Aligning English Sentences with Abstract Meaning Representation Graphs using Inductive Logic Programming

by

Shubham Agarwal

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved July 2017 by the
Graduate Supervisory Committee:

Chitta Baral, Chair
Yezhou Yang
Baoxin Li

ARIZONA STATE UNIVERSITY

August 2017

ABSTRACT

In this thesis, I propose a new technique of Aligning English sentence words with its Semantic Representation using Inductive Logic Programming(ILP). My work focusses on Abstract Meaning Representation(AMR). AMR is a semantic formalism to English natural language. It encodes meaning of a sentence in a rooted graph. This representation has gained attention for its simplicity and expressive power.

An AMR Aligner aligns words in a sentence to nodes(concepts) in its AMR graph. As AMR annotation has no explicit alignment with words in English sentence, automatic alignment becomes a requirement for training AMR parsers. The aligner in this work comprises of two components. First, rules are learnt using ILP that invoke AMR concepts from sentence-AMR graph pairs in the training data. Second, the learnt rules are then used to align English sentences with AMR graphs. The technique is evaluated on publicly available test dataset and the results are comparable with state-of-the-art aligner.

To Mother, Father and Brother

ACKNOWLEDGMENTS

I would like to thank many people for their support and guidance which made this possible. First of all, thanks to my advisor, Dr. Chitta Baral, who guided me at every step in the project with his experience and helped me in making sense out of numerous confusing scenarios. Thanks to my committee members, Dr. Baoxin Li and Dr. Yezhou Yang, who offered me guidance and support whenever I needed. Thanks to the Arizona State University for providing me an opportunity to work with such great minds and technology. Thanks to my colleagues in Dr. Baral's lab, especially Arindam Mitra and Arpit Sharma who encouraged me and corrected me at times with their knowledge of subject matter and experience. Without them this project would not have been possible. And finally, thanks to my parents, and numerous friends who endured this long process with me, always offering support and love.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Natural language facilitates the exchange of thoughts and ideas among people. These ideas are essence of the natural language sentences, also called semantics. In other words, meaning of a text is called its semantics. To fully address natural language semantics, it would require a complete theory of how people think and communicate their ideas. In one of the recent works in this direction, Banarescu *et al.* (2013) released a large Abstract Meaning Representation(AMR) corpus that captures the logical meaning of sentences. AMR is a single rooted, directed acyclic graph that incorporates semantic roles, coreference, modality, questions, negation, and many other linguistic attributes. Nodes of the graph are *Concepts* and edges are *Roles*. Figure 1 shows an example AMR graph for the sentence *The government can override the market.*

However, AMR is not annotated with alignment links between English words and AMR concepts. Such alignments are necessary in the construction of semantic parsers for AMR. Semantic parsing refers to the task of transforming natural language sentence into a formal representation of its meaning. To train a semantic parser, it is important to know which spans of words invoke which concepts in the corresponding graph, i.e. alignment links between each English token and its AMR representation are needed.

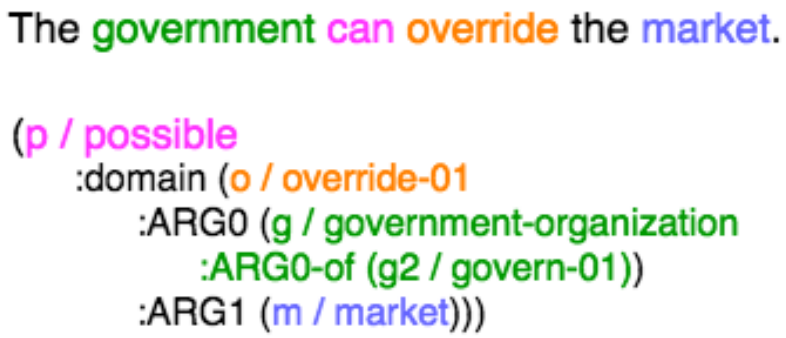

Figure 1. Alignment between sentence and its AMR graph. Each color is an alignment excluding black.

1.1 Sentence - AMR Alignment

Alignment between an English sentence and its AMR graph refers to the correspondances between individual word tokens and nodes in the graph. These alignments are not trivial as many AMR concepts are invoked by words that have no lexical similarity to the concept. Figure 1 shows an aligned example. Note that the concept *possible* is invoked by the word *can* depicting the non-trivial nature of AMR alignment.

Existing works for this problem statement propose methods based on manual rules and string-to-string Machine Translation. Section 1.2 discusses related works and Section 1.3 gives an overview of my approach.

1.2 Related Works

Flanigan *et al.* (2014) is the first work in English-AMR alignment. They propose a rule-based method that aligns spans of English tokens to an AMR graph concept fragment by manually writing a set of rules to be executed in a specific order.

They assume that AMR roles are associated to the aligned concepts so they do not explicitly align roles. They evaluate their method on a hand aligned dataset of 200 sentences. The limitation of the rule-based method is that it cannot benefit from more data annotation of AMR. Also if new concepts are introduced by AMR later, new set of rules will have to be defined which might possibly conflict with existing rules.

A Statistical Machine Translation(SMT) based string-to-string alignment method has been proposed by Pourdamghani *et al.* (2014). They align both concepts and roles to English tokens. They linearize the AMR graph and transform the original string-to-graph problem into string-to-string. They use unsupervised alignment models(IBM models, Brown *et al.* (1993)). Machine Translation generally requires a large amount of training data, hence this method severely suffers from data sparceness issues due to small amount of AMR training data. They report accuracy of their system on a different hand aligned dataset.

Chu and Kurohashi (2016) work on the base method of Pourdamghani *et al.* (2014) and propose a supervised syntax-based alignment model. They first convert AMRs to Constituency Trees and then perform Hierarchical Alignment on these Constituency Trees.

In AMR parsing, Flanigan *et al.* (2014) is the first work. They proposed a graph based parsing method that first generated concepts given a sentence. The concepts come from their own aligner. Then they form the graph from those concept fragments using relation(role) prediction. Their algorithm finds a maximum spanning and connected graph. Their parser is named JAMR and is publicly available. Werling

et al. (2015) extended the work of JAMR by proposing generative actions for subgraph derivation based on their alignment criteria. Wang *et al.* (2015) proposed a transition based method that first parses English sentence to dependency tree using a dependency parser and then transforms the dependency tree to AMR graph by learning the series of transitions requires to reach AMR graph as destination from the source dependency tree. They also use the JAMR aligner for feature extraction in their parser. Their parser is publicly available as CAMR parser. Artzi *et al.* (2015) proposed using the combinatory categorial grammar(CCG) for AMR parsing. Pust *et al.* (2015) used string to tree syntax based Machine Translation method. After transforming English sentences to trees, they further convert trees to AMR graphs. Their parser is publicly available as ISI AMR parser.

1.3 An Overview Of The Approach

As previously mentioned, the approach is based on learning alignment rules instead of manually defining them. This includes **three** main components: *Natural Language and AMR to ILP encoder* to encode the preprocessed data in a meaningful representation which can be fed to ILP engine for learning, *ILP engine* where rules are learnt from the training data for each category of AMR and *Aligner* to align sentences using learnt rules. Figure 2 shows the work-flow diagram.

