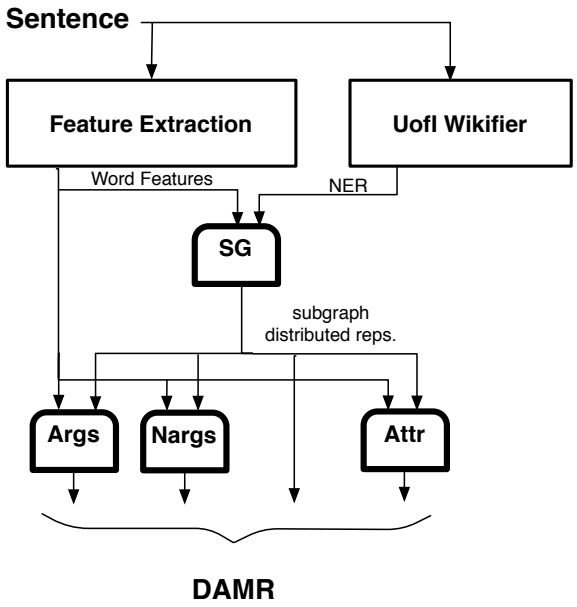


Figure 5.1: DAMR Parser

A sentence in isolation can usually be interpreted in many different ways. We can disambiguate the meaning of a sentence based on information not specifically specified by it, by using surrounding context, pragmatic evidence, or common sense, for example.

DAMR preserves some of the ambiguity associated with the semantic structure of represented meaning. It preserves probabilistic semantic representations which could be modified using other information to improve the interpretation of meaning.

The NN based AMR parser natively delivers probability distributions, and these distributions could be preserved until meaning must be resolved, for example, a symbolic representation is desired. Up until that time, new information could be used to alter or bias the probability distributions, taking into account all known information.

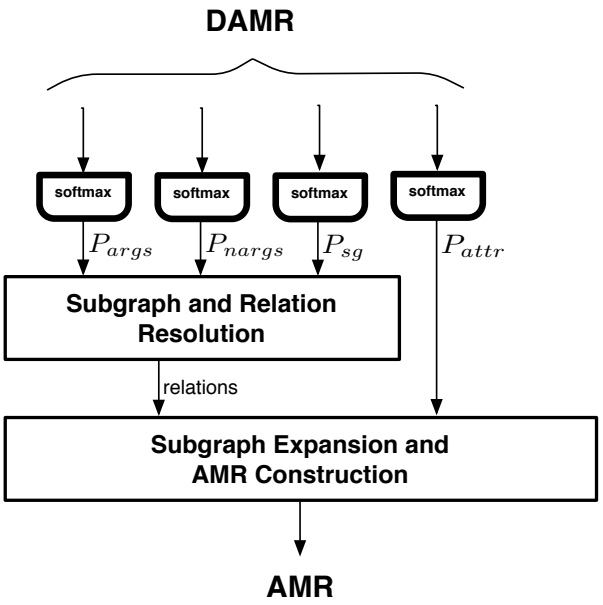


Figure 5.2: AMR Constructor

Even though DAMR representations are induced using supervised training, the training objective is not task specific, since it is based on inducing general semantic information as defined by the AMR semantic framework. DAMR should therefore be useful as a multi-task representation. The DAMR to Sentence Vector Bridge should be trainable with either task-specific or unsupervised (task-generic) objectives. For simplicity, we will use an inference task objective to test the DAMR representation.

5.1 Distributed Subgraph as a Training Feature

During training, the expected, or ”gold” subgraph type is derived from the human annotated AMR graphs provided in training data. An SG network is first trained using the gold subgraph target to feed an objective function for back propagation. The output of a trained SG network is then used to generate the distributed feature which will be used as input to the SG-dependent networks: Args, Nargs, Attr, and Cat.

The error rate from the trained SG network will impact the quality of the distributed SG feature. Introducing errors during training is a concern, but during forward (decode) mode, the input to the network will contain cascaded errors from the SG network, so it may be important to include them during training to simulate the expected forward environment. The tradeoffs of using the gold SG target to bias the distribution during training were investigated using the following vector feature schemes:

- **Forced Probability:** Train with pseudo probabilities generated from the gold SG target.
- **Corrected Probability:** Train with SG network probabilities, except when the most probable tag conflicts with the gold tag, in which case use a set of pseudo probabilities generated from the gold SG target.
- **Raw SG Probability:** Train with SG network probabilities with no influence of the gold SG target.

The results of this experiment showed that the Corrected and Raw SG probability features generated equivalent results, whereas Forced Probability feature performed significantly worse. Therefore, the Raw SG Probabilities were used to train SG dependent networks for DAMR creation.

5.2 Distributed AMR Description

Figure 5.4 shows a rough illustration of a DAMR, composed of the intermediate productions from BDLSTM neural networks as described.

Recall that the AMR parser divides spans of words in a sentence into concept subgraphs. A distributed representation of the probabilities over concept subgraph types for each word is generated by the SG network. A corresponding attribute representation for each word is generated by the Attr network, and a distribution over named categories is generated by the Cat network. These three vectors are concatenated together to form a single vector f for each word. Thus,

for a sentence which is described by s words, the distributed representation of these subgraphs is $f_0, f_1, ...f_{s-1}$.

The relations from predicate subgraph concepts, or Args, are obtained from the Args BDLSTM network, and form an sxs array called A, where each entry A_{ij} is the distributed representation of the relation between the predicate subgraph i to subgraph j (a directed relation). Note that a $noconnection$ relation is included in the class set, and that in the case where $i = j$, $noconnection$ is forced, since self loops are not allowed in the AMR description. The N matrix for Nargs is constructed in the same way as A for Args.

5.3 Results

The DAMR Parser was tested and compared with the AMR parser. This serves as an independent check of the quality of the DAMR representation. Each network was trained individually, and the F1 score for the dev partition was used to determine the best parameters. Statistics were not gathered for multiple parsers as for the AMR Parser, but the single DAMR parser derived from the best performing networks as judged by dev F1 was tested using the eval corpus as before. F1 results are shown in table 5.1. Parser results improved to 0.659 when the log probabilities for SG were used as input to Args, Nargs, Attr, and Cat networks, which was not completely expected. The reason for the improvement is probably due to the systems ability to exploit the extra information about ambiguous subgraph types that is maintained in the probabilistic representation. Unfortunately, the individual test results from the AMR Parser development were tested using the gold SG as input, instead of using the output of the SG network, as was done for the DAMR NN testing. Therefore, a direct comparison cannot be made without further experimentation.

The F1 Scores for each network are shown in figure 5.2.

The F1 score for the test partition was calculated for each network, and the confusion matrices are shown in figures 5.5, 5.6, 5.7, 5.8 and 5.9.

System	Description	Test F1	Eval (OOD) F1
DAMR Parser			**0.659**
AMR 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]		0.672	0.6196
Brandeis/cemantix.org/RPI [Wang et al., 2016]		0.667	0.6195
CU-NLP [Foland Jr and Martin, 2016]		0.661	0.6060
ICL-HD [Brandt et al., 2016]		0.620	0.6005
UCL+Sheffield [Goodman et al., 2016]		0.637	0.5983

Table 5.1: Smatch F1 results for the DAMR parser and top 5 parsers from semeval 2016 task 8.

Network	Precision	Recall	F1
Subgraph (SG)	0.828	0.863	0.845
Predicate Relation (Args)	0.662	0.586	0.622
Non-predicate Relations (Nargs)	0.613	0.529	0.568
Attributes (Attr)	0.811	0.814	0.812
Named Categories (Cat)	0.770	0.818	0.793

Table 5.2: Precision, Recall, and F1 results for DEFT test split for each individual neural network in the DAMR parser system

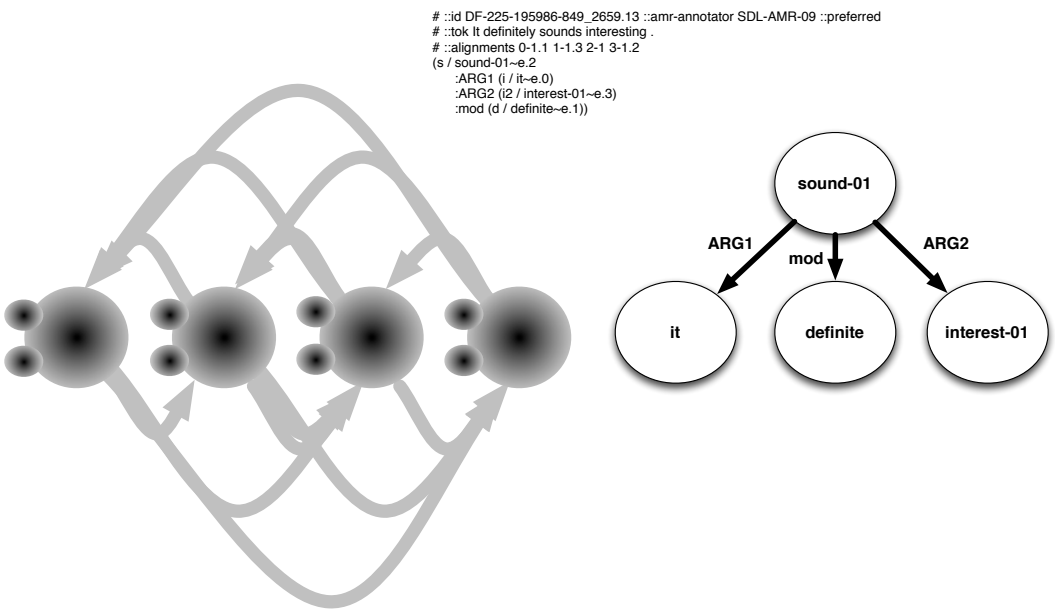


Figure 5.3: DAMR to AMR.

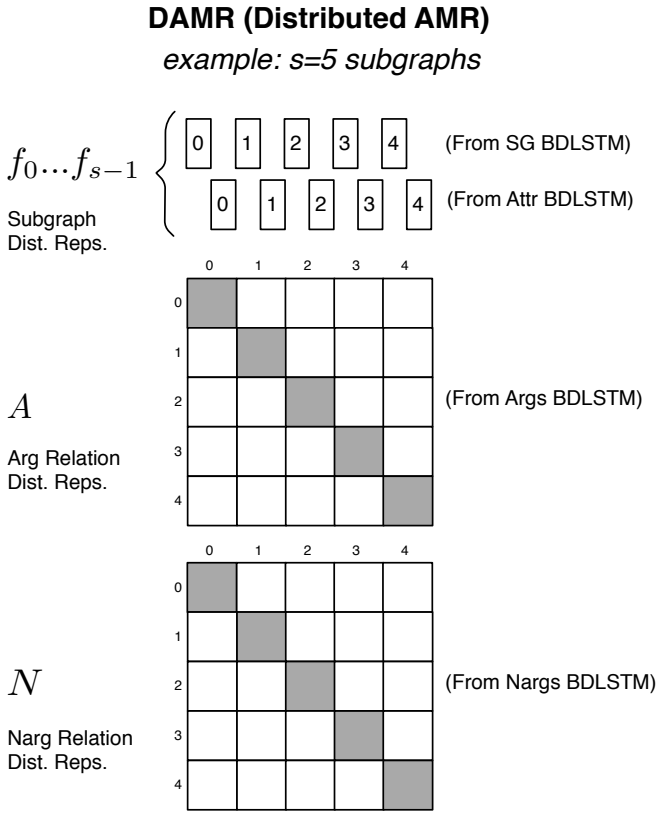


Figure 5.4: Rough example of a DAMR, for a sentence with five spans.

$f_0 \ldots f_{s-1}$	Sourced by AMR Parser, see Sections 4.4.2.3 and 4.4.2.6
A	Sourced by AMR Parser, see Section 4.4.2.4
N	Sourced by AMR Parser, see Section 4.4.2.5

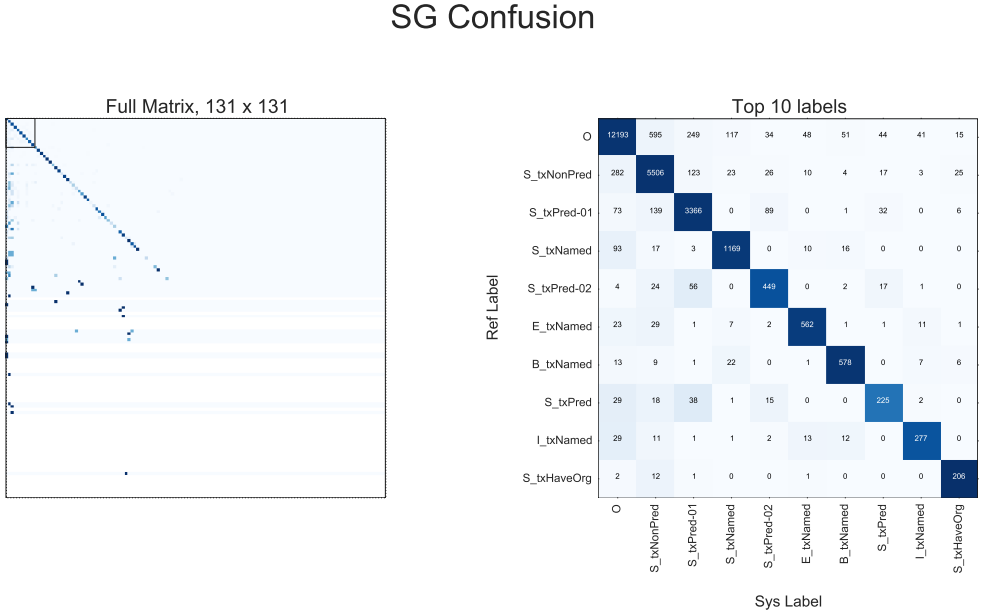


Figure 5.5: Confusion matrix for the 131 subgraph tags produced by the SG Network on the DEFT test dataset. The number of training samples for infrequent tags results in a higher error rate, as can be seen by the loss of a coherent diagonal map about a third of the way down. The detailed chart shows the most frequent 10 labels in the training data. The system occasionally confuses the three types of predicates: pred-01, pred-02, and pred (any sense), which is a difficult distinction which might be done more accurately by a second network. The BIOES tags are mostly useful just for named tags.

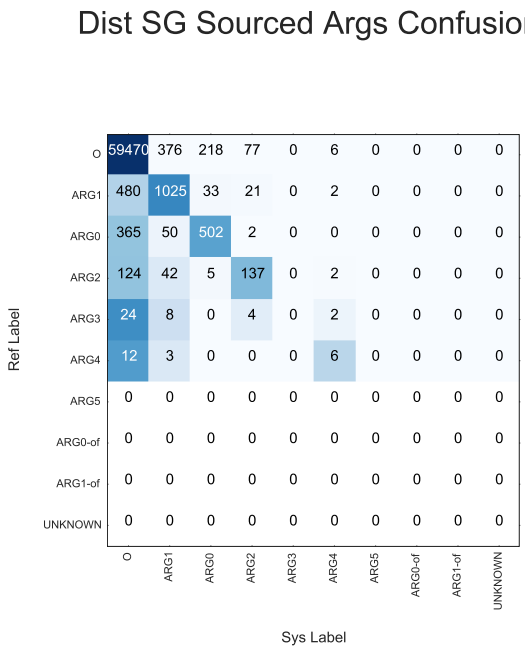


Figure 5.6: Confusion matrix for the 6 arg tags produced by the distSG-sourced Args Network on the DEFT test dataset. The bottom four tags are not present in the test data. The recall of 0.586 is evident in the high numbers in the left hand squares, indicating that no Arg was identified by the system. Some of the errors in precision (0.662) can be seen in the high numbers in the top row, indicating an argument was indicated by the system when none actually existed. Some of these errors are due to cascaded errors from the probability distribution produced by the SG Network (distSG).