The idea behind this approach is that predicting concepts invoked by words in a sentence is same as aligning words to those concepts. So effectively, learning rules to invoke these concepts becomes the primary target. After carefully studying the AMR guidelines, concepts can be categorized into following: *concepts as word tokens*, *modal concepts*, *imperative concept*, *negation concept*, *concepts as categorial variations*

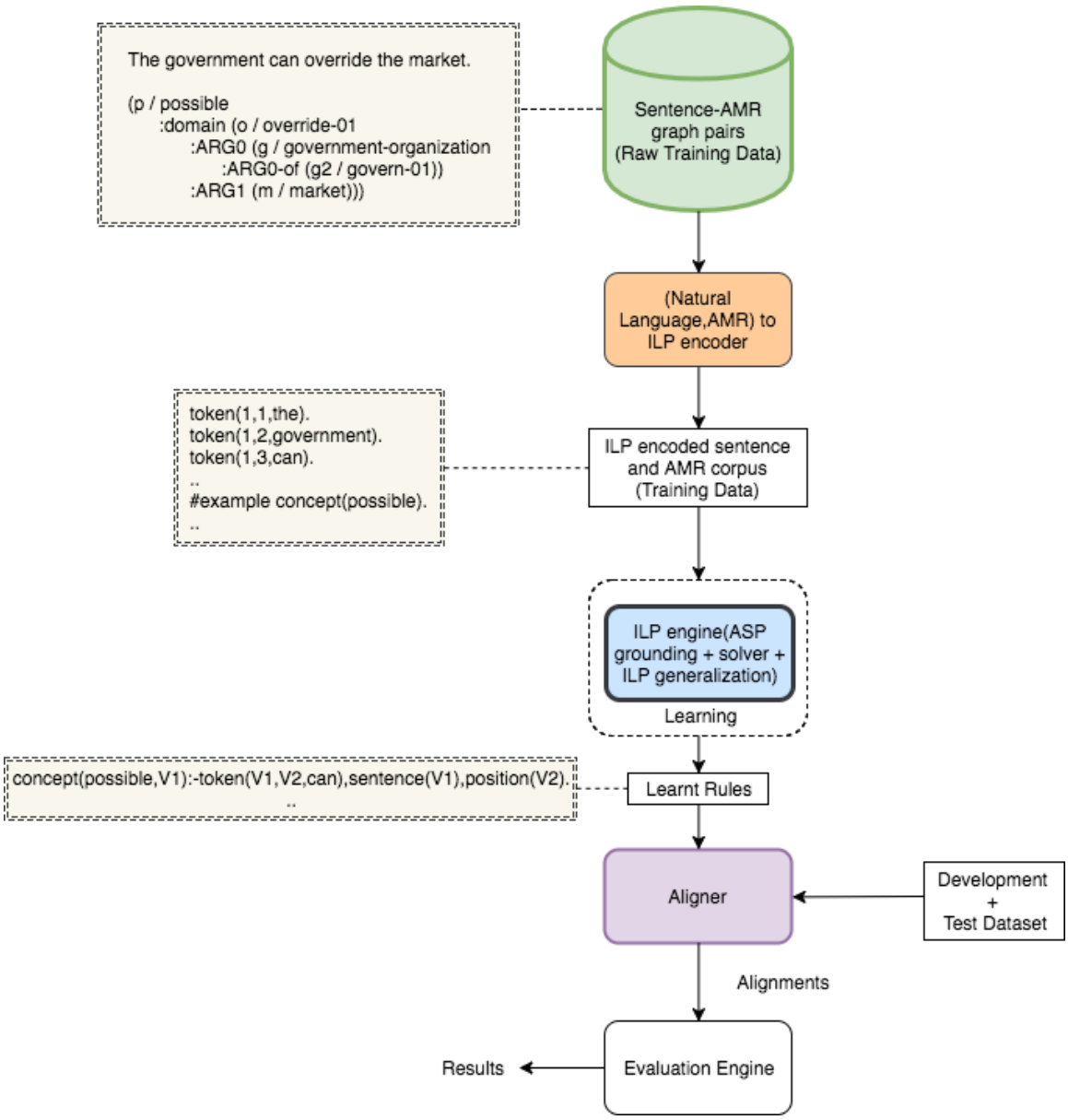

Figure 2. Work-Flow diagram of AMR aligner using ILP.

of word tokens, *concepts negated with prefix and suffix*, *question concepts*, *abstract concepts*, *causal concept*. The categories are defined as follows.

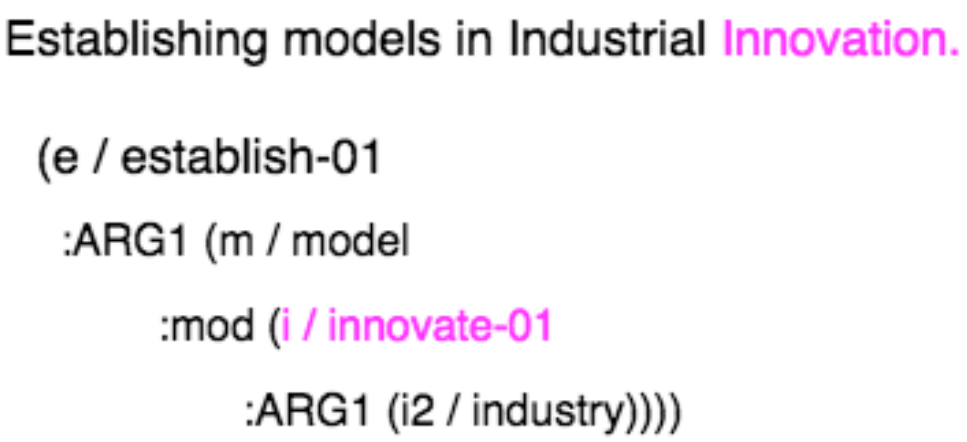

Figure 3. Example showing alignment of innovate-01 concept

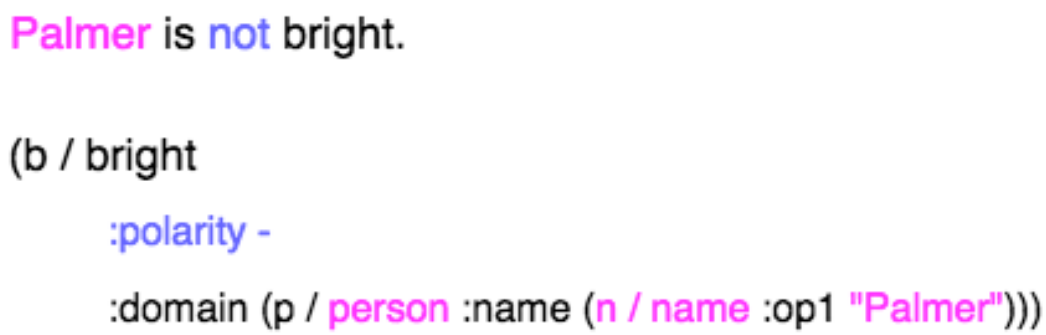

Figure 4. Example showing alignments of :polarity - concept and 'person' abstract concept

- **Concepts as word tokens:**

  In this category, AMR concepts are exactly the same as word tokens in the sentence. For example:

  Sentence: The government can override the market. (Figure 1)

  Concept in focus: market

  Aligning token in sentence: market

- **Modal concepts**

In this category, modal concepts like possible, obligate etc. are invoked from modal tokens in the sentence.

Sentence: The government can override the market. (Figure 1)

Concept in focus: possible

Aligning token in sentence: can

- **Imperative concepts**

  In this category, *imperative* concept is invoked with an exclamation mark token in the sentence.

  Sentence: Go China!

  Concept in focus: imperative

  Aligning token in sentence: !

- **Negation concept**

  In this category, modal concept :polarity - is invoked from negation words like no, not etc. in the sentence.

  Sentence: Palmer is not bright. (Figure 4)

  Concept in focus: :polarity -

  Aligning token in sentence: not

- **Concepts as Categorial variations of word tokens**

  In this category, categorial variations of word tokens in sentence are aligned with concepts in AMR which are defined as root forms. For example, the word innovation(noun) aligns to the concept innovate(verb)

  Sentence: Establishing Models in Industrial Innovation (Figure 3)

Concept in focus: :innovate-01

Aligning token in sentence: Innovation

- **Concepts negated with prefix and suffix**

  In this category, negation is inherent to the word itself. For example, word 'illegal' is equivalent to 'not legal'.