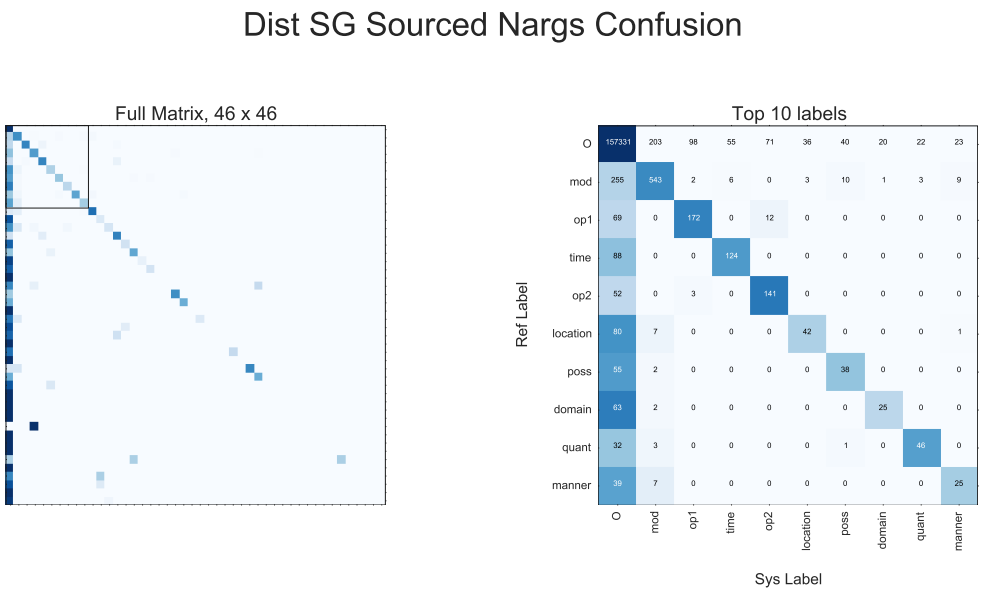


Figure 5.7: Confusion matrix for the 46 non-argument tags produced by the distSG-sourced Nargs Network on the DEFT test dataset. The low recall of 0.529 is shown by the high numbers in the left hand column. An analysis and repartitioning of the Nargs into separate networks might improve performance.

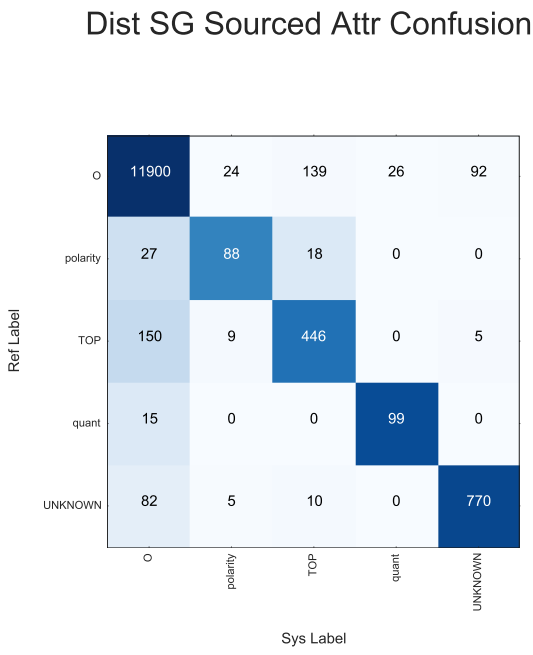


Figure 5.8: Confusion matrix for the 5 attribute tags produced by the distSG-sourced Attr Network on the DEFT test dataset. Note that the network detects three important labels: polarity, which reverses the meaning of a concept, TOP, which defines the root of the AMR tree, and quant, which identifies quantity relations. The UNKNOWN tag is not used for AMR construction, but catches many of the opx attributes, which will eventually be sourced by the order of words in the sentence for named entity spans.

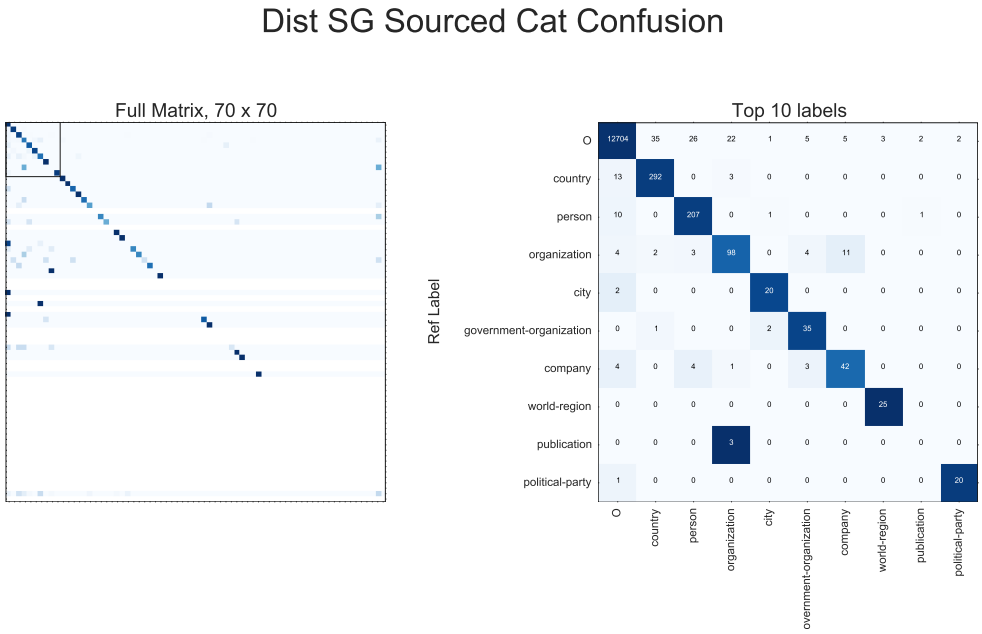


Figure 5.9: Confusion matrix for the 70 category tags produced by the distSG-sourced Cat Network on the DEFT test dataset. The Cat network is used to predict the category for named entities, for example, **country** for *France*. Recall is more important than precision, since the span must be identified as a named identity from the SG probabilities before the category will be used. Note the difficulty the network has in discerning the difference between publication and organization, or company and organization. Some of these distinctions are probably error prone for humans, too.

Chapter 6

Sentence Vectors from Semantic Graphs

In this chapter, we investigate the use of DAMR as the feature source for a model, called the DAMR Bridge, which creates fixed length sentence vectors. Two Natural Language Inference corpora were used as benchmark tasks to assess the quality of the sentence vectors produced.

The motivation for converting DAMR to sentence vectors is twofold: First, it allows us to assess the quality of AMR independently from the smatch graph comparison algorithm. The smatch algorithm is used to compare a golden AMR to the output from a parser. It does this by breaking down the two graphs into sets of triples for concepts, attributes, and relations, and comparing the two sets in order to produce precision, recall, and f1 scores. The correspondence between one set of concepts and another is obtained by using a hill-climbing algorithm to minimize error, and that algorithm is stochastic and dependent on the number of steps applied during computation. The idea is that if one concept is wrong is in a parsed graph, all other concepts should be aligned to each other, so that only one triple is in error. This matching can be prone to wide variation when the graphs are large and more than one concept differs.

Smatch also does not have a notion of relative semantic relevance, it penalizes the difference between slightly different word senses as much as a polarity error which negates the entire meaning of a sentence.

Second, we can investigate the effectiveness of using intermediate results from a pre-trained symbolic parser as the source for distributed sentence representations. Sentence vector sourced tasks are becoming more common. A graph-based intermediate representation like DAMR allows

graphical algorithms to be applied, like the abstractive summarization described in Chapter 2. The probabilistic form of DAMR also gives more options for influence with contextual or pragmatic evidence. These graphical advantages might give us more discourse level analysis options, while maintaining compatibility with some standard downstream vector-sourced application.

For these two reasons, it is interesting to explore the use of DAMR and DAMR bridge to assess meaning using a natural language inference corpus.

6.1 DAMR to Sentence Vector Bridge

DAMR is nonuniformly sized, expanding to express longer sentences and more complex semantic content. The Args matrix is a two dimensional sparse matrix whose dimensions are $p \times n$, where p is the number of identified predicates, and n is the number of words in the sentence. The size of the Nargs matrix is $n \times n$. A model which we call DAMR Bridge links the 2-D, nonuniformly sized DAMR to a fixed size sentence vector representation.

The DAMR Bridge is a compositional model that uses distributional triplets from DAMR to create a fixed size distributed vector representation. It is sourced by the DAMR Parser, and contains a Feature and Network section, which create a sentence vector. The Bridge Network is trained and tested based on data from a Natural Language Inference corpus. The entire system from sentences to inference tag probability is shown in figure 6.1. Note that the DAMR parser Ⓐ is pretrained on the AMR corpus, while the rest of the system is trained on an NLI corpus.

The model is trained with a cross-entropy loss objective, using minibatch Stochastic Gradient Descent. RMS Propagation [Tieleman and Hinton, 2012] was used to minimize sensitivity to learning rates, which were kept at 0.01 . The batch size is 512. Dropout of 0.2 is applied to the LSTM Recurrent and non-recurrent weights, and dropout of 0.3 is applied to all weights in the classifier. The 300D Glove 840B pretrained mixed-case vectors [Pennington et al., 2014] were used and were not altered during training. A dimension with value of 1.0 was added to represent Out-Of-Vocabulary words.

Model parameters are shared between hypothesis and premise sentences, to create a single

generic sentence encoding model that can is used for either sentence during testing.

All sentences are pre-padded up to a maximum length, which is necessary to form a consistent batch of multiple sentences to the GPU for fast processing. This maximum length will directly impact run time, so it was chosen carefully. A value of 42 was chosen for the SNLI corpus, and all sentences longer than that were truncated, which impacted less than 0.1%. The Multi-NLI corpus contains transcriptions of spoken dialog which can exceed 400 words in length. We chose to to use a padded sentence length of 100 for all Multi-NLI sentences, and to truncate the sentences longer than 100 (roughly 0.2%). Note that this was done for both the AMR neural networks and the DAMR bridge.

6.1.1 DAMR Bridge Features

6.1.1.1 Relation Compression

Rather than process the entire matrices of all possible relations described by Args and Nargs probabilities, we use an algorithm to select the most likely relation connections for each word and process using a ranked list.

Let n = number of words in the sentence. Define M as a matrix with dimension nxn, where

$$M_{sd} = max(A_{sdx}, N_{sdy}), \forall x \in Args \text{ and } \forall y \in Nargs$$

M describes the maximum probability of any inferred $Args$ or $Nargs$ connection between source word index s and destination word index d.

For each row in M, which represents the word with index s as the source of a relation, we sort the column indices d, representing destination word indices, based on the value of M_{sd}. This sorted list, D, is a list of relation destination indices d emanating from word index s, ranked by maximum probability of connection M_{sd}, for each s. We then have a list of destinations d for each source word s, ranked by maximum connection probability.

$$d = D(s, r)$$

The compression process is depicted in Figure 6.2.

Representing the word index of the source as s and destination d, the features representing distributed triples from the DAMR representation are shown in table 6.1

	Symbol	Description
Source Word	W_s	source word embedding
	SG_s	source word distribution over SG labels
	$Attr_s$	source word distribution over Attr labels
	Cat_s	source word distribution over Cat labels
Destination Word	W_d	destination word embedding
	SG_d	destination word distribution over SG labels
	$Attr_d$	destination word distribution over Attr labels
	Cat_d	destination word distribution over Cat labels
Relation	$Args_{sd}$	relation distribution over Arg labels
	$Nargs_{sd}$	relation distribution over Narg labels

Table 6.1: DAMR Bridge Features

6.1.2 DAMR Bridge Network

An LSTM row is used to reduce the sequence of features associated with a ranked row r into a fixed-size vector, which represents the words and their relations to other words in the sentence for a given relation probability rank from D. We refer to each LSTM row production as a ranked relation vector, K_r. The ranked relation vectors are then batch normalized and added together to create a sentence vector $\sum_{r=0}^{rows-1} K_r$ where rows is the number of LSTM rows we choose to add to the model for representing ranked probabilities from D.

The features input to the LSTM row r for source word s are defined as:

$$f_s^r = \{W_s, SG_s, Attr_s, Cat_s, W_d, SG_d, Attr_d, Cat_d, Args_{sd}, Nargs_{sd}\} \text{where } d = D(s, r)$$

Figure 6.4 shows the details of an LSTM row.

While the model is designed to support experimentation with up to five rows, we found that more than two did not improve performance and took a very long time to converge, so we limited the experiments for this version of the DAMR Bridge to two rows, representing the first and second most likely relations sourced by each word.

6.1.3 NLI Classifier

The NLI Classifier is used during sentence vector model training and for evaluating sentence vectors from either DAMR Bridge or simple Continuous Bag of Words models. The architecture is based on the best performing systems for NLI classification described in [Bowman et al., 2015, Marelli et al., 2014, Merity, 2016, Mou et al., 2016].

6.1.4 CBOW Projected Model

The Continuous Bag of Words (CBOW) Projected model is shown in figure 6.7. This model is based on composing word representations into a sentence vector by summing the representations together, completely ignoring word order. A projection layer projects the distributed word representations (in our case, Glove 640B, 300d) in to the model space, speeding up training and improving overall results. Batch Normalization is used to normalize the sentence vectors to reduce covariate shift [Ioffe and Szegedy, 2015b], which also speeds up training.

6.2 Testing Semantic Representations

We will use NLI to test the quality of DAMR-sourced sentence vector representations using standard published corpora, and compare with published results from other models on these corpora.

NLI standardized corpora have been steadily improving over the years. Up until 2015 or so, the primary sources of annotated NLI corpora have been the Recognizing Textual Entailment (RTE) challenge tasks. Another commonly used corpus was provided for the SemEval 2014 task

called Sentences Involving Compositional Knowledge (SICK). These are mostly hand-labeled data sets which were used for a number of innovative advances in the field, but they are limited by three important factors.

- They are limited to less than 5000 training samples in size, which is very small for distributed techniques such as neural networks, which require very large amounts of training data.
- A lack of consistency for handling indeterminacies of event and entity coreference.
- The quality of the data sets is lower due to automatic construction of many of the examples [Marelli et al., 2014].

6.3 SNLI

The Stanford Natural Language Inference Corpus, (SNLI Corpus)[Bowman et al., 2015] was created to overcome some of the problems with previous NLI Corpora. Over half a million sentences are generated and verified by humans, using Amazon mechanical turk.

The premise sentences are descriptions of photos from Flickr. These descriptions were provided by an independent crowd-sourced effort, to eliminate photographer specific bias. These descriptions are relatively simple and literal. The hypotheses are generated by mechanical turkers based on just the premise description text, and they are provided with instructions for how to generate entailing, contradicting, and neutral sentences. The instructions are specifically tailored to help prevent indeterminacies.

A validation step was performed by another group of mechanical turkers on a randomly segmented test and development data split. Five independent assessments for each pair were made, and the majority vote determined a gold label for the pair. In 92% of the validated pairs, the gold label matched the author's label, and only 2% of the pairs did not have a majority voted gold label, which means the process generated good quality data suitable for the task. The gold labels were used for validation and test, with the 2% without a gold label discarded.

A standard split was published, with 550,152 training pairs, 10,000 development pairs, and 10,000 test pairs. Table 6.2 shows some randomly selected samples from test dataset. The Premise sentences are generally simple and well formed, since they are all meant to describe a visual scene. The hypotheses are generally much shorter than the premises, and are also relatively simple.

Performance from a variety of models have been published, using the standard convention that only the training set is used for training, dev for monitoring progress, and test for final evaluation. The models can be considered to be of two types:

- Models which construct sentence representations independently for premise and hypothesis, and then classify based on the sentence representations.
- Models which consider the sentences together in some way, for example using attention between sentences, or which otherwise do not strictly generate sentence representations independently.

We will compare DAMR experiments with the published models which create premise and hypothesis representations independently. The SNLI website [1] shows many of the up to date results. By evaluating DAMR and the DAMR Bridge using this corpus, and comparing with the performance of using other representations, we should be able to gain a good perspective of the quality of the semantic content expressed in DAMR.

The fully populated DAMR Experiment model features are shown in 6.3, and these can be gated on or off independently for ablation studies. The system is described with a label which contains a 1 for each enabled feature, and a 0 if it is not used. Table 6.3 shows an example of a system where features are limited to: word embeddings for source and destination, subgraph types for source and destination, and Args for the relation between source and destination. The label for this system is 1-1-100-100-10 as shown in the table.

Table 6.4 shows DAMR experiments accuracy in context with accuracy of other relevant published sentence encoding models for the SNLI Corpus. The circled numbers will be used to

[1] http://nlp.stanford.edu/projects/snli/

reference the models in the following discussion.

① is a baseline classifier presented by [Bowman et al., 2015] that emphasizes handcrafted features.

② [Bowman et al., 2016] encodes the premise and hypothesis with two different LSTMs.

③ [Vendrov et al., 2015] use unsupervised skip-thoughts pre-training with GRU encoders. Skip-thoughts uses sentence to sentence context for training, similar to word2vec for words.

④ [Mou et al., 2015] use a tree-based CNN to capture sentence-level semantics.

⑤ [Merity, 2016] uses a well-tuned CBOW model with projection, surpassing previous published CBOW model performance by about 2%. This provided us with an example keras code base and a higher baseline target for CBOW.