  Sentence: This is illegal.

  Concept in focus: : l / legal :polarity -

  Aligning token in sentence: illegal

- **Question concepts**

  In this category, question words like who, what, where etc. are aligned to amr-unknown concept.

  Sentence: Where is John?

  Concept in focus: : amr-unknown

  Aligning token in sentence: Where

- **Abstract concepts**

  In this category, concepts like person, organization, thing etc. that do not directly match with any word token are aligned.

  Sentence: Palmer is not bright. (Figure 4)

  Concept in focus: : person

  Aligning token in sentence: Palmer

- **Causal concept**

  In this category, concept *cause* is invoked if there is a causal relation between two events in the sentence.

  Sentence: The government can override the market because it is powerful.

  Concept in focus: cause

  Aligning token in sentence: because

For each of these categories, rules are learnt using ILP on the training dataset. The learnt rules are then put together to align sentences with their AMR. Note that information related to each category is provided to split the data into each category. Also when learning rules for each category, concepts related to only that category are considered while the rest are ignored. In the following Chapters each component of this approach is discussed and explained in detail with appropriate examples.

Chapter 2

LINGUISTIC BACKGROUND KNOWLEDGE EXTRACTION AND ENCODING

This chapter details the prerequisites of rule learning process. For learning, appropriate background knowledge needs to be extracted. Each data instance will be represented using this background knowledge in a logical form. Subsection 2.1 talks about *Background Knowledge* and Subsection 2.2 talks about the *Answer Set Programming* logical form.

2.1 Background Knowledge

Learning alignment rules from sentence-AMR input pairs requires extraction of linguistic background knowledge from sentence. The input pairs are represented using this background knowledge. Extracted background knowledge is according the categorization of AMR as listed in section 1.3. Using the definition of each category provided by AMR guidelines, following is the list of background knowledge used:

- **Lemma**

  Lemma is the outcome of lemmatization which refers to removing inflectional endings and to return the base form of a word using a vocabulary and morphological analysis of words. Stanford CoreNLP(Manning *et al.* (2014)) is used to perform lemmatization.

  For example: Lemma of word *saw* attempts to return either *see* or *saw*

depending on whether the use of the word was as a verb or a noun.

- **Part of Speech**

  Part of Speech(POS) tags like *verb, noun, pronoun, modal, adverb, wh-determiner, wh-pronoun, etc.* from Penn Treebank part-of-speech tag set are used as background knowledge. Stanford POS tagger(Toutanova *et al.* (2003)) is used to tag the sentences with POS.

- **Modal**

  Modals like *can, could, may, might, shall, should, will, would* invoke modal concepts. Sentences having modals have this as one of the distinctive background knowledge. Again Stanford POS tagger is used to find modals in sentences recognized by 'MD' tag.

- **Named Entity**

  Stanford's Named Entity Recognizer(NER)(Finkel *et al.* (2005)) is used to find named entities like *person, location, organization* etc. in sentences.

  For example: In the sentence, *My name is Shubham.*, Stanford NER marks 'Shubham' as a person.

- **Question tokens**

  Again the POS tagger was used to find out question words like *who, whom, where, how, which etc.*. Their POS tags are one of 'WP, WP$, WRB'.

- **Categorial Variation**

Many concept in AMR are categorial variations of word tokens in sentences. For example, 'innovate' aligns to 'innovation' and lemmatizer will not return innovate as the lemma of innovation because former is a verb while latter is a noun. To align such concepts, categorial variation of words is required as background knowledge. CATVAR 2.0(Habash *et al.*, 2003) is used for it. It is a database of clusters of uninflected words (lexemes) and their categorial (i.e. part-of-speech) variants. For example, the words hunger(V), hunger(N), hungry(AJ) and hungriness(N) are different English variants of some underlying concept describing the state of being *hungry*. Another example is the *developing* cluster:(develop(V), developer(N), developed(AJ), developing(N), developing(AJ), development(N)).

2.2 Answer Set Programming Encoding

Now, since we have the background knowledge, its important to represent input data using it, in a formal way that can be read by Inductive Logic Programming engine. I am using the XHAIL open source software[1] for rule learning using ILP. It takes input in Answer Set Programming(ASP) format. ASP is a form of declarative programming oriented towards difficult search problems. It is particularly useful in knowledge-intensive applications. ASP is based on the stable model (answer set) semantics of logic programming (Gelfond and Lifschitz (1988)), which applies ideas of autoepistemic logic (Moore (1984)) and default logic (Reiter (1980)) to the analysis of negation as failure.

It is a collection of rules of the form,

[1]https://github.com/stefano-bragaglia/XHAIL

$L_0 \leftarrow L_1, ..., L_m, \mathbf{not} L_{m+1}, ..., \mathbf{not} L_n$

where L_i's are literals as in classical logic. Intuitively, the above rule means that if $L_1, ..., L_m$ are true and if $L_{m+1}, ..., L_n$ can be safely assumed to be false then L_0 must be true (Baral (2003)) . The left-hand side of an ASP rule is called the head and the right-hand side is called the body. The semantics of ASP is based on the stable model (answer set) semantics of logic programming (Gelfond and Lifschitz (1988)).

An example representation is as follows:

Sentence : *Even the government cannot override the market!*

%% Background
position(I) :- token(S,I,L).
sentence(S) :- token(S,I,L).
lemmaList(L) :- token(S,I,L).

modalConcepts(possible;likely;obligate;permit;recommend;prefer).

token(28,1,even).
token(28,2,the).
token(28,3,government).
token(28,4,can).
token(28,5,nt).
token(28,6,override).
token(28,7,the).
token(28,8,market).

modal(can).

%% Examples
#example concept(possible,28).

In the above example, sentence is represented as numbered lemma tokens(28 is sentence number and 1-8 are index/position of tokens in the sentence) and the background knowledge *modal(can). Background* is a general definition of representation.

Here it defines what a positon and lemma is in an arity 3 token of sentence. *Modal Concepts* is a list of modal concepts AMR uses as by its definition. *Example* defines the target concept annotated in this sentence by AMR. Here *possible* is a concept in the corresponding AMR of this sentence.

Chapter 3

RULE LEARNING USING INDUCTIVE LOGIC PROGRAMMING

3.1 Inductive Logic Programming

Inductive Logic Programming (ILP) (Muggleton (1991)) is a subfield of Machine learning that is focused on learning logic programs. Given some background knowledge, a set of positive examples ξ^+, negative examples ξ^-, an ILP algorithm finds a Hypothesis H (answer set program) such that $BUH \models \xi^+$ and $BUH \not\models \xi^-$.

A language bias restricts the possible hypothesis space by a series of Mode declarations M (Muggleton (1991)). A *modeh(s)* declaration denotes a literal s that can appear as the head of a rule (Table 3). A *modeb(s)* declaration denote a literal s that can appear in the body of a rule (Table 3). The argument s is called *schema* and consists of two parts:

1) an identifier for the literal
2) a list of placemakers for each argument of that literal.

A placemaker is either +type (input), -type (output) or $type (constant), where *type* denotes the type of the argument. An answer set rule is in the hypothesis space defined by L (call it L(M)) iff its head (resp. each of its body literals) is constructed from the schema s in a modeh(s) (resp. in a modeb(s)) in L(M)) as follows:

- By replacing an output (-) placemaker by a new variable.

- By replacing an input (+) placemaker by a variable that appears in the head or in a previous body literal.

- By replacing a ground ($) placemaker by a ground term.

As mentioned in Mitra and Baral (2016), note that the set of negative examples ξ^- is required to restrain H from being over generalized. Informally, given an ILP task, an ILP algorithm finds a hypothesis H that is general enough to cover all the examples in ξ^+ and also specific enough to not cover any example in ξ^-. Without ξ^-, the learned H will contain only facts. In this thesis, negative examples are automatically generated from positive examples by assuming the answers are complete, i.e. if a sentence-AMR pair says that *possible* is the only modal concept in it, then its assumed that other modal concepts are negative examples for this instance.

I use the XHAIL system for rule learning. Next section discusses the details of XHAIL.

3.2 The XHAIL System

As mentioned in Katzouris *et al.* (2015), XHAIL is an abductive-inductive system that constructs hypotheses in a three-phase process. Given an ILP task *ILP(B, E, M)*, the first two phases return a ground program K, called *Kernel Set of* E[2], such

[2]This kernel is the learnt model using Inductive Logic Programming. It is different from a typical Kernel in Machine Learning which generally helps to do certain calculations faster that otherwise would involve computations in higher dimensional space.

that $B\ UK \vDash E$. The first phase generates the head's of K's clauses by abductively deriving from B a set Δ of instances of head atoms, as defined by the language bias, such that $BU\Delta \vDash E$. The second phase generates K, by saturating each previously abduced atom with instances of body atoms that deductively follow from $B\ U\Delta$. The language bias used by XHAIL is *mode declarations* as mentioned above.

By construction, the Kernel Set covers the provided examples. In order to find a good hypothesis, XHAIL thus searches in the space of theories that subsume the Kernel Set. To this end, the latter is variabilized, i.e. each term that corresponds to a variable, according to the language bias, is replaced by an actual variable. The variabilized Kernel Set K_v is subject to a syntactic transformation of its clauses.

For each clause $C_i\ \epsilon\ K_v$ and each body literal $\delta_i^j\ \epsilon\ C_i$, a new atom $v(\delta_i^j)$ is generated, as a special term that contains the variables that appear in δ_i^j. The new atom is wrapped inside an atom of the form $try(i, j, v(\delta_i^j))$. An extra atom $use(i, 0)$ is added to the body of C_i and two new clauses $try(i, j, v(\delta_i^j)) \leftarrow use(i, j), \delta_i^j$ and $try(i, j, v(\delta_i^j)) \leftarrow \textbf{not}\ use(i, j)$ are generated, for each body literal $\delta_i^j\ \epsilon\ C_i$.

These clauses are all put together into a program U_{K_v} . From U_{K_v}, literals and clauses may be selected in order to construct a hypothesis that accounts for the examples. As explained in Ray (2009), the intuition is as follows: In order for the head atom of clause $C_i\ \epsilon\ U_{K_v}$ to contribute towards the coverage of an example, each of its $try(i, j, v(\delta_i^j))$ atoms must succeed. By means of the two rules added for each such atom, this can be achieved in two ways: Either by assuming *not* $use(i, j)$, or by satisfying δ_i^j and abducing $use(i, j)$. A hypothesis clause is constructed by the

head atom of the i-th clause C_i of K_v, if $use(i, 0)$ is abduced, and the j-th body literal of C_i , for each abduced $use(i, j)$ atom. All other clauses and literals from K_v are discarded. Search is biased by minimality, i.e. preference towards hypotheses with fewer literals. This is realized by means of abducing a minimal set of *use/2* atoms.

3.3 Learning

In this section, I illustrate the formulation of an ILP task for alignment rule learning and the way the answer set programs are learned. I explain the approach with the XHAIL (Ray (2009)) algorithm. Rules are learnt for each category of AMR as described in *Introduction*.

Given an ILP task $ILP(B, \xi = \xi^+ \ U \ \xi^-, M)$, XHAIL derives the hypothesis in a three step process:

- **Grounding**
- **Finding kernel**
- **Hypothesis generation/Generalization**

These three steps remain the same for each category rule learning, just the content involved in each step changes. Lets see them one by one.

3.3.1 Modal Concepts

In this category, modal concepts like possible, obligate, recommend etc. are aligned to modals in the sentence. So rules stating which modal concepts are invoked by which particular modals are to be learnt. For example in Figure 1, *can* aligns to *possible* concept. Note here that while learning rules for this category, only concepts corresponding to this category are considered while the others are ignored. So information is being provided on what concepts the learning target has but the ILP system learns which of these concepts align to which words in the sentence. Lets discuss the three steps of learning in this case.

Following is ASP encoding of the two examples to be referred for the three steps in this category.

% sentence: The government can override the market.
%% Background
position(I) :- token(S,I,L).
sentence(S) :- token(S,I,L).
lemmaList(L) :- token(S,I,L).

modalConcepts(possible;likely;obligate;permit;recommend;prefer).

token(0,1,the).
token(0,2,government).
token(0,3,can).

modal(can).

token(0,4,override).

token(0,5,the).

token(0,6,market).

%% Examples

#example concept(possible,0).

#example not concept(likely,0).

#example not concept(obligate,0).

#example not concept(permit,0).

#example not concept(recommend,0).

#example not concept(prefer,0).

% sentence: I like you.

token(1,1,i).

token(1,2,like).

token(1,2,you).

#example not concept(possible,1).

#example not concept(likely,1).

#example not concept(obligate,1).

#example not concept(permit,1).

#example not concept(recommend,1).

#example not concept(prefer,1).

%% Mode(s)

#modeh concept($modalConcepts,+sentence).

#modeb token(+sentence,-position,$modal).

Step 1 : In the first step the XHAIL algorithm finds a set of ground (variable free) atoms $\Delta = U_{i=1}^{n}\alpha_i$ such that $B\ U\ \Delta \models \xi$ where each α_i is a ground instance of modeh(s) declaration atoms. For the ILP problem above, there is one *modeh* declaration. Thus the Δ can contain ground instances of this atom in the *modeh* declaration. Following shows one possible Δ that meets the above requirements for the ILP task:

Δ = *concept(possible,0)*

Step 2 : In the second step, XHAIL computes a clause $\alpha_i \leftarrow \delta_i^1 ... \delta_i^{m_i}$ for each α_i in Δ, where $B\ U\ \Delta \models \delta_i^j,\ \forall 1 \leq i \leq n,\ 1 \leq j \leq m_i$ and each clause $\alpha_i \leftarrow \delta_i^1 ... \delta_i^{m_i}$ is a ground instance of the rule in $L(M)$. In the running example, Δ contains one atom that must lead to a clause k_1. k_1 = *concept(possible,0):-token(0,2,can).* is initialized to the head of clause k_1. The body of k_1 is saturated by adding all possible ground instances of the literals in *modeb(s)* declarations that satisfy the constraints mentioned above. The only ground clause K constructed in this step for the running example and its variabilized version K_v that is obtained by replacing all input and output terms by variables is shown below:

$K = concept(possible, 0) :- token(0, 2, can).$

$K_v = concept(possible, V1) :- token(V1, V2, can).$

Step 3 : In this step XHAIL tries to find a compressive theory H by deleting from K_v as many literals (and clauses) as possible while ensuring that $B\ U\ H\ \models\ \xi$. In the running example, it will lead to $H = K_v$. Following is the expanded learnt rule:

$$H = concept(possible, V1) : -token(V1, V2, can), sentence(V1), position(V2).$$

When put into words, this rule reads: *If 'can' is a token in the sentence V1 at position V2 then the concept 'possible' aligns to this token.*

Following the exact same pattern for all the sentences in the training dataset, Table 1 lists the set of rules learnt.