⑥ **1-0-000-000-00 1 Row** is really just a single LSTM row with projected word representation input. This system performs about the same as CBOW. Since it actually adds word order to the model over CBOW, this is a strange result, but not unique to the DAMR experiments, for example, [Williams et al., 2017] report lower numbers with LSTM models than with CBOW on the SNLI corpus also. Perhaps this is due to the simplicity of SNLI sentences, and word order is not as important for good performance.

⑦ **1-1-111-111-11 2 Rows** is an attempt to extend the model from ⑨ into a higher number of rows, taking into account the second most probable relations in addition to the first. Instead of better accuracy, the result was worse, which is probably an indication that concentrated tuning is needed for increased DAMR rows.

⑧ **CBOW Projected** is an experiment which replicates the performance of ⑤ [Merity, 2016] using projected word embeddings and tuning the system for best performance. Note that the classifier width is 300 vs. 600 for ⑤ , but the system performs slightly better. The parameters found for this configuration were used for all others to try to normalize experiments somewhat.

⑨ **1-1-111-111-11 1 Row** performs about 0.5% better than the ⑧ CBOW Projected model. This represents using all the DAMR features for the most probable relation and destination from each word. See Figure 6.9 for details.

⑩ [Bowman et al., 2016] introduces a stack-augmented parser-interpreter neural network (SPINN) which combines CFG parsing and sequence interpretation within a single hybrid model.

⑪ [Munkhdalai and Yu, 2016b] use an S-LSTM structure which applies to tree structures, but do not use a pre-parser to define the tree, instead dividing the sentence words into a binary tree.

⑫ **1-0-111-111-11 1 Row + CBOW Projected** concatenates the DAMR bridge vector with the Projected CBOW model vector, and was the best performing DAMR SNLI configuration. The concatenated model performs about 1% better than CBOW alone, and about 0.5% better than DAMR alone. Earlier experiments that leaving out the destination word representation did not have an effect on performance, so this experiment was conducted without it. The model mainly performs better for contradiction than with CBOW alone, but also improves for the entailment prediction. See Figure 6.8 for details.

⑬ [Liu et al., 2016] use BiLSTM to generate sentence representations, and replace average pooling with inner-attention.

⑭ [Munkhdalai and Yu, 2016a] use a memory augmented neural network, which they call neural semantic encoder (NSE), they encode sentence vectors using read, compose and write operations.

In summary, DAMR improves the accuracy for the SNLI corpus by about 0.5%. An ablation study did not show a steady progressive improvement as more features were added. The best performance resulted from a DAMR bridge generated sentence vector concatenated to a CBOW model generated vector, about 0.5% better than with DAMR alone.

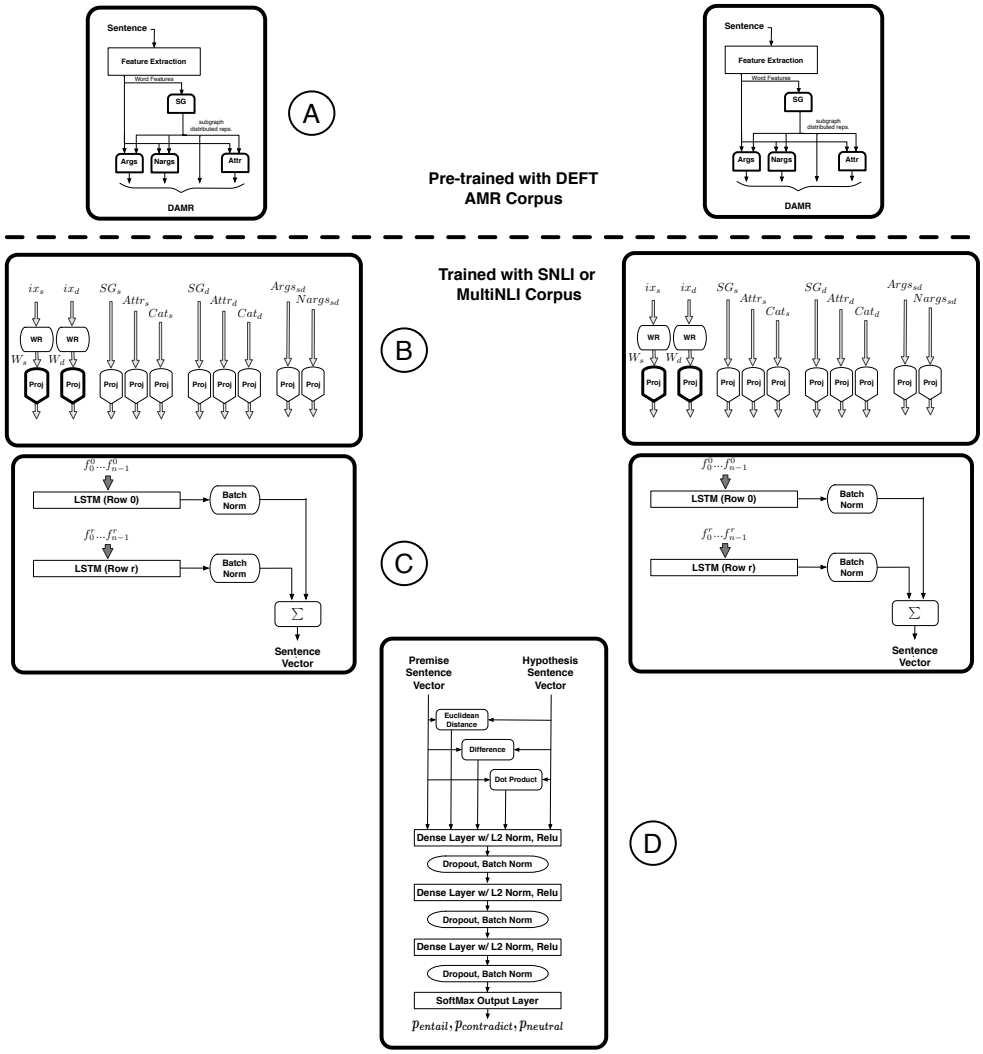


Figure 6.1: DAMR Sourced NLI System overview

Ⓐ	DAMR Parser	Chapter 5	Figure 5.1
Ⓑ	DAMR Bridge Features	Section 6.1.1	Figure 6.3
Ⓒ	DAMR Bridge	Section 6.1.2	Figures 6.4 and 6.5
Ⓓ	NLI Classifier	Section 6.1.4	Figure 6.6

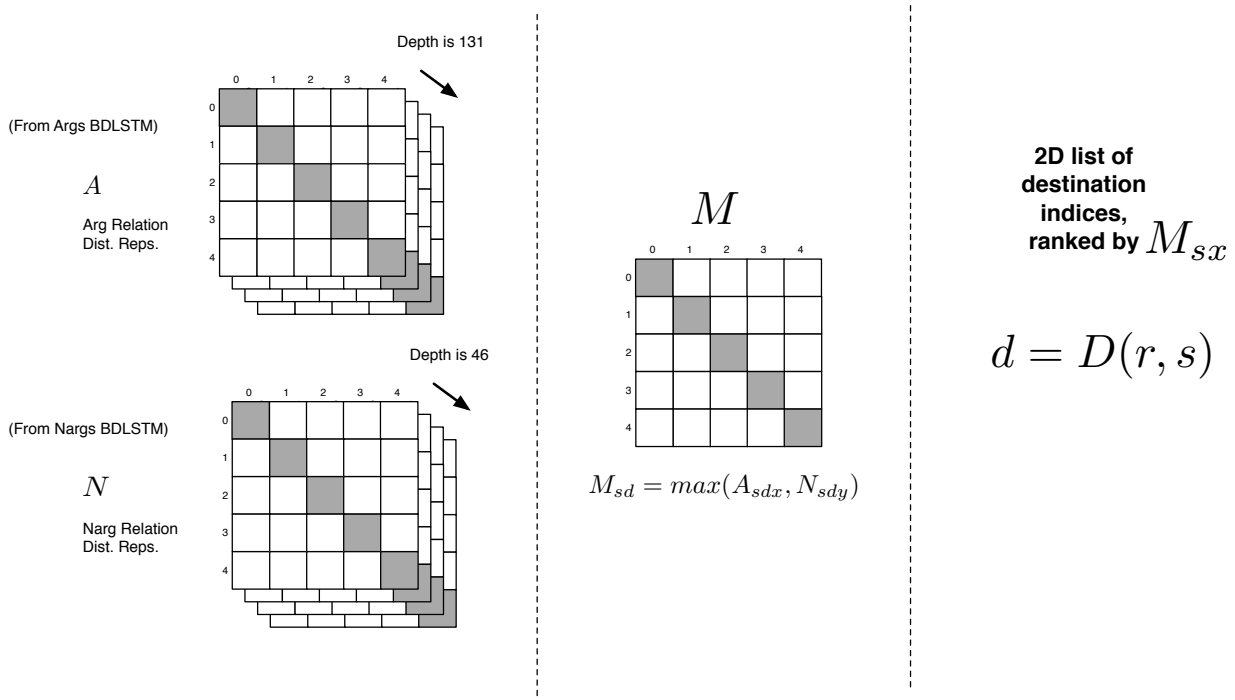


Figure 6.2: DAMR Compression based on relation probabilities.

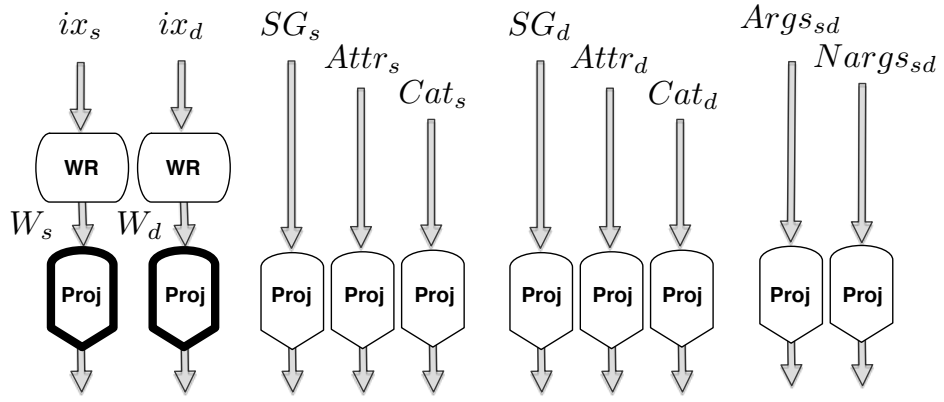


Figure 6.3: DAMR Feature Input Stage, producing a feature vector f_s^r. f_s^r represents a relation from source s to destination d. ($\{W_s, SG_s, Attr_s, Cat_s, W_d, SG_d, Attr_d, Cat_d, Args_{sd}, Nargs_{sd}\}$ where $d = D(s, r)$).

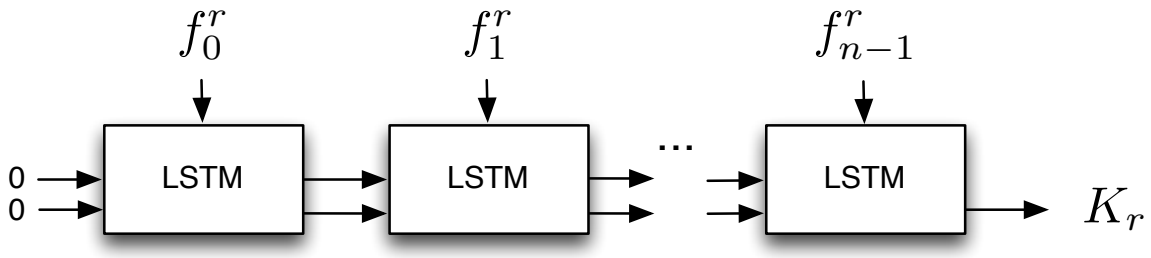


Figure 6.4: Feature input to an LSTM row with rank r. f_s^r represents a relation from source s to destination d. ($\{W_s, SG_s, Attr_s, Cat_s, W_d, SG_d, Attr_d, Cat_d, Args_{sd}, Nargs_{sd}\}$ where $d = D(s, r)$).

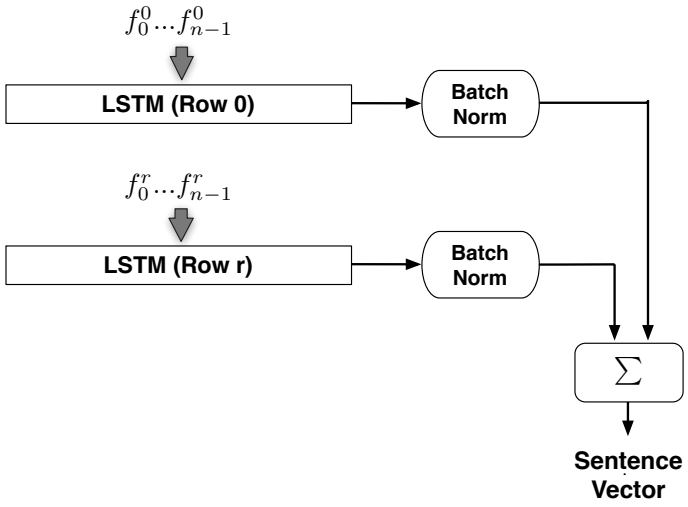


Figure 6.5: DAMR to SV Bridge.

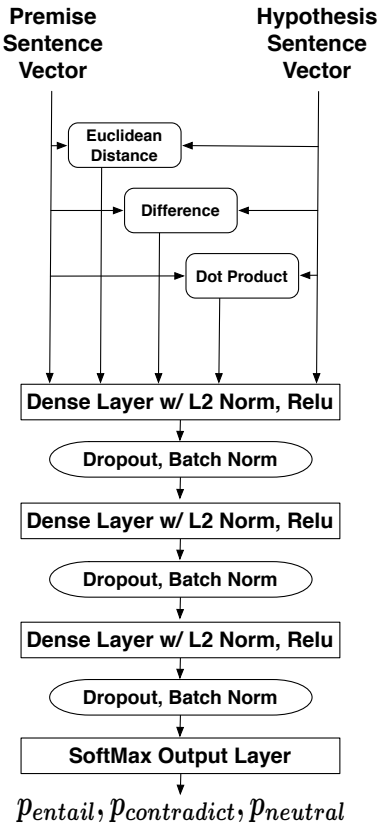


Figure 6.6: NLI Classifier.

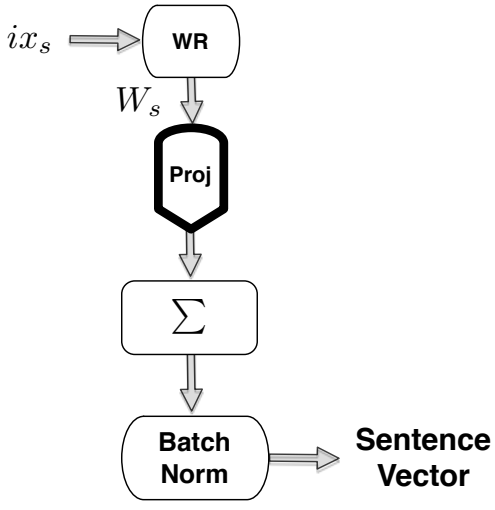


Figure 6.7: CBOW Projected Model.

Premise	**Hypothesis**	**Label**
Two children are sitting in the same wooden lawn chair while a green rake is leaning against the same chair.	The children are inside playing video games.	contradiction
	The children are getting ready for anap.	neutral
	Two children are on the lawn.	entailment
A man with a gray beard and an apron smiles.	A man eating a cow.	contradiction
	A grandfather smiling while baking	neutral
	A man smiling.	entailment
This young man is wearing blue jeans and a cowboy hat, and is walking away from a calf that is currently roped within an arena.	Nobody has jeans	contradiction
	A tall person in jeans	neutral
	A person in jeans	entailment
A black and white dog playing with a broken volleyball	Two dogs are walking down the road.	contradiction
	The dogs are playing with the ball that the kids just broke.	neutral
	Two dogs are playing with a ball.	entailment

Table 6.2: SNLI Data Triplets, randomly sampled from the test dataset.

W_s	-	W_d	-	SG_s	$Attr_s$	Cat_s	-	SG_d	$Attr_d$	Cat_d	-	$Args_{sd}$	$Nargs_{sd}$
1	-	1	-	1	0	0	-	1	0	0	-	1	0

Table 6.3: DAMR Bridge Feature Ablation Notation. A "1" indicates that the feature is enabled. In this example we illustrate the meaning of 1-1-100-100-10, where features are limited to: word embeddings for source and destination, subgraph types for source and destination, and Args for the relation between source and destination.