3.3.2 Negation Concept

The Negation Concept is represented as *:polarity* - in AMR which aligns to negation words like *no, not.* For example,

% sentence: We can not say for certain.
%% Background
position(I) :- token(S,I,L).
sentence(S) :- token(S,I,L).
lemmaList(L) :- token(S,I,L).

modalConcepts(possible;likely;obligate;permit;recommend;prefer).

concept(possible,V1):-token(V1,V2,can),sentence(V1),position(V2).
concept(recommend,V1):-token(V1,V2,can),sentence(V1),position(V2).
concept(obligate,V1):-token(V1,V2,must),sentence(V1),position(V2).
concept(obligate,V1):-token(V1,V2,shall),sentence(V1),position(V2).
concept(likely,V1):-token(V1,V2,should),sentence(V1),position(V2).
concept(recommend,V1):-token(V1,V2,shld),sentence(V1),position(V2).
concept(possible,V1):-token(V1,V2,may),sentence(V1),position(V2).
concept(obligate,V1):-token(V1,V2,may),sentence(V1),position(V2).
concept(possible,V1):-token(V1,V2,will),sentence(V1),position(V2).
concept(obligate,V1):-token(V1,V2,will),sentence(V1),position(V2).
concept(recommend,V1):-token(V1,V2,will),sentence(V1),position(V2).
concept(possible,V1):-token(V1,V2,would),sentence(V1),position(V2).
concept(prefer,V1):-token(V1,V2,would),sentence(V1),position(V2).

Table 1. Rules - Modal concepts

negativeConcept(polarityNeg).

negationTokens(no;not_tok).

token(0,1,we).

token(0,2,can).

modal(can).

token(0,3,not_tok).

token(0,4,say).

token(0,5,for).

token(0,6,certain).

%% Examples

#example concept(possible,0).

#example not concept(likely,0).

#example not concept(obligate,0).

#example not concept(permit,0).

#example not concept(recommend,0).

#example not concept(prefer,0).

#example concept(polarityNeg,0).

% sentence: I like you.

token(1,1,i).

token(1,2,like).

token(1,2,you).

#example not concept(possible,1).

#example not concept(likely,1).

#example not concept(obligate,1).

#example not concept(permit,1).

#example not concept(recommend,1).

#example not concept(prefer,1).

#example not concept(polarityNeg,1).

#modeh concept($negativeConcept,+sentence).

#modeb token(+sentence,-position,-negationTokens).

%% Mode(s)

#modeh concept($modalConcepts,+sentence).

#modeb token(+sentence,-position,$modal).

The bold lines highlight information related to this category. The three steps for this category in learning are as follows:

Step 1 : There are two *modeh* declarations in this example. Thus the Δ can contain ground instances of this atom in the *modeh* declaration. Following shows two possible Δ that meet the above requirements for the ILP task:

$$\Delta = \begin{cases} concept(polarityNeg, 0) \\ concept(possible, 0) \end{cases}$$

Step 2 : In the second step, Kernel is formed using the Δ instances as the head to a clause and its body is saturated by adding all possible ground instances of the literals in *modeb(s)* declarations. The only ground clause K constructed in this step for the running example and its variabilized version K_v that is obtained by replacing all input and output terms by variables is shown below:

$$K = \begin{cases} concept(possible, 0) : -token(0, 2, can), token(0, 3, not_t ok). \\ concept(polarityNeg, 0) : -token(0, 2, can), token(0, 3, not_t ok). \end{cases}$$

concept(polarityNeg,V1):-token(V1,V3,V4),sentence(V1),position(V3), negationTokens(V4).

Table 2. Rule - Negation concept

$$K_v = \begin{cases} concept(possible, V1) : -token(V1, V2, can), token(V1, V3, V4) \\ concept(polarityNeg, V1) : -token(V1, V2, can), token(V1, V3, V4). \end{cases}$$

Step 3 : In this step XHAIL tries to find a compressive theory H by deleting from K_v as many literals (and clauses) as possible while ensuring that $B\ U\ H\ \models\ \xi$. In the running example, it will lead to rule in Table 2. Note that hypothesis relevant to this category is shown only.

3.3.3 Question Concept

The question concept is represented as *amr-unknown* in AMR which aligns to question words in sentence like *how, when, where, what, who, why.* Following is an ASP encoded example,

% sentence: How did they set it up ?
%% Background
position(I) :- token(S,I,L).
sentence(S) :- token(S,I,L).

lemmaList(L) :- token(S,I,L).

questionConcept(amrUnknown).

questionWords(where;when;why;how;what;who;whom).

posTags(wdt;wp;wpdollar;wrb;vbd;prp;vb;rp;dot;vbp).

% this is just a subset of POS tags used for illustration

quesPos(wdt;wp;wpdollar;wrb).

token(0,1,how).

pos(0,1,wrb).

token(0,2,did).

pos(0,2,vbd).

token(0,3,they).

pos(0,3,prp).

token(0,4,set).

pos(0,4,vb).

token(0,5,it).

pos(0,5,prp).

token(0,6,up).

pos(0,6,rp).

token(0,7,questionMark).

pos(0,7,dot).

%% Examples

#example concept(amrUnknown,0).

% sentence: I like you.

token(1,1,i).

pos(1,1,prp).

token(1,2,like).

pos(1,2,vbp).

token(1,2,you).

pos(1,3,prp).

#example not concept(amrUnknown,1).

#modeh concept($questionConcept,+sentence).

#modeb token(+sentence,-position,$questionWords).

#modeb pos(+sentence,-position,$quesPos).

Step 1 : There is one *modeh* declaration in this example. Thus the Δ can contain ground instances of this atom in the *modeh* declaration. Following shows the possible Δ that meets the above requirement for the ILP task:

$$\Delta \ = \ concept(amrUnknown, 0)$$

Step 2 : In the second step, Kernel is formed using the Δ instances as the head to a clause and its body is saturated by adding all possible ground instances of the literals in *modeb(s)* declarations. The only ground clause K constructed in this step for the running example and its variabilized version K_v that is obtained by replacing

all input and output terms by variables is shown below:

$$K \;=\; concept(amrUnknown, 0) : -token(0, 1, how), pos(0, 1, wrb).$$

$$K_v \;=\; concept(amrUnknown, V1) : -token(V1, V2, how), pos(V1, V2, wrb).$$

Step 3 : In this step XHAIL tries to find a compressive theory H by deleting from K_v as many literals (and clauses) as possible while ensuring that $B \; U \; H \; \models \; \xi$. In the running example, it will lead to following rule.

$$H \;=\; concept(amrUnknown, V1) : -token(V1, V2, how), pos(V1, V2, wrb), \\ sentence(V1), position(V2).$$

Following the same pattern, Table 3 lists all the rules learnt for question concept.

3.3.4 Concept As Word Tokens

In this category, word tokens in the sentence are the concept themselves. Following is an ASP encoded example,

% sentence: They can set it .
%% Background
position(I) :- token(S,I,L).
sentence(S) :- token(S,I,L).

concept(amrUnknown,V1) :- token(V1,V2,how), pos(V1,V2,wrb), sentence(V1), position(V2).
concept(amrUnknown,V1) :- token(V1,V2,which), pos(V1,V2,wdt), sentence(V1), position(V2).
concept(amrUnknown,V1) :- token(V1,V2,what), pos(V1,V2,wp), sentence(V1), position(V2).
concept(amrUnknown,V1) :- token(V1,V2,who), pos(V1,V2,wp), sentence(V1), position(V2).
concept(amrUnknown,V1) :- token(V1,V2,whom), pos(V1,V2,wp), sentence(V1), position(V2).
concept(amrUnknown,V1) :- token(V1,V2,whose), pos(V1,V2,wpdollar), sentence(V1), position(V2).
concept(amrUnknown,V1) :- token(V1,V2,where), pos(V1,V2,wrb), sentence(V1), position(V2).
concept(amrUnknown,V1) :- token(V1,V2,when), pos(V1,V2,wrb), sentence(V1), position(V2).
concept(amrUnknown,V1) :- token(V1,V2,why), pos(V1,V2,wrb), sentence(V1), position(V2).

Table 3. Rule - Question concept

lemmaList(L) :- token(S,I,L).

conceptList(they;set;it;placeholder).

modalConcept(possible;likely;obligate;permit;recommend;prefer).

token(0,1,they).

token(0,2,can).

modal(can).

token(0,3,set).

token(0,4,it).

%% Examples

#example concept(they,0).

#example concept(possible,0).

#example concept(set,0).

#example concept(it,0).

#example not concept(likely,0).

#example not concept(obligate,0).

#example not concept(permit,0).

#example not concept(recommend,0).

#example not concept(prefer,0).

#example not concept(placeholder,0).

#modeh concept($modalConcept,+sentence).

#modeb token(+sentence,-position,$modal).

#modeh concept(+conceptList,+sentence).

#modeb token(+sentence,-position,+conceptList).

The bold lines highlight information related to this category. The three steps for this category in learning are as follows:

Step 1 : There are two *modeh* declarations in this example. Thus the Δ can contain ground instances of this atom in the *modeh* declaration. Following shows four

possible Δ that meet the above requirements for the ILP task:

$$\Delta = \begin{cases} \textbf{concept(it,0)} \\ \textbf{concept(they,0)} \\ \textbf{concept(set,0)} \\ concept(possible, 0) \end{cases}$$

Step 2 : In the second step, Kernel is formed using the Δ instances as the head to a clause and its body is saturated by adding all possible ground instances of literals in *modeb(s)* declarations. The only ground clause K constructed in this step for the running example and its variabilized version K_v that is obtained by replacing all input and output terms by variables is shown below:

$$K = \begin{cases} \textbf{concept(it,0):-token(0,2,can),token(0,4,it).} \\ \textbf{concept(they,0):-token(0,2,can),token(0,1,they).} \\ \textbf{concept(set,0):-token(0,2,can),token(0,3,set).} \\ concept(possible, 0) : -token(0, 2, can) \end{cases}$$

$$K_v = \begin{cases} \textbf{concept(V1,V2):-token(V2,V4,V1).} \\ concept(possible, V1) : -token(V1, V2, can) \end{cases}$$

concept(V1,V2) :- token(V2,V4,V1), sentence(V2), position(V4), conceptList(V1).

Table 4. Rule - Word token as concepts

Step 3 : In this step XHAIL tries to find a compressive theory H by deleting from K_v as many literals (and clauses) as possible while ensuring that $B\ U\ H\ \models\ \xi$. In the running example, it will lead to rule in Table 4. Note that hypothesis relevant to this category is shown only.

3.3.5 Concepts As Categorial Variations Of Word Tokens

In this category, categorial variations of word tokens in sentence are aligned with concepts in AMR which are defined as root forms. This category is a similar to a fuzzy matching of words and concepts, except that I am using a categorial variation database of word clusters called CATVAR (Habash and Dorr (2003)). If a token lies in the catvar cluster of a concept, it is aligned to that concept. For example, catvar cluster of concept *innovate* is *[innovate, innovator,innovation,innovative,innovational,innovativeness]*. If sentence has any token that matches these words in this cluster, then that token is aligned to this concept *innovate*. This is somewhat similar to the previous category except that we are matching with words in catvar cluster of concept instead of concept itself.

The rule looks like in Table 5.

concept(V1,V2) :- token(V2,V4,V3), catword(V3,V1), sentence(V2), position(V4), conceptList(V1), lemmaList(V1).

Table 5. Rule - Concepts as Categorial variations of word tokens

concept(polarityNeg,V2) :- token(V2,V4,V3), hasNegativePrefix(V3), appliesCatvarRule(V5), rootWithoutNegativePrefix(V5,V3), sentence(V2), position(V4), lemmaList(V3)
concept(V1,V2) :- token(V2,V4,V3), hasNegativePrefix(V3), appliesCatvarRule(V5), rootWithoutNegativePrefix(V5,V3), resultCatvarRule(V1,V5), sentence(V2), position(V4), lemmaList(V3)

Table 6. Rule - Concepts as Categorial variations of word tokens

3.3.6 Concepts Negated With Prefix And Suffix

In this category, negative prefix or suffix invokes the *:polarity* - concept along with the concept corresponding to root of the word. For example, *illegal* will invoke *:polarity* - concept aligning to prefix *il-* and *legal* aligning to root *legal*. Prefixes and suffix *[a,im,ir,in,un,dis,non,less]* are part of this rule. To align the root of the prefixed word, CATVAR is used as in the Table 5 rule. Learnt rule for this category is in Table 6.

3.3.7 Imperative Concept

In this category, concept *imperative* aligns to token *!* in the sentence. For Example,

Sentence: Go China ***!***

$(g2/go-01 : mode$ **imperative**

concept(imperative,V2) :- token(V2,V4,exclamationMark), sentence(V2), position(V4)

Table 7. Rule - Imperative Concept

$ARG1(c/country : name(n/name : op1\ \text{“}China\text{”})))$

Table 7 shows the learnt rule for this category.

3.3.8 Causal Concept

In this category, concept *cause* is invoked by tokens like *because, since* depicting a causal relation of two events in the sentence. The way AMR is defined allows causal relations to be represented in this way too. For example,

Sentence: ***Since*** *our road expenditures nowhere near match the increase in business or state domestic product , it is no wonder we suffer congestion problems .*

$(w2/wonder - 02$
$: ARG1(s/suffer - 01$
$: ARG0(w/we)$
$: ARG1(p/problem$
$: topic(c/congest - 01)))$
$: polarity-$
$: ARG1 - of(c2/$**cause-01**
$: ARG0(m/match - 01$
$: ARG1(t/thing$

concept(cause,V2) :- token(V2,V4,because), sentence(V2), position(V4)
concept(cause,V2) :- token(V2,V4,since), sentence(V2), position(V4)

Table 8. Rule - Causal Concept

$$
\begin{aligned}
&: ARG1 - of(e/expend - 01 \\
&\quad : ARG0w \\
&\quad : ARG2(r/road))) \\
&: polarity- \\
&: ARG1(t2/thing \\
&\quad : ARG2 - of(i/increase - 01 \\
&\quad\quad : ARG1(o2/or \\
&\quad\quad\quad : op1(b2/business) \\
&\quad\quad\quad : op2(p2/product \\
&\quad\quad\quad\quad : mod(d/domestic) \\
&\quad\quad\quad\quad : mod(s2/state)))))))
\end{aligned}
$$

The word *Since* invokes the concept *cause*.
Learnt rule for this category is in Table 8.

3.3.9 Abstract Concept

In this category, concepts like *person, organization* are aligned with proper nouns(actual names or people and organization). These are aligned when their children nodes are already aligned and are named entities as identified by Stanford

Parser.

Chapter 4

ALIGNING

In chapter 3, all the rules are learnt. Now these rules need to be used for aligning sentence-AMR pairs in the Aligner engine.