Ref	Author	Model	Test Acc
①	[Bowman et al., 2015]	100D LSTM encoders	77.6
②	[Bowman et al., 2016]	300D LSTM encoders	80.6
③	[Vendrov et al., 2015]	1024D GRU encoders w/ unsupervised 'skip-thoughts' pre-training	81.4
④	[Mou et al., 2015]	300D Tree-based CNN encoders	82.1
⑤	[Merity, 2016]	CBOW w/projection	82.5
⑥	**DAMR Experiments**	**1-0-000-000-00 1 Row**	**82.5**
⑦	**DAMR Experiments**	**1-1-111-111-11 2 Rows**	**82.5**
⑧	**DAMR Experiments**	**CBOW Projected**	**82.6**
⑨	**DAMR Experiments**	**1-1-111-111-11 1 Row**	**83.1**
⑩	[Bowman et al., 2016]	300D SPINN-PI encoders	83.2
⑪	[Munkhdalai and Yu, 2016b]	300D NTI-SLSTM-LSTM encoders	83.4
⑫	**DAMR Experiments**	**1-0-111-111-11 1 Row + CBOW Projected**	**83.6**
⑬	[Liu et al., 2016]	600D (300+300) BiLSTM encoders with inner-attention	84.2
⑭	[Munkhdalai and Yu, 2016a]	300D NSE encoders	84.6

Table 6.4: SNLI Performance Accuracies for Various Models, including DAMR Experiments. All DAMR Experiments use LSTM, Sentence Vector, and Classifier Output widths of 300.

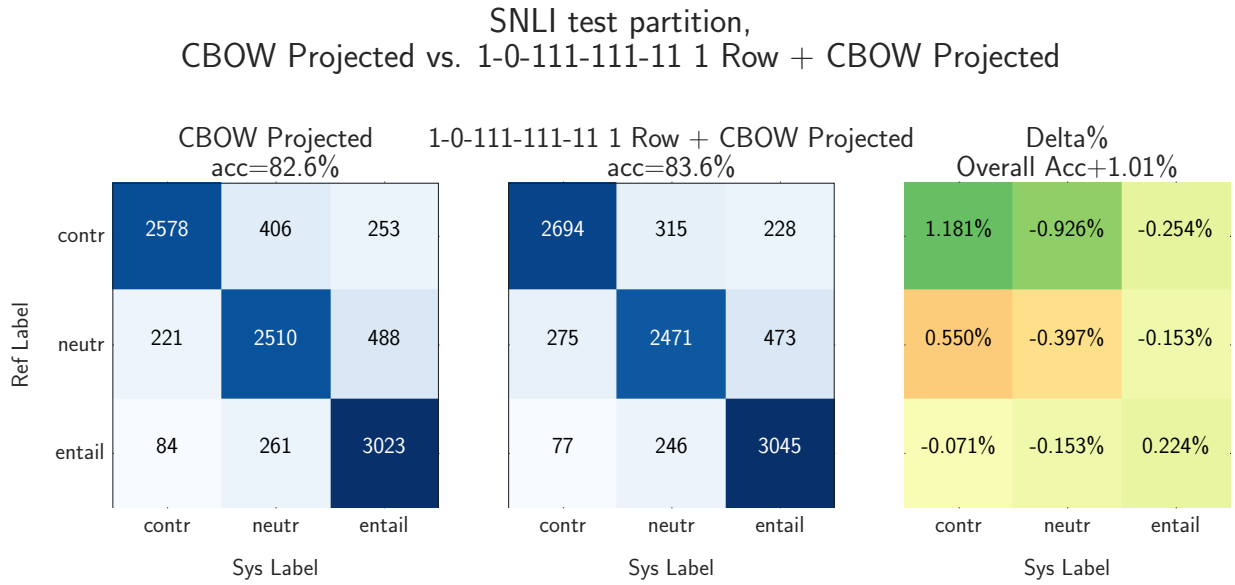


Figure 6.8: SNLI CBOW vs. 1-0-111-111-11 1 Row + CBOW-Projected

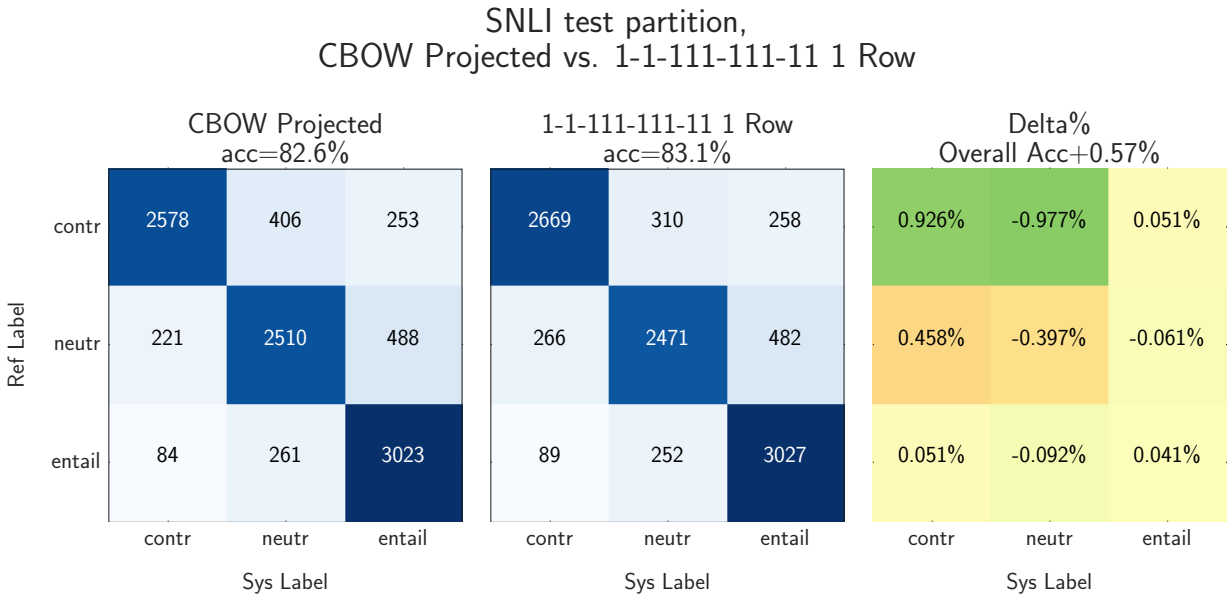


Figure 6.9: SNLI CBOW vs. 1-1-111-111-11 1 Row

6.4 MultiNLI

While the SNLI corpus improved significantly over the RTE and SICK datasets, it still has some recognized room for improvement. The image caption premises in SNLI represent a limited subset of language that describes scenes, and are short and simple. The simplicity of the sentences makes it difficult to sort out the effectiveness of differing models, because there is not much performance margin to distinguish them. This is particularly evident with models that process the two sentences together, using attention or memory based mechanisms to discern inference relations. The restricted origin of the sentences leaves out a lot of the variation associated with language, which makes the models less general.

The latest NLI corpus is called Multi-NLI [Williams et al., 2017], and it maintains the large dataset advantages (433k pairs), but represents english language from ten different genres. These genres are derived from both written text and transcribed speech, and were selected to cover a range of styles, degrees of formality, and topics. Five are used for training data, and there are held-out datasets from those five genres that are referred to as the matched_dev and matched_test data. There are also held-out datasets for the other five genres, which are referred to as mismatched. The matched and mismatched test datasets had not yet been released to the public during our DAMR experimentation, they are being used in conference challenge. We therefore randomly divided the dev dataset into two parts, both representing in-domain genres, and used one half as a development set, the other as a held-out test dataset. The out-of-domain dataset was used without repartitioning. This leaves us with about 1000 of each of the in-domain genre for each of dev and test, and a full 2000 of each of the out of domain genres for test. See Table 6.5.

The in-domain, or matched, genres, as described in [Williams et al., 2017], are:

- FICTION: contemporary fiction written between 1912 and 2010, spanning crime, mystery, humor, western, adventure, science fiction, and fantasy. The authors of these works include Isaac Asimov, Agatha Christie, Ben Essex (Elliott Gesswell), Nick Name (Piotr Kowalczyk), Andre Norton, Lester del Ray, and Mike Shea. See Table 6.8.

Genre	Train	Dev.	Test	Available	ESIM	CBOW
SNLI	550,152	10,000	10,000	9,800	86.7%	80.6%
FICTION	77,348	2,000	2,000	980	73.0%	67.5%
GOVERNMENT	77,350	2,000	2,000	988	74.8%	67.5%
SLATE	77,306	2,000	2,000	967	67.9%	60.6%
TELEPHONE	83,348	2,000	2,000	990	72.2%	63.7%
TRAVEL	77,350	2,000	2,000	982	73.7%	64.6%
9/11	0	2,000	2,000	1974	71.9%	63.2%
FACE-TO-FACE	0	2,000	2,000	1974	71.2%	66.3%
LETTERS	0	2,000	2,000	1977	74.7%	68.3%
OUP	0	2,000	2,000	1961	71.7%	62.8%
VERBATIM	0	2,000	2,000	1946	71.9%	62.7%
MultiNLI Overall	392,702	20,000	20,000	22.3	72.2%	64.7%

Table 6.5: Various MultiNLI Performance Accuracies From [Williams et al., 2017]

- GOVERNMENT: reports, speeches, letters, and press releases from public domain government websites. See Table 6.9.
- SLATE: articles on popular culture, written between 1996-2000, taken from the archives of Slate Magazine. See Table 6.10.
- TELEPHONE: transcriptions of two-sided, conversations held in 1990 to 1991 by speakers of both sexes from several major dialect regions, taken from the University of Pennsylvanias Linguistic Data Consortium Switchboard corpus. See Table 6.12.
- TRAVEL: travel guides discussing vacation and traveling abroad, released in the early 2000s by Berlitz Publishing. See Table 6.14.

The out-of-domain, or mismatched, genres, as described in [Williams et al., 2017], are:

- 9/11: report on the September 11th 2001 terrorist attacks, released on July 22, 2004 by the National Commission on Terrorist Attacks Upon the United States. See Table 6.16.
- FACE-TO-FACE: transcriptions of two-sided in-person conversations from the Charlotte,

NC area in the early 2000s, taken from the the Charlotte Narrative and Conversation Collection. See Table 6.17.

- LETTERS: letters promoting fundraising for non-profit organizations written in late 1990s-early 2000s, collected by The Indiana Center for Intercultural Communication of Philanthropic Fundraising Discourse. See Table 6.18.
- OUP: five non-fiction works published by Oxford University Press on the textile industry and child development. See Table 6.19.
- VERBATIM: articles from a quarterly magazine containing short posts about language and linguistics for non-specialists, written between 1990 and 1996, taken from the Verbatim archives. See Table 6.20.

We use the training data as specified for training networks, dev for selecting the best parameters during training. We later show the results for the two test datasets in detail, on a per genre basis.

Table 6.6 shows Multi-NLI In-Domain Genre experiments accuracy in context with accuracy of other relevant published sentence encoding models for the Multi-NLI In-Domain Corpus. Table 6.6 shows In-Domain Genre experiment accuracy, and a confusion matrix of results is shown in Figure 6.10.

The circled identifiers will be used to reference the models in the following discussion. Different letters represent different models, the number refer to in-domain vs. out-of-domain, for example (A1) and (A2) refer to the same model, run on in-domain and out-of-domain datasets.

6.4.1 In-Domain Performance Discussion

We first discuss the results for the in-domain dataset, summarized in Table 6.6. The progression of adding features was meant to explore the benefit using DAMR.

(A1) CBOW [Williams et al., 2017] is a baseline CBOW model which was used to initially test Multi-NLI.

(B1) CBOW Projected is the optimized CBOW model developed for DAMR bridge testing, which outperforms the baseline significantly due to improved architecture, primarily word projection, multi-layer classification, batch normalization, dropout, and parameter tuning. The model serves as a better baseline for evaluating DAMR performance impact, and is used as a reference in most of the confusion chart results presented for various networks and genres. See figures 6.12 through 6.15.

(C1) 1-0-000-000-00 is a single LSTM row with the projected word embeddings for the sentence as input. The accuracy of this model is about 1.2% better than the CBOW Projected model.

(D1) 1-0-100-000-00 adds the distributed subgraph representation feature for source words to model (C1), and improves accuracy by 0.9%.

(E1) 1-0-111-000-00 adds the distributed attribute and category feature to model (D1), improving accuracy by about 0.1%.

(F1) 1-1-100-100-00 adds the word embedding and distributed subgraph representation feature for destination words to model (D1), and drops accuracy by -0.8%.

(G1) 1-1-111-111-00 adds the distributed attribute and accuracy representation features for source and destination words to model (F1), without any accuracy effect.

(H1) 1-1-100-100-10 adds the word embedding for destination, the distributed subgraph representation features for source and destination words, and the distributed Args feature to model (C1), and improves accuracy by 0.9%, the same improvement as for model (D1).

(I1) 1-1-111-111-11 uses all DAMR features for one row, but performs about 0.5% worse than models (D1) and (H1).

(J1) ESIM [Chen et al., 2017] is an Enhanced Sequential Inference Model, implemented by [Williams et al., 2017] as a baseline model which was used to initially test Multi-NLI, selected because it showed the best accuracy for the SNLI dataset at the time (May, 2017). Although the model exploits information from both premise and hypothesis together, without independent sentence vector encoding, it is presented here for comparison, and as one of the few published

models for MultiNLI performance to date.

In summary, the ablation study for the in-domain Multi-NLI corpus shows some positive impact from using the DAMR bridge over just LSTM processing of word representations, improving performance by 0.9% for two of the models ((E1) and (H1) . It did not constently show improvement as feature complexity was increased. This is possibly due to a lack of tuning, perhaps more dropout is necessary due to overfitting. The models were intentionally not trained individually to avoid biasing the study, and widths were restricted to 80 for LSTM and 100 for the classifier in order to speed up training.

6.4.2 Out-of-Domain Performance Discussion

We next discuss the results for the out-of-domain dataset, summarized in Table 6.7, a confusion matrix for results is shown in Figure 6.11. The same models, and the feature progression used to create them, were used as in the in-domain analysis.

(A2) CBOW [Williams et al., 2017] is a baseline CBOW model which was used to initially test Multi-NLI.

(B2) CBOW Projected is the optimized CBOW model developed for DAMR bridge testing. All out-of-domain models were compared with this model, details are shown in figures 6.14 through 6.18.

(C2) 1-0-000-000-00 is a single LSTM row with the projected word embeddings for the sentence as input. The accuracy of this model is about 1.6% better than (B2) . This model performance is almost as good as circledE1, and already represents nearly the highest accuracy of the OOD experiments.

(D2) 1-0-100-000-00 adds the distributed subgraph representation feature for source words to model (C2) , and accuracy decreases by -0.1%.

(E2) 1-0-111-000-00 adds the distributed attribute and category feature to model (D2) , improving accuracy by about 0.1%, getting the performance up to slightly better than the simple (B2) .

(F2) 1-1-100-100-00 adds the word embedding and distributed subgraph representation feature for destination words to model (D2), and drops accuracy by -0.2%.

(G2) 1-1-111-111-00 adds the distributed attribute and accuracy representation features for source and destination words to model (F2), and accuracy drops by -0.4%.

(H2) 1-1-100-100-10 adds the word embedding for destination, the distributed subgraph representation features for source and destination words, and the distributed Args feature to model (C2), and accuracy reduces by 0.3%.

(I2) 1-1-111-111-11 uses all DAMR features for one row, but performs about 0.4% worse than models (C2) and (E2).

(J2) ESIM [Chen et al., 2017] is the Enhanced Sequential Inference Model, explained earlier.

In summary, the ablation study for out-of-domain data shows little evidence of positive impact from using the DAMR bridge over just LSTM processing of word representations. It did not consistently show improvement as feature complexity was increased, it is more a random effect. The performance indicates that the models may be overfitting to the in-domain training data. This is possibly due to a lack of model tuning, perhaps more dropout is necessary to prevent overfitting. These are the same models used for in-domain testing, and the models were intentionally not trained individually to avoid biasing the study. This dataset looks promising, so the results are surprising. Published results from other models and ablations should help to provide context for the unexpected model performance.

Ref	System	Accuracy						
		dev	test (all ID)	fiction	gov	slate	telephone	travel
A1	CBOW [Williams et al., 2017]	-	64.8%	67.5%	67.5%	60.6%	63.7%	64.6%
B1	CBOW Projected	69.0%	68.7%	69.7%	**74.7%**	61.8%	70.6%	66.4%
C1	1-0-000-000-00	70.0%	69.8%	69.4%	72.9%	63.0%	72.2%	71.4%
D1	1-0-100-000-00	70.3%	70.6%	70.2%	72.9%	65.5%	72.4%	71.7%
E1	1-0-111-000-00	70.5%	**70.7%**	70.0%	74.0%	**65.7%**	**72.8%**	71.1%
F1	1-1-100-100-00	70.6%	69.8%	70.8%	72.2%	63.7%	71.6%	70.4%
G1	1-1-111-111-00	70.0%	69.8%	70.3%	72.3%	63.8%	71.5%	71.1%
H1	1-1-100-100-10	**70.8%**	**70.7%**	**71.4%**	73.8%	65.5%	70.9%	**71.9%**
I1	1-1-111-111-11	70.4%	70.2%	70.0%	73.0%	64.4%	72.3%	71.1%
J1	ESIM [Williams et al., 2017]	-	72.3%	73.0%	74.8%	67.9%	72.2%	73.7%

Table 6.6: Multi-NLI In-Domain Genre DAMR Experiment Results. All DAMR Models have 100 width output and 80 width LSTM hidden states.

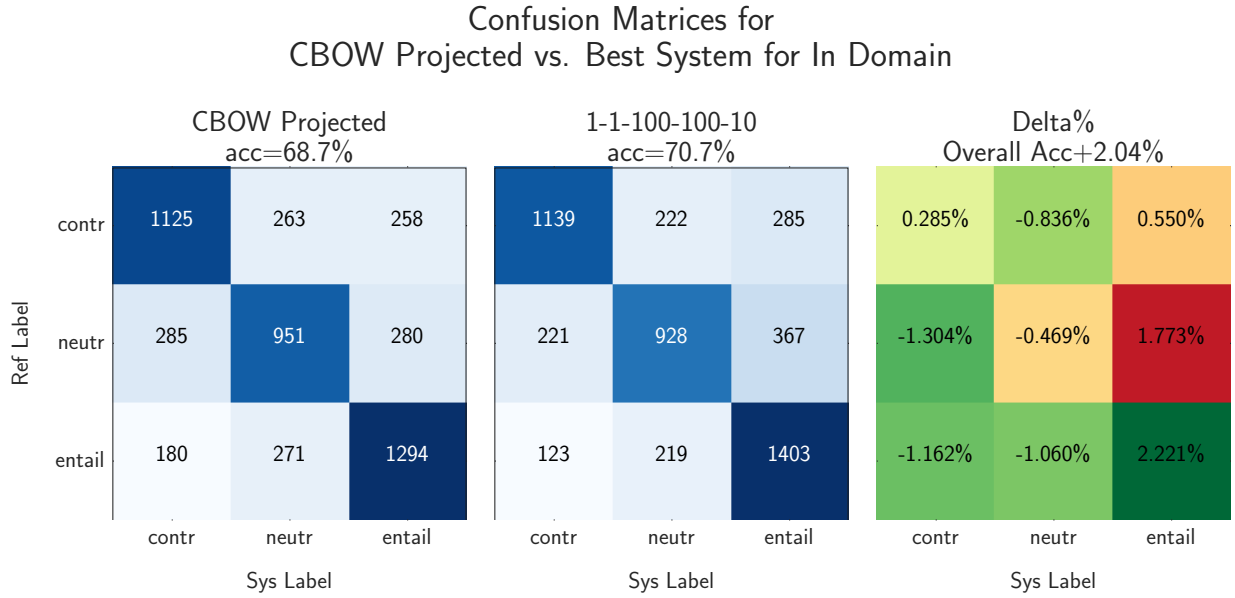


Figure 6.10: Confusion matrix shows the results for the best performing In-Domain Multi-SNLI model, labeled (H1) 1-1-100-100-10, which is a single row LSTM.

id	System	Accuracy						
		dev	all OOD (test)	9/11	facetoface	letters	oup	verbatim
A2	CBOW [Williams et al., 2017]	-	64.7%	63.2%	66.3%	68.3%	62.8%	62.7%
B2	CBOW Projected	69.0%	69.1%	67.0%	71.4%	72.8%	68.6%	65.6%
C2	1-0-000-000-00	70.0%	**70.7%**	69.3%	71.9%	73.7%	**70.6%**	67.7%
D2	1-0-100-000-00	70.3%	70.6%	68.6%	72.0%	74.8%	69.6%	68.2%
E2	1-0-111-000-00	70.5%	**70.7%**	**69.5%**	72.2%	74.5%	69.7%	67.5%
F2	1-1-100-100-00	70.6%	70.5%	67.8%	71.8%	73.4%	**70.6%**	**68.9%**
G2	1-1-111-111-00	70.0%	70.1%	68.8%	70.4%	**75.0%**	69.5%	66.9%
H2	1-1-100-100-10	**70.8%**	70.4%	68.4%	**73.1%**	74.5%	68.8%	67.4%
I2	1-1-111-111-11	70.4%	70.3%	68.5%	71.4%	74.3%	69.3%	67.8%
J2	ESIM [Williams et al., 2017]	-	72.3%	71.9%	71.2%	74.7%	71.7%	71.9%

Table 6.7: Multi-NLI Out-of-Domain Genre DAMR Experiment Results. All DAMR Models have 100 width output and 80 width LSTM hidden states.

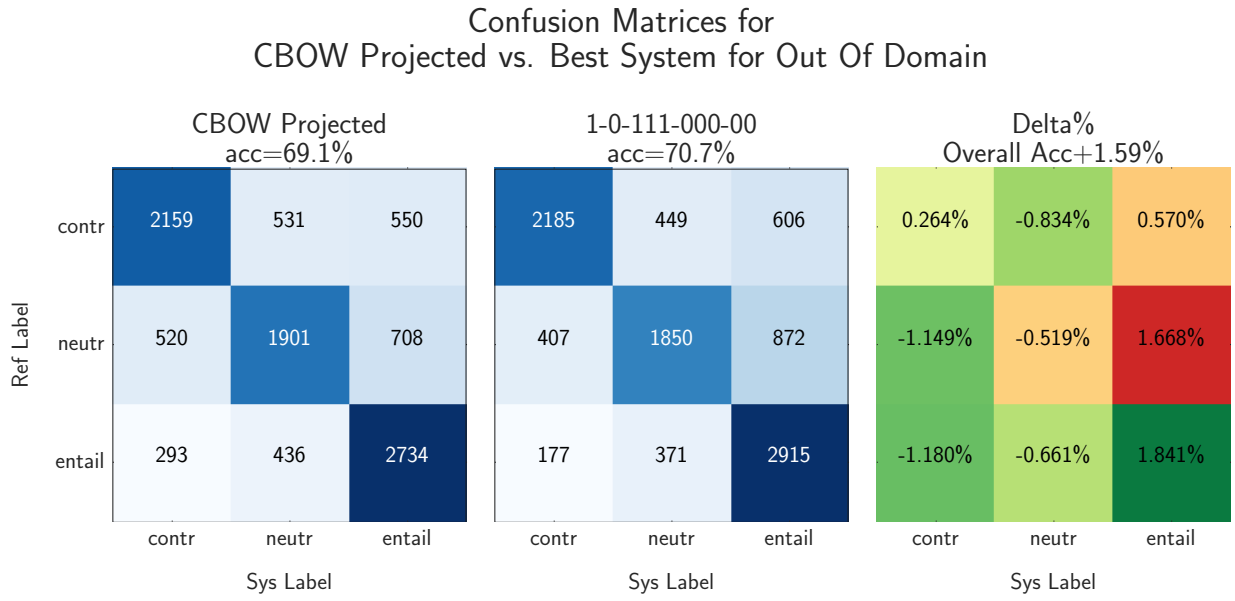


Figure 6.11: Confusion matrix shows the results for the best performing Out-of-Domain Multi-SNLI model, labeled E2 1-0-111-000-00.

Genre	Premise	Hypothesis	Label
fiction	I was pulled into the bar.	I managed to escape their grasp and ran just outside the bar.	contradiction
		The lure of alcohol was unrelenting and the bar pulled me in.	neutral
		I was dragged through the door of the pub.	entailment
fiction	Ca'daan's mouth hung open.	Ca'daan kept his mouth shut.	contradiction
		Ca'daan's mouth was cut wide open.	neutral
		Ca'daan had his mouth wide open.	entailment
fiction	He had forgotten about Adrin.	He remembered Adrin all this time.	contradiction
		He had forgotten that Adrin was going to join them.	neutral
		He didn't remember Adrin.	entailment
fiction	I felt an immeasurable 230 contempt for him .	I felt great respect for him...	contradiction
		I had no reason for feeling the way I did about him.	neutral
		I felt intense disrespect for him...	entailment

Table 6.8: Multi-NLI In-Domain Fiction Genre Examples, randomly selected.

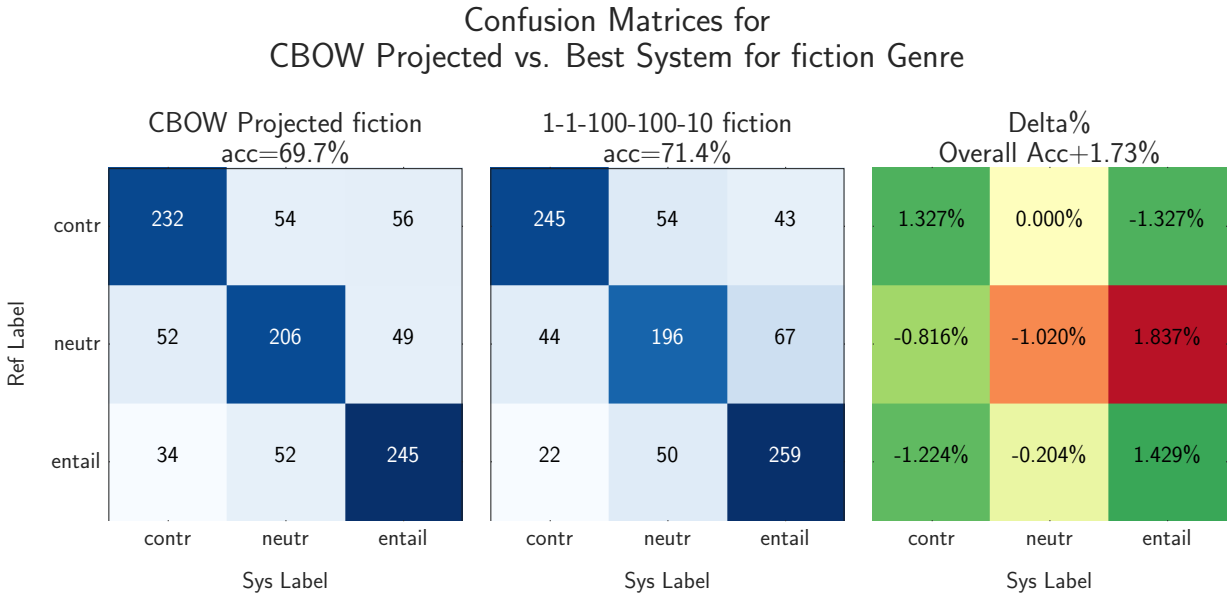


Figure 6.12: Confusion matrix compares the results of the CBOW Projected model against the best performing model, labeled (H1) 1-1-100-100-10, for the In-Domain Fiction Genre.

Genre	Premise	Hypothesis	Label
government	The Congress, which controls our funding levels, began to include many members who did not support the purpose and goals of a federal civil legal services program.	the congress has no responsibility when it comes to controlling funding levels.	contradiction
		the congress also has different functions, though they are all broad scoping.	neutral
		the congress is generally responsible for controlling funding levels.	entailment
government	They are levied through the power of the Government to compel payment, and the person or entity that pays these fees does not receive anything of value from the Government in exchange.	They are not levied through the power of the Government to compel payment.	contradiction
		They are levied through the power of the Government to compel payment in cash only	neutral
		They are levied through the power of the Government to compel payment.	entailment

Table 6.9: Multi-NLI In-Domain government Genre Examples, randomly selected.

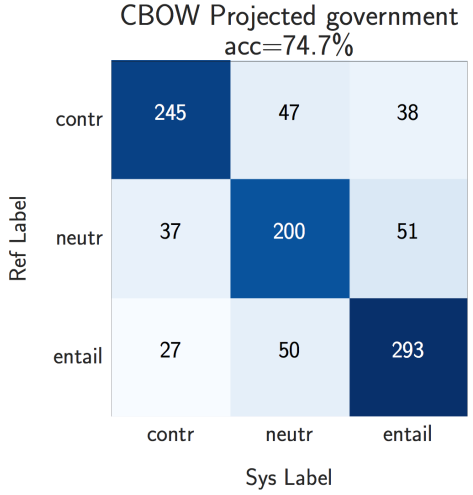


Figure 6.13: Confusion matrix for the CBOW Projected model, which had the best accuracy for the In-Domain government Genre.

Genre	Premise	Hypothesis	Label
slate	What's needed, alongside an evacuation plan, is a realistic program to stabilize conditions for those left behind.	Once everyone has been evacuated as much as possible, we can't worry about those left behind.	contradiction
		Evacuation is always the first line of response.	neutral
		Those left behind will need a program for stabilizing conditions if they cannot evacuate.	entailment
slate	5) The Democrats are reaping what they sowed (after torturing Robert Bork, John Tower, and Clarence Thomas).	Democrats rarely have any political relevance.	contradiction
		Democrats are replacing many Republicans because they were always very forceful in their approach.	neutral
		After torturing Robert Bork, John Tower, and Clarence Thomas, the Democrats are getting what they deserved.	entailment

Table 6.10: Multi-NLI In-Domain Slate Genre Examples, randomly selected.

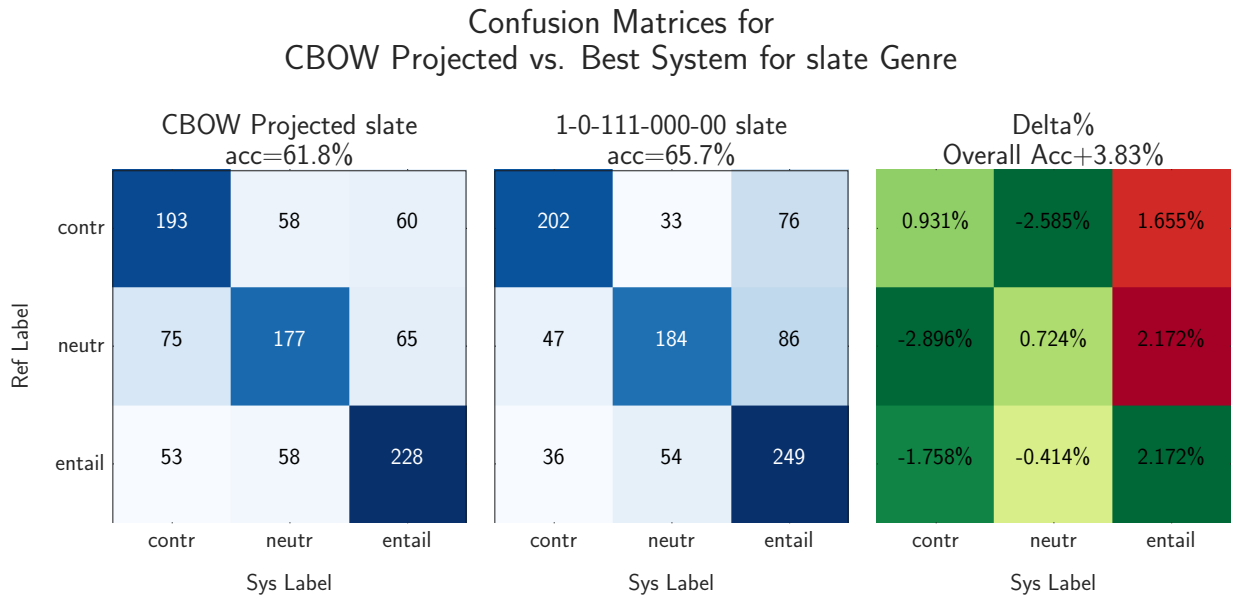


Table 6.11: Confusion matrix compares the results of the CBOW Projected model against the best performing model, labeled (E1) 1-0-111-000-00, for the In-Domain Slate Genre.

Genre	Premise	Hypothesis	Label
telephone	oh boy it the i think it's like one or the other isn't it i mean you either	It's definitely that one.	contradiction
		I think it's one or the other kind of shirt.	neutral
		I think it's one or the other.	entailment
telephone	um-hum yeah that's very true you know how many is it they say we have so many lawyers in this country and i guess i i live near Washington being in in Baltimore it's something like one in four people in the Washington	There are barely any lawyers in this country.	contradiction
		We don't need so many lawyers.	neutral
		There are a lot of lawyers in this country.	entailment
telephone	um yeah that sounds kind of neat uh is location at all important to you like you know how far it is from your house or whatever	That sounds really stupid.	contradiction
		Location doesn't matter to some people but it may matter to you I don't know.	neutral
		If something is far from your house does it matter to you, is location important to you?	entailment

Table 6.12: Multi-NLI In-Domain Slate Genre Examples, randomly selected.