The learnt rules do not directly say the alignments. Instead they invoke concepts based on tokens in the sentence. But the invoked concepts are essentially the alignments too. So these rules are also used as alignments interchangeably. For example, a rule in modal concept category invokes concept *possible* if there is a token can in the sentence. This rule is used in aligner by saying - *if there is a concept 'possible' in the concept set of an data instance, and there is a token 'can' in the sentence, then the token aligns to this concept.*

I am using context also at code level to make sure concepts are not misaligned when the same concept appears more than once in the same data instance. For example,

Sentence: On the receiving end of the message were officials from giants like Du Pont ***and*** *Maytag , along with lesser knowns like Trojan Steel* ***and*** *the Valley Queen Cheese Factory .*

$(r/receive-01$

$:ARG0(p/person$

$:ARG0-of(h/have-org-role-91$

$$
\begin{aligned}
&: ARG1(a3/and \\
&\quad : op1(g/giant \\
&\quad\quad : example(a/\textbf{and} \\
&\quad\quad\quad : op1(c/company : name(d/name : op1\text{“}Du\text{”} : op2\text{“}Pont\text{”})) \\
&\quad\quad\quad : op2(c2/company : name(m/name : op1\text{“}Maytag\text{”})))) \\
&\quad : op2(c5/company \\
&\quad\quad : ARG1 - of(k/know - 01 \\
&\quad\quad\quad : degree(l/less)) \\
&\quad\quad : example(a2/\textbf{and} \\
&\quad\quad\quad : op1(c3/company : name(t/name : op1\text{“}Trojan\text{”} : op2\text{“}Steel\text{”})) \\
&\quad\quad\quad : op2(c4/company : name(v/name : op1\text{“}Valley\text{”} : \\
&op2\text{“}Queen\text{”} : op3\text{“}Cheese\text{”} : op4\text{“}Factory\text{”}))))) \\
&\quad : ARG2(o5/official))) \\
&: ARG1(t2/thing \\
&\quad : ARG1 - of(m3/message - 01)))
\end{aligned}
$$

In this example, there are two *and* concepts(bold) and its important to check the context while alining with *and* tokens to make sure they are aligned correctly.

The learnt rules are executed in an order so that no rule conflicts any other rule and rules that are very specific to concepts and particular tokens are executed first than rules that are more generic. Following is the order of execution of rules:

1. Concept as word tokens
2. Imperative concept

3. Concepts negated with prefix and suffix
4. Modal concepts
5. Negation concepts
6. Concepts as categorial variation of word tokens
7. Question concept
8. Abstract concept
9. Causal concept

Chapter 5

EXPERIMENTS AND RESULTS

The obtained alignments in Chapter 4 are evaluated through a series of experiments on the available alignment annotated dataset for AMR. The results are explained with proper justification.

5.1 Dataset

The data consists of 13,050 publicly available AMR/English sentence pairs[3]. Pourdamghani *et. al.* (2014) have hand aligned 200 of these pairs to be used as development and test sets[4,5]. Table 9 shows the details of AMR/English corpus used for training and testing. I train my aligner on 12,850 pairs of sentences and test on the 200 aligned sentences.

The results are computed for each category of AMR mentioned in Chapter 3 for the 200 aligned sentences. Also results are computed for the 200 aligned sentences after combining all the categories together.

[3]LDC AMR release 1.0, Release date: June 16, 2014 ; https://catalog.ldc.upenn.edu/LDC2014T12

[4]https://www.isi.edu/natural-language/mt/dev-gold.txt

[5]https://www.isi.edu/natural-language/mt/test-gold.txt

	Train	Dev	Test
Sent. pairs	12850	100	100
AMR tokens	430 K	3.8 K (52%)	2.3 K (55%)
ENG tokens	248 K	2.3 K (76%)	1.7 K (74%)

Table 9. AMR/English corpus. The number in parentheses is the percent of tokens aligned in gold annotation.

	Precision	Recall	F-Score
Pourdamghani et al. 2014	1.0	0.89	0.94
ILP Aligner	**1.0**	**1.0**	**1.0**

Table 10. Results - Category Modal Concepts

5.2 Results - Category Modal Concepts

Table 10 shows the results for this category. My system beats the state-of-art aligner in this category. Pourdamghani's system removes some stop-words to gain some precision on the cost of recall. 'Would' is one of the stop words which is not aligned in their system. My system has a rule learnt corresponding to 'would' modal in this category.

5.3 Results - Negation Category

Table 11 shows the results for this category. My system beats the state-of-art aligner in this category. Pourdamghani's system does not perform well on prefix and suffix negated words like *illegal(= not legal)*. My system has learnt separate rules(Concepts Negated with Prefix and Suffix category) for this type of words besided the regular negation words(no, not).

	Precision	Recall	F-Score
Pourdamghani et al. 2014	0.95	0.87	0.90
ILP Aligner	**1.0**	**0.9125**	**0.95**

Table 11. Results - Category Negation Concepts

	Precision	Recall	F-Score
Pourdamghani et al. 2014	0.966	0.90	0.930
ILP Aligner	**0.961**	**0.924**	**0.938**

Table 12. Results - Category Question Concepts

5.4 Results - Question Category

Table 12 shows the results for this category. My system beats the state-of-art aligner in this category for overall F-score. The precision loss is due to some sentences that have the word 'which' as a non-question word while the learnt rule considers it as a question word. The major recall hain is because Pourdamghani's stop-words list contains 'which' that is removed in their system for precision gain on cost of recall.

5.5 Results - Imperative Category

Table 13 shows the results for this category. My system does not perform well as compared to the state-of-art aligner in this category. The learnt rule in my system only looks for exclamation mark. There many sentences that are imperative type witout the presence of '!'. For example: *Just think if this money had been put on some good purpose.*

Here 'imperative' concept is aligned to 'think'.

	Precision	Recall	F-Score
Pourdamghani et al. 2014	1.0	0.76	0.86
ILP Aligner	**1.0**	**0.52**	**0.68**

Table 13. Results - Category Imperative Concepts

	Precision	Recall	F-Score
Pourdamghani et al. 2014	1.0	1.0	1.0
ILP Aligner	**1.0**	**1.0**	**1.0**

Table 14. Results - Category Concept as Word Tokens

	Precision	Recall	F-Score
Pourdamghani et al. 2014	0.93	0.95	0.94
ILP Aligner	**0.961**	**0.952**	**0.956**

Table 15. Results - Category Concept as Categorial Variation

5.6 Results - Concept as Word Tokens Category

Table 14 shows the results for this category. Both systems perform perfectly in this category as this is the simplest category.

5.7 Results - Concept as Categorial Variation Category

Table 15 shows the results for this category. My system beats the state-of-art aligner in this category. Pourdamghani's system depends on occurence of alignments of words in this category to appear in training data for it to align in test data.

	Precision	Recall	F-Score
Pourdamghani et al. 2014	1.0	0.75	0.85
ILP Aligner	**1.0**	**0.781**	**0.877**

Table 16. Results - Abstract Concept Category

5.8 Results - Abstract Concept Category

Table 16 shows the results for this category. My system beats the state-of-art aligner in this category. My system performs better in mine for the concept 'person' because in many sentences it is aligned to non-proper nouns which my system is able to handle during aligning.

5.9 Results - Causal Concept Category

Table 17 shows the results for this category. My system does not perform well as compared to the state-of-art aligner in this category. The major recall loss is because many sentences have implicit causality in them which my learnt rules are not able to capture. For example : *Knowing a tasty and free meal when they eat one, the executives gave the chefs a standing ovation.* Pourdamghani's system predicts the alignments for these implicit causalities better because in the traning set, they see some similar sentences. This increases their conditional probability P(causal concept | sentence fragment containing words knowing, let etc.) Essentially my rules trade recall loss for significant precision gain.

Overall results on Development and Test Data is shown in Table 18 and 19. some changes are made in rule learning and rule execution order based on the results of

	Precision	Recall	F-Score
Pourdamghani et al. 2014	0.931	0.92	0.92
ILP Aligner	**1.0**	**0.67**	**0.80**

Table 17. Results - Causal Concept Category

	Precision	Recall	F-Score
Pourdamghani et al. 2014	0.972	0.882	0.925
ILP Aligner	**0.951**	**0.813**	**0.869**

Table 18. Results - Development Dataset

	Precision	Recall	F-Score
Pourdamghani et al. 2014	0.955	0.847	0.898
ILP Aligner	**0.971**	**0.858**	**0.91**

Table 19. Results - Test Dataset

development data to improve the results obtained.