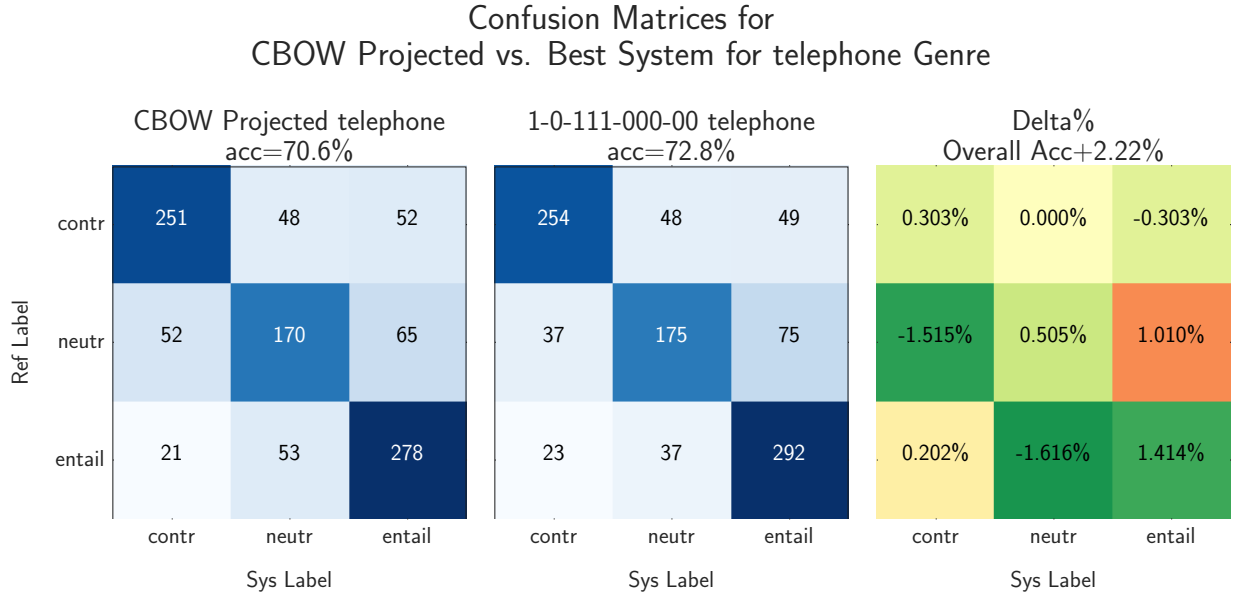


Table 6.13: Confusion matrix compares the results of the CBOW Projected model against the best performing model, labeled (E1) 1-0-111-000-00, for the In-Domain Telephone Genre.

Genre	Premise	Hypothesis	Label
travel	It recalls William Randolph Hearst's castle in Caleornia, with its imaginative juxtaposition of ancient Roman and Chinese sculpture, fine Venetian glass chandeliers, Syvres porcelain, old Flemish masters, and naughty French erotica.	There is no art or sculptures located there.	contradiction
		William Randolph Hearst didn't love chandeliers, but kept them to make his lady happy.	neutral
		William Randolph Hearst's castle housed a collection of sculptures and other fine arts.	entailment
travel	Growth continued for ten years, and by 1915 the town had telephones, round-the-clock electricity, and a growing population many of whom worked in the railroad repair shop.	Growth was stifled, and most of the population couldn't find work.	contradiction
		The town was hooked up to the electricity and telephone grids because of its geographical importance.	neutral
		Economic growth continued apace, with many people employed by the railroad repair shop.	entailment

Table 6.14: Multi-NLI In-Domain Travel Genre Examples, randomly selected.

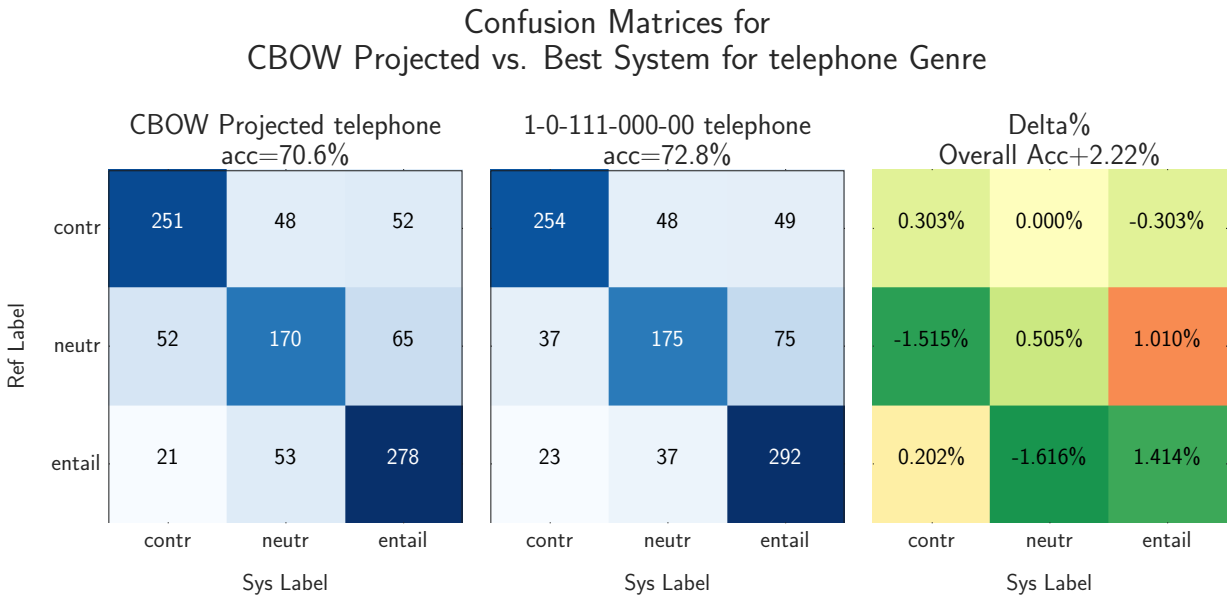


Table 6.15: Confusion matrix compares the results of the CBOW Projected model against the best performing model, labeled (H1) 1-1-100-100-10, for the In-Domain Travel Genre.

Genre	Premise	Hypothesis	Label
nineeleven	United 93 crashed in Pennsylvania at 10:03:11, 125 miles from Washington, D.C.	United 93 never crashed and is still flying to this day.	contradiction
		The United 93 crash was caused by the passengers.	neutral
		125 miles away from DC, United 93 crashed.	entailment
nineeleven	Some family members who listened to the recording report that they can hear the voice of a loved one among the din.	No one was allowed to listen to the recordings.	contradiction
		The loved ones of the victims heard screaming on the recording.	neutral
		Folks who listened to the tape heard their family members on it.	entailment

Table 6.16: Multi-NLI Out-of-Domain NineEleven Genre Examples, randomly selected.

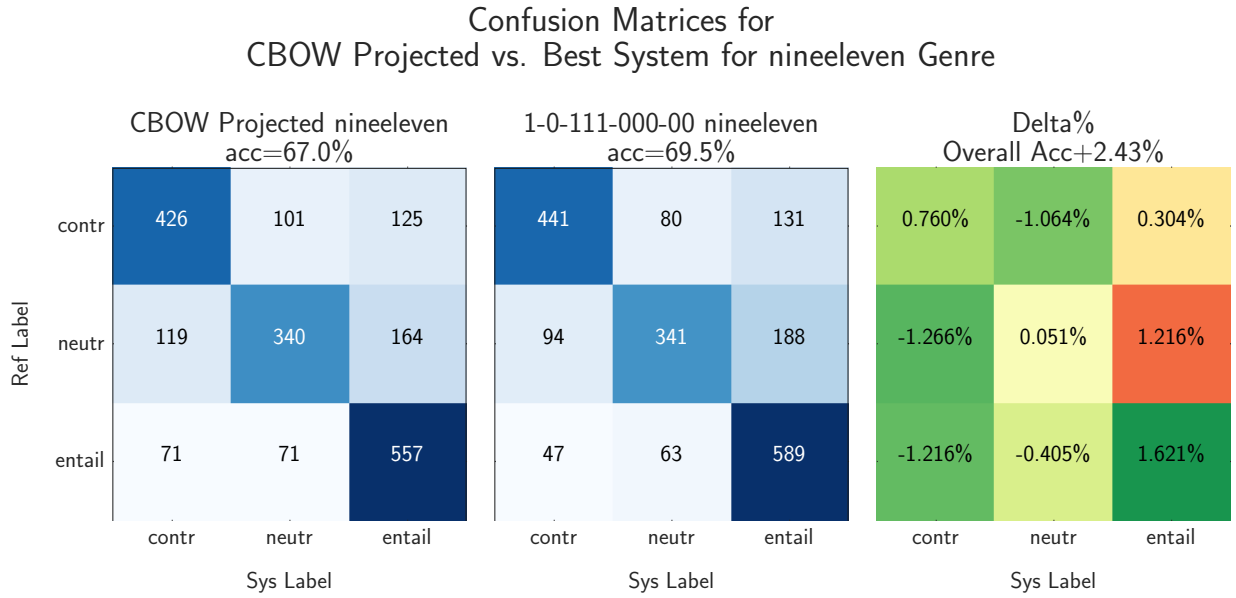


Figure 6.14: Confusion matrix compares the results of the CBOW Projected model against the best performing model, labeled (E2) 1-0-111-000-00, for the Out-of-Domain Nine Eleven Genre.

Genre	Premise	Hypothesis	Label
facetoface	I don't, I kind of don't think about that though, because I'm like my life is my life, you know?	I think about it every day of my life.	contradiction
		I do what I want with my life.	neutral
		This is my life I just do not want to think about it.	entailment
facetoface	That was the deal for them to get off my back and let me do what I wanted to do.	They never cared and I always did what I wanted.	contradiction
		I agreed to being financially cut-off so that I could do what I wanted.	neutral
		I made a deal with them so that I could make my own decisions.	entailment
facetoface	But, then like me and my mom will talk all the time and she'll come over here and she'll like have a beer and a cigarette with me and it doesn't matter anymore!	I would never dare drink in front of my mother.	contradiction
		My mom and I both like to have the same kind of beer.	neutral
		My mom and I hang out all the time and chat, we even share a drink.	entailment

Table 6.17: Multi-NLI Out-of-Domain Face to Face Genre Examples, randomly selected.

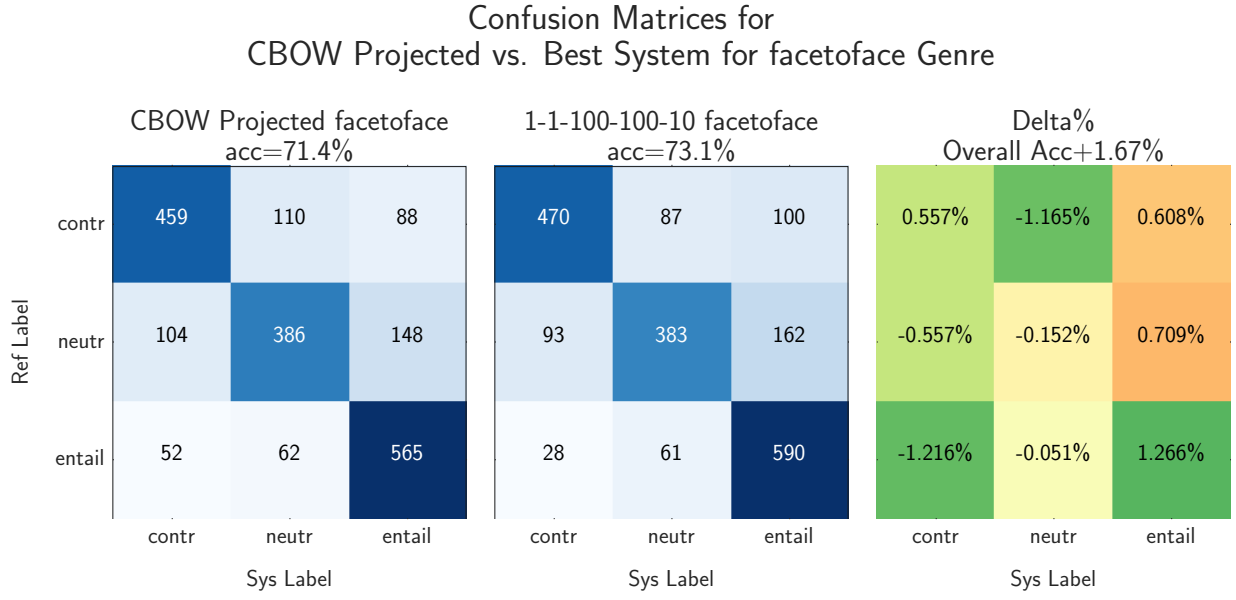


Figure 6.15: Confusion matrix compares the results of the CBOW Projected model against the best performing model, labeled (H2) 1-1-100-100-10, for the Out-of-Domain Face to Face Genre.

Genre	Premise	Hypothesis	Label
letters	Information regarding inadequate classrooms, moot court facilities, offices, the library and support service areas follow the above paragraphs.	All of the adequate spaces are listed in the paragraphs above.	contradiction
		There are too many professional spaces with inadequacies.	neutral
		Information about inadequate community or services is listed above.	entailment
letters	These are challenging times for public institutions of higher education, with legislative appropriations unable to fund schools to the level they have in the past.	Public institutions are being funded fine and we don't need your help.	contradiction
		No one cares about public education anymore so schools aren't being funded like they used to.	neutral
		Legislative appropriations can't fund schools as well as they could in the past.	entailment

Table 6.18: Multi-NLI Out-of-Domain Letters Genre Examples, randomly selected.

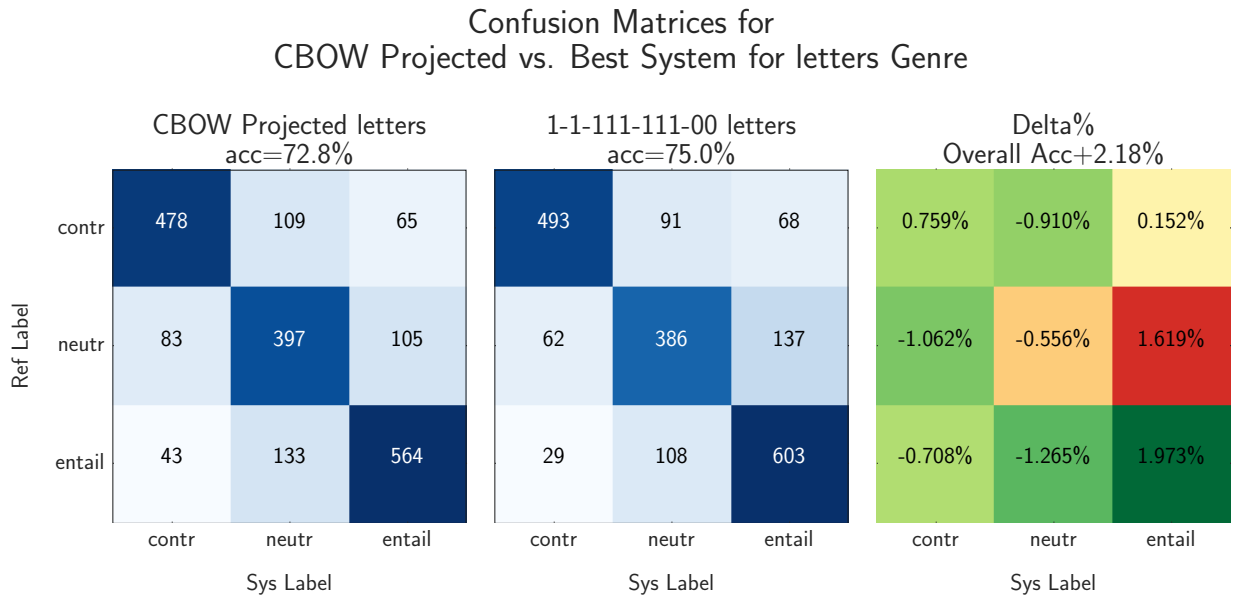


Figure 6.16: Confusion matrix compares the results of the CBOW Projected model against the best performing model, labeled (G2) 1-1-111-111-00, for the Out-of-Domain Letters Genre.

Genre	Premise	Hypothesis	Label
oup	The changes now going on have their analog in the last century, when technological innovations of the day like railroads, telegraph, and steam power'developed for purposes far afield of retail, apparel, or textiles'helped transform the mass distri-bution of goods and information.	Steam power definitely did not have any bearing on the distribution of goods and information.	contradiction
		The telegraph had more of a bearing on information than the railroad.	neutral
		Mass distribution of goods and information was affected by technological innovations such the steam power.	entailment
oup	In colonial days, housewives typically did spinning, weaving, and tailoring for the family.	Housewives went out hunting in colonial days.	contradiction
		Housewives also took care of children in colonial days.	neutral
		Housewives usually did basic textile works for the family in colonial days.	entailment

Table 6.19: Multi-NLI Out-of-Domain OUP Genre Examples, randomly selected.

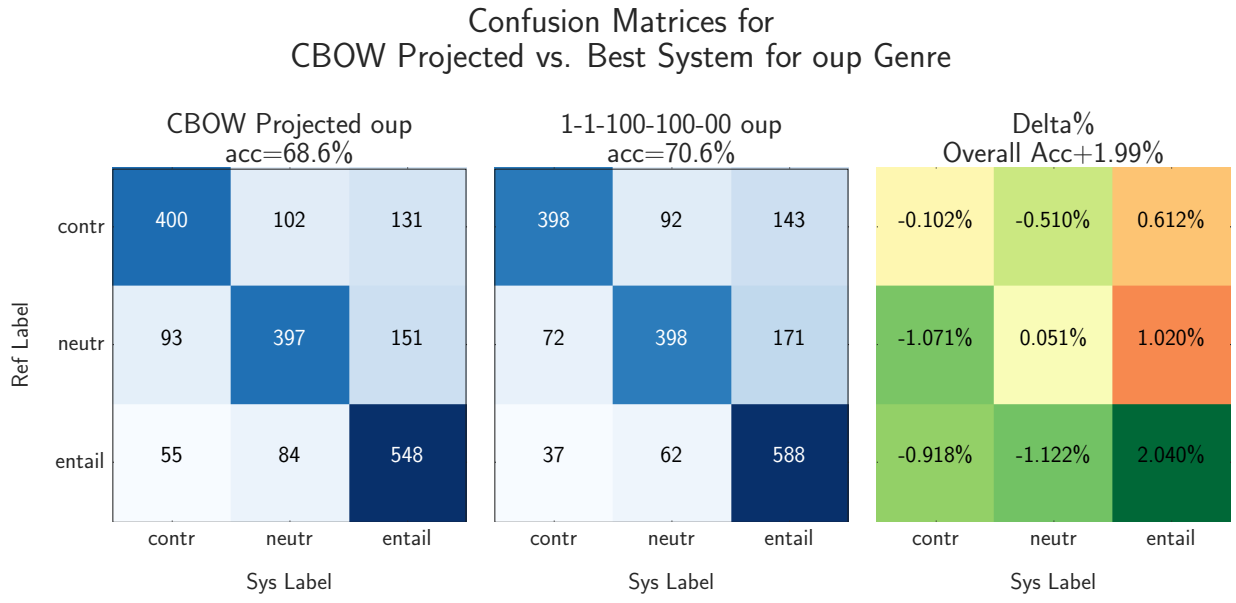


Figure 6.17: Confusion matrix compares the results of the CBOW Projected model against the best performing model, labeled (F2) 1-1-100-100-00, for the Out-of-Domain OUP (Oxford University Press) Genre.

Genre	Premise	Hypothesis	Label
verbatim	It is difficult to make any sensible connection between the lives of authors and their creations.	It is very easy to connect the author's life to their writing.	contradiction
		Authors rarely write about their life experiences.	neutral
		It's hard to make any sort of connection between the author's life and their work.	entailment
verbatim	The soldier in charge would command, after each firing, that the rank on the scaffold step down and be replaced by the rank that had just reloaded, thus alternating ranks and sustaining the rifle fire.	The commanding soldiers just stayed and fought themselves with no rotation.	contradiction
		This system does not seem like it would have worked.	neutral
		The commanding soldier demanded after each firing that the rank be reloaded.	entailment

Table 6.20: Multi-NLI Out-of-Domain Verbatim Genre Examples, randomly selected.

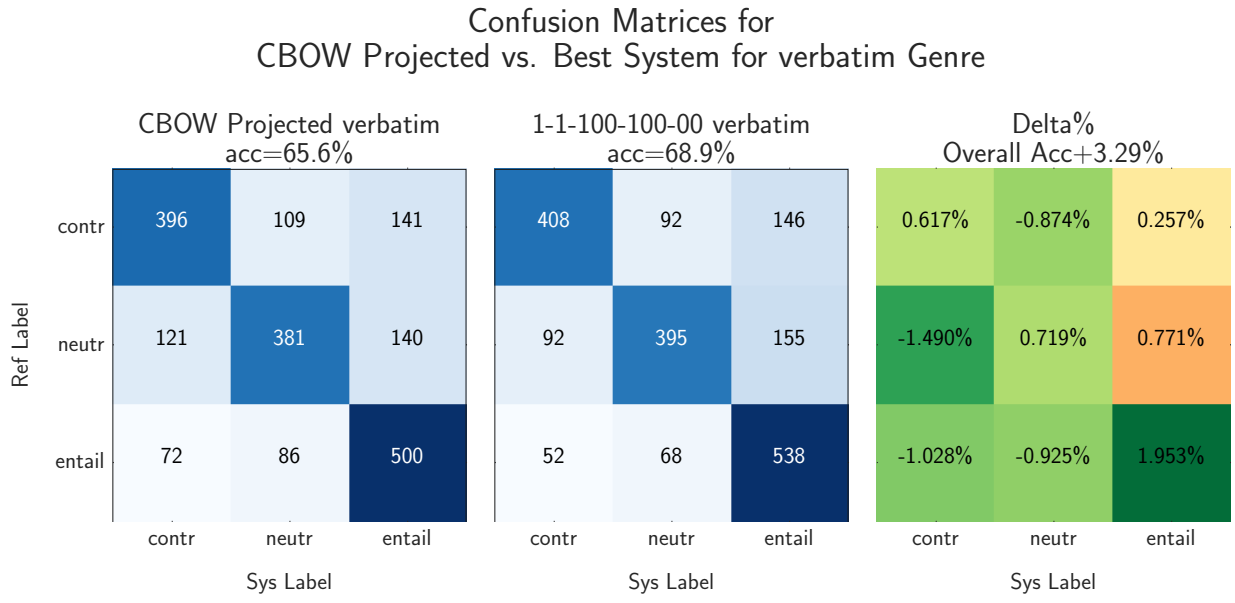


Figure 6.18: Confusion matrix compares the results of the CBOW Projected model against the best performing model, labeled (F2) 1-1-100-100-00, for the Out-of-Domain Verbatim Genre.

6.5 Conclusion

In this chapter, we investigated the use of a DAMR Bridge model to create semantic sentence vectors from an AMR parser, and tested these representation using natural language inference.

One motivation for converting DAMR to sentence vectors is to provide a more semantically-driven AMR test approach in contrast with the symbolic graph comparison approach currently in use. Another is to investigate the quality of the vector meaning representations themselves, as a novel approach to generate sentence vectors.

Two corpora were described and tested, for a series of models designed to check the effect of adding more and more information from the DAMR representations. The results were mixed. The ablation studies did not show consistent performance increases with increasing feature contribution, but SNLI performance was improved with DAMR by about 1% over either a CBOW model or LSTM of word embeddings.

Testing with in-domain Multi-NLI corpus data shows an accuracy improvement of about 0.9% for the best two DAMR model configurations. Testing using the DAMR bridge with out-of-domain Multi-NLI data, however, did not improve accuracy over the optimized CBOW model.

The difference between in and out of domain performance indicates that the model may be overfitting to the in-domain training data. A possible option for the future is to expand the widths of the bridge LSTM and classifier outputs, since these were held at fairly narrow widths in order to be able to run more experiments and to help to prevent model tuning from being a factor in the ablation study. Increasing the dropout rates for the various portions of the model can also be explored to prevent any overfitting that might be occurring.

Chapter 7

Conclusions

Neural networks, especially bidirectional LSTM networks, are able to analyze sequences efficiently and accurately, and they can be used to create very effective models for natural language. The convolutional neural network architecture chosen for the semantic role labeler described in chapter 3, can be improved by using the LSTM-based approaches of later chapters. Convolutional networks impose a local context window, and are not able to handle long-term dependencies within a sentence as well as recurrent networks like LSTM.

The ablation studies for semantic role labeler show that the addition of path information from a dependency parser as a feature improved performance.

There was no need to identify the predicates since they were given in the dataset. However, predicates needed to be disambiguated by determining the predicate sense label. Because the sense labels don't generalize well between predicates, a set of model weights was maintained separately for each predicate in the training dataset.

While the AMR Parser described in Chapter 4 does perform well against other methods currently in use, the 66% smatch score loosely implies that a third of the semantic information is not properly parsed.

The alignment of words in the sentence to symbols in the AMR is essential for training, but it is not a required part of human annotation and is most commonly supplied by automatic alignment methods. These methods currently have a fairly high error rate, which causes training data to suffer, leading to deteriorated system performance.

AMR parsing requires the identification of predicates, which is done by the SG network. In addition, the SG network identifies the predicate sense for senses 00 and 01, which seem to generalize better than the rest. Having the SG network also decide the verb sense was probably not the best choice. A better architecture might be to first identify general spans using the SG network, and then to use a predicate-specific network for predicate sense disambiguation as was done in the semantic role labeler described in Chapter 3. This sense-identifying model could be trained with resources outside the AMR corpus, such as propbank, in order to improve its performance. The same could be done for identifying semantic roles.

Non-predicate relation identification performance (from Nargs) seems to have room for improvement. Perhaps a finer segmentation of this class of relations could be used to create more specialized handling of them in order to improve performance.

The AMR Parser architecture uses two levels, first, SG, then the rest of the networks. Its possible that performing the parsing using more cascading levels would improve performance. For example, first SG, next Args, then Nargs which uses the output of the first two levels, etc.

By eliminating syntactic pre-parsing, the system resource requirements have been reduced, but the use of an external named entity recognizer still keeps the system dependent on pre-processing. The named entity process has been done with LSTM networks and some readily available gazetteers, and incorporating this technique into the architecture would make it more self-contained. The wikification process is a similar, list dependent task that might also be implemented internally.

In Chapter 5, we created a distributed AMR representation, and evaluated the meaning content it contains by converting it to an AMR, and then using the smatch program to compare with gold AMR's in order to generate an F1 score. The result was about 0.5% better than the process described in Chapter 4, which decides on the identity of SG tags earlier in the program flow.

An expansion of the DAMR concept in the AMR parser would be to make SG decisions based on a combination of SG prob distribution and Arg/Narg distributions, which might lead to a higher quality decision overall. Currently, even though distributed SG representations are used as features for the Args, Nargs, Attr, and Cat networks, the SG is selected based only on the highest

probability tag for SG.

In Chapter 6, we test DAMR quality in a different way, by first creating premise and hypothesis vectors, and then using those vectors as input to an inference classifier. We conducted feature ablation experiments for various models in order to investigate the effect of adding increasing amounts of information from DAMR, but failed to find a direct correlation with system performance. It is difficult to identify the reasons for this lack of direct feature impact, but general AMR quality improvements might make a difference. The AMR training data content and the NLI data content are different from each other, which also might explain lack of the trend we were searching for. Even so, we demonstrated that both the SNLI and MultiNLI results can improved by adding DAMR content to sentence vector models.

DAMR might allow us to make discourse-level meaning extraction more accurate, for both symbolic and distributed inference flows. Investigation into using DAMR as an intermediate, pliable form of meaning representation would be an interesting future direction. DAMR can be modified based on pragmatic, context, or other evidence outside the scope of an individual sentence. This can be done, for example, to disambiguate the sentence before a hard AMR is generated. The same can be done prior to creating sentence vectors, in order to improve the semantic quality of the generated vectors.

Finally, a few thoughts on measuring AMR quality. A graph comparison program called smatch is the primary means of assessing AMR quality from different parser models. The smatch program compares symbols between two AMR graphs using a stochastic, hill-climbing algorithm, and does not use any measure of semantic relevance to create its score. Opposite meaning and slight differences are the same to smatch, making the scores almost irrelevant for true meaning assessment. The DAMR bridge makes using NLI possible because the AMR parser from Chapter 4 creates distributed representations prior to producing a symbolic AMR graph.

If it was possible to start with an AMR and convert it to a distributed form, suitable as a feature for a Bridge model like that discussed in Chapter 6, a semantic comparison of AMR's using NLI would be possible. This would allow work to proceed towards improving the semantic content

described in AMR instead of improving the smatch score, for any AMR parsing architecture.

A difficulty with creating a DAMR from an AMR is that AMR expresses lemmatized symbols. The words *leave, leaves, leaving, left* all might be mapped to the lemma *leave* and assigned a predicate sense in the AMR. There is no generic Glove embedding representation for leave-01, or leave-02, for example. Another aspect to consider is that AMR contains disambiguated symbols, so the representation for the word *left* in **Right, not left** needs to be different from **They left**. Standard word embeddings like Glove will represent both meanings for *left* with a single representation. One solution would be to add representations for lemmatized and disambiguated AMR symbols to an existing semantic space, like that defined by Glove embeddings. We could then use this expanded, large vocabulary embedding space, for mapping from AMR to DAMR. We could then use a bridge model, as described in Chapter 6, to create sentence vectors to feed a natural language inference model in order to measure AMR semantic quality.

Bibliography

[Artzi et al., 2015] Artzi, Y., Lee, K., and Zettlemoyer, L. (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.

[Baker et al., 1998] Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The berkeley framenet project. In Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1, pages 86–90. Association for Computational Linguistics.

[Banarescu et al., 2012] Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., and Schneider, N. (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.

[Banerjee and Pedersen, 2002] Banerjee, S. and Pedersen, T. (2002). An adapted lesk algorithm for word sense disambiguation using wordnet. In Computational linguistics and intelligent text processing, pages 136–145. Springer.

[Barzdins and Gosko, 2016] Barzdins, G. and Gosko, D. (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.

[Bengio et al., 2007] Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. Advances in neural information processing systems, 19:153.

[Björkelund et al., 2010] Björkelund, A., Bohnet, B., Hafdell, L., and Nugues, P. (2010). A high-performance syntactic and semantic dependency parser. In Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations, pages 33–36. Association for Computational Linguistics.

[Björkelund et al., 2009] Björkelund, A., Hafdell, L., and Nugues, P. (2009). Multilingual semantic role labeling. In Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task, pages 43–48. Association for Computational Linguistics.

[Blackburn and Bos, 2005] Blackburn, P. and Bos, J. (2005). Representation and inference for natural language. A first course in computational semantics. CSLI.

[Bos, 2016] Bos, J. (2016). Expressive power of abstract meaning representations. Computational Linguistics, 42(3):527–535.

[Bos and Markert, 2005] Bos, J. and Markert, K. (2005). Recognising textual entailment with logical inference. In Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, pages 628–635. Association for Computational Linguistics.

[Bowman et al., 2015] Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. arXiv preprint arXiv:1508.05326.

[Bowman et al., 2016] Bowman, S. R., Gauthier, J., Rastogi, A., Gupta, R., Manning, C. D., and Potts, C. (2016). A fast unified model for parsing and sentence understanding. CoRR, abs/1603.06021.

[Brandt et al., 2016] Brandt, L., Grimm, D., Zhou, M., and Versley, Y. (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.

[Bridle, 1990] Bridle, J. S. (1990). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In Neurocomputing, pages 227–236. Springer.

[Cai and Knight, 2013] Cai, S. and Knight, K. (2013). Smatch: an evaluation metric for semantic feature structures. In ACL (2), pages 748–752.

[Carreras and Màrquez, 2004] Carreras, X. and Màrquez, L. (2004). Introduction to the conll-2004 shared task: Semantic role labeling. In Proceedings of the Eighth Conference on Computational Natural Language Learning, CoNLL 2004, Held in cooperation with HLT-NAACL 2004, Boston, Massachusetts, USA, May 6-7, 2004, pages 89–97.

[Carreras and Màrquez, 2005] Carreras, X. and Màrquez, L. (2005). Introduction to the conll-2005 shared task: Semantic role labeling. In Proceedings of the Ninth Conference on Computational Natural Language Learning, pages 152–164. Association for Computational Linguistics.

[Chen and Manning, 2014] Chen, D. and Manning, C. D. (2014). A fast and accurate dependency parser using neural networks. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 740–750.

[Chen et al., 2017] Chen, Q., Zhu, X., Ling, Z.-H., Wei, S., Jiang, H., and Inkpen, D. (2017). Enhanced lstm for natural language inference. In Proc. ACL.

[Cheng and Roth, 2013] Cheng, X. and Roth, D. (2013). Relational inference for wikification. In EMNLP.

[Chiu and Nichols, 2015] Chiu, J. P. and Nichols, E. (2015). Named entity recognition with bidirectional lstm-cnns. arXiv preprint arXiv:1511.08308.

[Collobert and Weston, 2008] Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th international conference on Machine learning, pages 160–167. ACM.

[Collobert et al., 2011] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. The Journal of Machine Learning Research, 12:2493–2537.

[Condoravdi et al., 2003] Condoravdi, C., Crouch, D., De Paiva, V., Stolle, R., and Bobrow, D. G. (2003). Entailment, intensionality and text understanding. In Proceedings of the HLT-NAACL 2003 workshop on Text meaning-Volume 9, pages 38–45. Association for Computational Linguistics.

[Dagan et al., 2006] Dagan, I., Glickman, O., and Magnini, B. (2006). The pascal recognising textual entailment challenge. In Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment, pages 177–190. Springer.

[Dai and Le, 2015] Dai, A. M. and Le, Q. V. (2015). Semi-supervised sequence learning. In Advances in Neural Information Processing Systems, pages 3079–3087.

[Duchi et al., 2011] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research, 12:2121–2159.

[Durrett and Klein, 2015] Durrett, G. and Klein, D. (2015). Neural crf parsing. arXiv preprint arXiv:1507.03641.

[Emanuele et al., 2013] Emanuele, B., Castellucci, G., Croce, D., and Basili, R. (2013). Textual inference and meaning representation in human robot interaction. In Joint Symposium on Semantic Processing, page 65.

[Fellbaum, 1998] Fellbaum, C. (1998). WordNet. Wiley Online Library.

[Flanigan et al., 2014] Flanigan, J., Thomson, S., Carbonell, J. G., Dyer, C., and Smith, N. A. (2014). A discriminative graph-based parser for the abstract meaning representation.

[Florian et al., 2003] Florian, R., Ittycheriah, A., Jing, H., and Zhang, T. (2003). Named entity recognition through classifier combination. In Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4, CONLL '03, pages 168–171, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Foland Jr and Martin, 2016] Foland Jr, W. R. and Martin, J. H. (2016). Cu-nlp at semeval-2016 task 8: Amr parsing using lstm-based recurrent neural networks. Proceedings of SemEval, pages 1197–1201.

[Fyodorov et al., 2000] Fyodorov, Y., Winter, Y., and Francez, N. (2000). A natural logic inference system. In Proceedings of the 2nd Workshop on Inference in Computational Semantics (ICoS-2).

[Gers et al., 2001] Gers, F. A., Eck, D., and Schmidhuber, J. (2001). Applying lstm to time series predictable through time-window approaches. In Artificial Neural NetworksICANN 2001, pages 669–676. Springer.

[Gildea and Jurafsky, 2002] Gildea, D. and Jurafsky, D. (2002). Automatic labeling of semantic roles. Computational Linguistics, 28(3):245–288.

[Glorot et al., 2011] Glorot, X., Bordes, A., and Bengio, Y. (2011). Domain adaptation for large-scale sentiment classification: A deep learning approach. In Proceedings of the 28th international conference on machine learning (ICML-11), pages 513–520.

[Goodman et al., 2016] Goodman, J., Vlachos, A., and Naradowsky, J. (2016). Ucl+ sheffield at semeval-2016 task 8: Imitation learning for amr parsing with an α-bound. Proceedings of SemEval, pages 1167–1172.

[Graves et al., 2013a] Graves, A., Mohamed, A., and Hinton, G. E. (2013a). Speech recognition with deep recurrent neural networks. CoRR, abs/1303.5778.

[Graves et al., 2013b] Graves, A., Mohamed, A.-r., and Hinton, G. (2013b). Speech recognition with deep recurrent neural networks. In Acoustics, speech and signal processing (icassp), 2013 ieee international conference on, pages 6645–6649. IEEE.

[Hajič et al., 2009] Hajič, J., Ciaramita, M., Johansson, R., Kawahara, D., Martí, M. A., Màrquez, L., Meyers, A., Nivre, J., Padó, S., Štěpánek, J., et al. (2009). The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task, pages 1–18. Association for Computational Linguistics.

[Hannun et al., 2014] Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., et al. (2014). Deep speech: Scaling up end-to-end speech recognition. arXiv preprint arXiv:1412.5567.

[Hinton et al., 2006] Hinton, G., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. Neural computation, 18(7):1527–1554.

[Hinton et al., 2012] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. CoRR, abs/1207.0580.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8):1735–1780.

[Ioffe and Szegedy, 2015a] Ioffe, S. and Szegedy, C. (2015a). Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR, abs/1502.03167.

[Ioffe and Szegedy, 2015b] Ioffe, S. and Szegedy, C. (2015b). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International Conference on Machine Learning, pages 448–456.

[Kai and Grishman, 2015] Kai, X. L. T. H. N. and Grishman, C. R. (2015). Improving event detection with abstract meaning representation. ACL-IJCNLP 2015, page 11.

[Kalchbrenner et al., 2015] Kalchbrenner, N., Danihelka, I., and Graves, A. (2015). Grid long short-term memory. arXiv preprint arXiv:1507.01526.

[Kingsbury and Palmer, 2002] Kingsbury, P. and Palmer, M. (2002). From treebank to propbank. In LREC. Citeseer.

[Kiros et al., 2015] Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Skip-thought vectors. In Advances in neural information processing systems, pages 3294–3302.

[Kumar et al., 2015] Kumar, A., Irsoy, O., Su, J., Bradbury, J., English, R., Pierce, B., Ondruska, P., Gulrajani, I., and Socher, R. (2015). Ask me anything: Dynamic memory networks for natural language processing. arXiv preprint arXiv:1506.07285.

[Kurzweil, 2012] Kurzweil, R. (2012). How to create a mind: The secret of human thought revealed. Penguin.

[Landauer et al., 1997] Landauer, T. K., Laham, D., Rehder, B., and Schreiner, M. E. (1997). How well can passage meaning be derived without using word order? a comparison of latent semantic analysis and humans. In Proceedings of the 19th annual meeting of the Cognitive Science Society, pages 412–417.

[Liu et al., 2015] Liu, F., Flanigan, J., Thomson, S., Sadeh, N., and Smith, N. A. (2015). Toward abstractive summarization using semantic representations.

[Liu et al., 2016] Liu, Y., Sun, C., Lin, L., and Wang, X. (2016). Learning natural language inference using bidirectional LSTM model and inner-attention. CoRR, abs/1605.09090.

[Loper et al., 2007] Loper, E., Yi, S., and Palmer, M. (2007). Semlink 1.1.

[MacCartney and Manning, 2009] MacCartney, B. and Manning, C. D. (2009). An extended model of natural logic. In Proceedings of the eighth international conference on computational semantics, pages 140–156. Association for Computational Linguistics.

[Marelli et al., 2014] Marelli, M., Bentivogli, L., Baroni, M., Bernardi, R., Menini, S., and Zamparelli, R. (2014). Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In SemEval@ COLING, pages 1–8.

[May, 2016] May, J. (2016). Semeval-2016 task 8: Meaning representation parsing. Proceedings of SemEval, pages 1063–1073.

[Merity, 2016] Merity, S. (2016). Keras snli baseline example. https://github.com/Smerity/keras_snli.

[Mikolov, 2012] Mikolov, T. (2012). Statistical language models based on neural networks. Presentation at Google, Mountain View, 2nd April.

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

[Mikolov et al., 2013b] Mikolov, T., Yih, W.-t., and Zweig, G. (2013b). Linguistic regularities in continuous space word representations. In Hlt-naacl, volume 13, pages 746–751.

[Miller et al., 1985] Miller, G. A. et al. (1985). Wordnet: a dictionary browser. Information in Data, pages 25–28.

[Mitra and Baral, 2015] Mitra, A. and Baral, C. (2015). Addressing a question answering challenge by combining statistical methods with inductive rule learning and reasoning.

[Montague, 1973] Montague, R. (1973). The proper treatment of quantification in ordinary english. In Philosophy, language, and artificial intelligence, pages 141–162. Springer.

[Moschitti et al., 2003] Moschitti, A., Morarescu, P., and Harabagiu, S. M. (2003). Open domain information extraction via automatic semantic labeling. In FLAIRS conference, pages 397–401.

[Mou et al., 2015] Mou, L., Men, R., Li, G., Xu, Y., Zhang, L., Yan, R., and Jin, Z. (2015). Recognizing entailment and contradiction by tree-based convolution. CoRR, abs/1512.08422.

[Mou et al., 2016] Mou, L., Meng, Z., Yan, R., Li, G., Xu, Y., Zhang, L., and Jin, Z. (2016). How transferable are neural networks in nlp applications? arXiv preprint arXiv:1603.06111.

[Munkhdalai and Yu, 2016a] Munkhdalai, T. and Yu, H. (2016a). Neural semantic encoders. CoRR, abs/1607.04315.

[Munkhdalai and Yu, 2016b] Munkhdalai, T. and Yu, H. (2016b). Neural tree indexers for text understanding. CoRR, abs/1607.04492.

[Nadeau and Sekine, 2007] Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. Lingvisticae Investigationes, 30(1):3–26.

[Nguyen et al., 2017] Nguyen, D. Q., Dras, M., and Johnson, M. (2017). A novel neural network model for joint pos tagging and graph-based dependency parsing. arXiv preprint arXiv:1705.05952.

[Osgood and Suci, 1957] Osgood, S. and Suci, G. J. (1957). The measurement of meaning. University of Illinois Press.

[Palmer, 2009] Palmer, M. (2009). Semlink: Linking propbank, verbnet and framenet. In Proceedings of the generative lexicon conference, pages 9–15. Pisa Italy.

[Palmer et al., 2005a] Palmer, M., Gildea, D., and Kingsbury, P. (2005a). The proposition bank: An annotated corpus of semantic roles. Computational linguistics, 31(1):71–106.

[Palmer et al., 2005b] Palmer, M., Kingsbury, P., and Gildea, D. (2005b). The proposition bank: An annotated corpus of semantic roles. Computational Linguistics, 31(1):71–106.

[Pei et al.,] Pei, W., Ge, T., and Chang, B. An effective neural network model for graph-based dependency parsing.

[Peng et al., 2015] Peng, X., Song, L., and Gildea, D. (2015). A synchronous hyperedge replacement grammar based approach for amr parsing. CoNLL 2015, page 32.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014), 12.

[Persson et al., 2009] Persson, J., Johansson, R., and Nugues, P. (2009). Text categorization using predicate–argument structures.

[Pourdamghani et al., 2014] Pourdamghani, N., Gao, Y., Hermjakob, U., and Knight, K. (2014). Aligning english strings with abstract meaning representation graphs. In EMNLP, pages 425–429.

[Pust et al., 2015] Pust, M., Hermjakob, U., Knight, K., Marcu, D., and May, J. (2015). Parsing english into abstract meaning representation using syntax-based machine translation. Training, 10:218–021.

[Rabiner, 1989] Rabiner, L. (1989). A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2):257–286.

[Ratinov et al., 2011] Ratinov, L., Roth, D., Downey, D., and Anderson, M. (2011). Local and global algorithms for disambiguation to wikipedia. In ACL.

[Rosenblatt, 1962] Rosenblatt, F. (1962). Principles of neurodynamics. Spartan Book.

[Rumelhart et al., 1988] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Learning representations by back-propagating errors. Cognitive modeling, 5(3):1.

[Schapire and Singer, 1998] Schapire, W. W. C. R. E. and Singer, Y. (1998). Learning to order things. In Advances in Neural Information Processing Systems 10: Proceedings of the 1997 Conference, volume 10, page 451. MIT Press.

[Schmidhuber et al., 2002] Schmidhuber, J., Gers, F., and Eck, D. (2002). Learning nonregular languages: A comparison of simple recurrent networks and lstm. Neural Computation, 14(9):2039–2041.

[Schuler, 2005] Schuler, K. K. (2005). Verbnet: A broad-coverage, comprehensive verb lexicon.

[Schwenk, 2007] Schwenk, H. (2007). Continuous space language models. Computer Speech & Language, 21(3):492–518.

[Shen and Lapata, 2007] Shen, D. and Lapata, M. (2007). Using semantic roles to improve question answering. In Emnlp-conll, pages 12–21.

[Smith et al., 2013] Smith, J. R., Saint-Amand, H., Plamada, M., Koehn, P., Callison-Burch, C., and Lopez, A. (2013). Dirt cheap web-scale parallel text from the common crawl. In ACL (1), pages 1374–1383.

[Socher et al., 2011a] Socher, R., Huang, E. H., Pennin, J., Manning, C. D., and Ng, A. Y. (2011a). Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In Advances in Neural Information Processing Systems, pages 801–809.

[Socher et al., 2011b] Socher, R., Lin, C. C., Manning, C., and Ng, A. Y. (2011b). Parsing natural scenes and natural language with recursive neural networks. In Proceedings of the 28th international conference on machine learning (ICML-11), pages 129–136.

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

[Surdeanu et al., 2003] Surdeanu, M., Harabagiu, S., Williams, J., and Aarseth, P. (2003). Using predicate-argument structures for information extraction. In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1, pages 8–15. Association for Computational Linguistics.

[Tan et al., 2015] Tan, M., Xiang, B., and Zhou, B. (2015). Lstm-based deep learning models for non-factoid answer selection. arXiv preprint arXiv:1511.04108.

[Tieleman and Hinton, 2012] Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning, 4(2):26–31.

[Turney, 2013] Turney, P. D. (2013). Distributional semantics beyond words: Supervised learning of analogy and paraphrase. arXiv preprint arXiv:1310.5042.

[Vanderwende et al., 2015] Vanderwende, L., Menezes, A., and Quirk, C. (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.

[Vendrov et al., 2015] Vendrov, I., Kiros, R., Fidler, S., and Urtasun, R. (2015). Order-embeddings of images and language. CoRR, abs/1511.06361.

[Verhagen et al., 2010] Verhagen, M., Sauri, R., Caselli, T., and Pustejovsky, J. (2010). Semeval-2010 task 13: Tempeval-2. In Proceedings of the 5th international workshop on semantic evaluation, pages 57–62. Association for Computational Linguistics.

[Waibel et al., 1989] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. Acoustics, Speech and Signal Processing, IEEE Transactions on, 37(3):328–339.

[Wang et al., 2016] Wang, C., Pradhan, S., Xue, N., Pan, X., and Ji, H. (2016). Camr at semeval-2016 task 8: An extended transition-based amr parser. Proceedings of SemEval, pages 1173–1178.

[Wang et al., 2015] Wang, C., Xue, N., Pradhan, S., and Pradhan, S. (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.

[Werling et al., 2015] Werling, K., Angeli, G., and Manning, C. (2015). Robust subgraph generation improves abstract meaning representation parsing. arXiv preprint arXiv:1506.03139.

[Weston et al., 2011] Weston, J., Bengio, S., and Usunier, N. (2011). Wsabie: Scaling up to large vocabulary image annotation. In IJCAI, volume 11, pages 2764–2770.

[Weston et al., 2015] Weston, J., Bordes, A., Chopra, S., and Mikolov, T. (2015). Towards ai-complete question answering: A set of prerequisite toy tasks. arXiv preprint arXiv:1502.05698.

[Weston et al., 2014] Weston, J., Chopra, S., and Bordes, A. (2014). Memory networks. arXiv preprint arXiv:1410.3916.

[Weston et al., 2012] Weston, J., Ratle, F., Mobahi, H., and Collobert, R. (2012). Deep learning via semi-supervised embedding. In Neural Networks: Tricks of the Trade, pages 639–655. Springer.

[Williams et al., 2017] Williams, A., Nangia, N., and Bowman, S. R. (2017). A broad-coverage challenge corpus for sentence understanding through inference. arXiv preprint arXiv:1704.05426.

[Woodsend and Lapata, 2014] Woodsend, K. and Lapata, M. (2014). Text rewriting improves semantic role labeling. Journal of Artificial Intelligence Research, pages 133–164.

[Zhou and Xu, 2015] Zhou, J. and Xu, W. (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.

7.0.1 Frequently used Acronyms

CBOW	Continuous Bag of Words. When applied to a model, this is also called the sum of word embeddings model.
AMR	Abstract Meaning Representation (Chapter 2.4)
DAMR	Distributed Abstract Meaning Representation (Chapter 5.2)
LSTM	Long Short Term Memory, a type of recurrent neural network, capable of learning long-term dependencies.
NLI	Natural Language Inference [Bowman et al., 2015].
SNLI	Stanford Natural Language Inference Corpus [Bowman et al., 2015].