Data coverage with the categories AMR is split into is 88.3% which is almost equal to that of state-of-art aligner(88.6%).

Chapter 6

CONCLUSION AND FUTURE WORK

One of the main advantage of Inductive Logic Programming is that it enables the user to provide domain-specific background knowledge to be used in learning. The use of background knowledge enables the user both to develop a suitable problem representation and to introduce problem-specific constraints into the learning provess. In the application of AMR aligner in this work, domain-specific background knowledge is given for each AMR category and rules are being learnt for each category.

Inductive Logic Programming has been used in various Natural Language applications earlier like Constructing Biological Knowledge Bases by Extracting Information from Text Sources but this is the first time it is being used to align English sentences with their semantic representations. From the results obtained and analyzed, it can be concluded that ILP performs better than state-of-the-art aligner in most of the categories(Modal, Negation, Question, Word Tokens, Categorial Variations, Abstract Concept) where the information of data instances related to that category can be very well represented as background knowledge. It doesn't perform well in the categories(Imperative, Causal Concept) where information is implicit to sentences or if there is ambiguity in the sentences, especially when English is ambiguous at many places. It needs more precise information on the cost of generalization to be able to learn the missing rules.

Another outcome from the analysis in this work is the use of Catvar database in

background knowledge for two categories(Categorial Variation, Concepts negated with Prefix and Suffix) in which ILP based aligner performs better than state-of-art aligner. It proves good for the application for aligner as this is the first time it is being used in this kind of application.

The effectiveness of AI systems is limited by the machine's inability to explain its thoughts and actions to human users. Explainable AI has been in recent talks among research communities and is a program initiative of DARPA.

6.1 Explainability of the Inductive Logic Programming Approach

Explainable AI aims to create a suite of machine learning techniques that:

- Produce more explainable models, while maintaining a high level of learning performance (prediction accuracy); and
- Enable human users to understand, appropriately trust, and effectively manage the emerging generation of artificially intelligent partners.

Figure 5 shows that AI systems today suffer from explainability in their learnt model. The end user encounters an error in output and is unable to find out the reason because the learnt model is not explainable at all.

On the other hand, the learnt models in ILP systems are very easy to understand by humans. For example, the learnt model for Modal concepts category is:

$K \ = \ concept(possible, 0) : -token(0, 2, can).$

$K_v \ = \ concept(possible, V1) : -token(V1, V2, can).$

which means possible is a concept that aligns to can word in sentence. Clearly it is

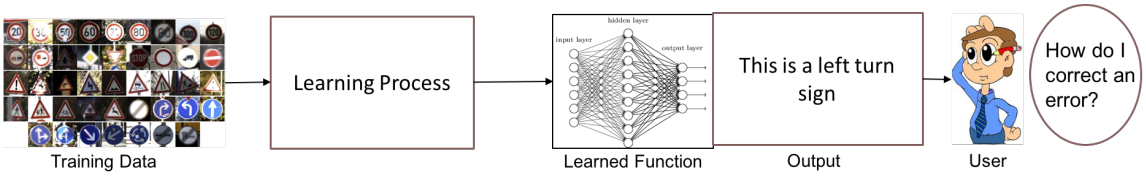

Figure 5. Explainability issues in AI systems today

human readable and understandable. Given this property, ILP systems are one of the best in having an explainable learnt model.

6.2 Future Work

Some of the applications of ILP based aligner as part of future work would be:

- Current AMR parsers are using either manual rule based aligner or statistical machine translation based aligner. It would be interesting to learn how ILP based aligner performs in AMR parsing
- The ILP based aligner in this work performs well on AMR semantic representation. Generalizing it to other semantic representations as their aligners would be another exciting work in this field.
- In the task of Machine Translation, an intermediate semantic representation can help the learning process. To benefit from this intermediate representation, an aligner would be very helpful in finding the links between AMR graph fragments and language phrases thereby providing fine grained translation correspondances during training.

BIBLIOGRAPHY

Artzi, Y., K. Lee and L. Zettlemoyer, "Broad-coverage ccg semantic parsing with amr.", in "EMNLP", pp. 1699–1710 (2015).

Banarescu, L., C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer and N. Schneider, "Abstract meaning representation for sembanking", in "LAW@ACL", (2013).

Baral, C., *Knowledge representation, reasoning and declarative problem solving* (Cambridge university press, 2003).

Brown, P. F., V. J. D. Pietra, S. A. D. Pietra and R. L. Mercer, "The mathematics of statistical machine translation: Parameter estimation", Computational linguistics **19**, 2, 263–311 (1993).

Chu, C. and S. Kurohashi, "Supervised syntax-based alignment between english sentences and abstract meaning representation graphs", arXiv preprint arXiv:1606.02126 (2016).

Finkel, J. R., T. Grenager and C. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling", in "Proceedings of the 43rd annual meeting on association for computational linguistics", pp. 363–370 (Association for Computational Linguistics, 2005).

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

Gelfond, M. and V. Lifschitz, "The stable model semantics for logic programming.", in "ICLP/SLP", vol. 88, pp. 1070–1080 (1988).

Habash, N. and B. Dorr, "A categorial variation database for english", in "Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1", pp. 17–23 (Association for Computational Linguistics, 2003).

Katzouris, N., A. Artikis and G. Paliouras, "Incremental learning of event definitions with inductive logic programming", Machine Learning **100**, 2-3, 555–585 (2015).

Manning, C. D., M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard and D. McClosky, "The stanford corenlp natural language processing toolkit.", in "ACL (System Demonstrations)", pp. 55–60 (2014).

Mitra, A. and C. Baral, "Addressing a question answering challenge by combining statistical methods with inductive rule learning and reasoning", (2016).

Moore, R. C., "Possible-world semantics for autoepistemic logic", Tech. rep., DTIC Document (1984).

Muggleton, S., "Inductive logic programming", New generation computing **8**, 4, 295–318 (1991).

Pourdamghani, N., Y. Gao, U. Hermjakob and K. Knight, "Aligning english strings with abstract meaning representation graphs.", in "EMNLP", pp. 425–429 (2014).

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

Ray, O., "Nonmonotonic abductive inductive learning", Journal of Applied Logic **7**, 3, 329–340 (2009).

Reiter, R., "A logic for default reasoning", Artificial intelligence **13**, 1-2, 81–132 (1980).

Toutanova, K., D. Klein, C. D. Manning and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network", in "Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1", pp. 173–180 (Association for Computational Linguistics, 2003).

Wang, C., N. Xue, S. Pradhan and S. Pradhan, "A transition-based algorithm for amr parsing.", in "HLT-NAACL", pp. 366–375 (2015).

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

APPENDIX A

ILP ALIGNER IMPLEMENTATION DETAILS

This appendix gives implementation details of the ILP based aligner, starting from raw data processing to aligning. From implementation point of view, the system consists of four major modules - *Data Preprocessing, Data Encoding, Rule Learning and Aligning*. Note that some of these modules differ from the ones discussed in Chapter 1 since these are strictly in coherence with code implementation perspective. Each of these modules is discussed in the following sections.

The code of this ILP based Aligner is available on GitHub. [6]

The tools used in the development process are the following:

- Stanford CoreNLP [7], for background knowledge
- Catvar 2.0 [8], for categorial variation clusters
- XHAIL [9], for Inductive Logic Programming

A.1 Data Preprocessing

The raw data consists of English sentences and AMR representation. Preprocessing steps on this raw data are as follows:

- The English sentence is first converted into lowercase.

[6] https://github.com/agarwalshubham2007/AMR-Aligner-ILP

[7] https://stanfordnlp.github.io/CoreNLP/

[8] https://clipdemos.umiacs.umd.edu/catvar/

[9] https://github.com/stefano-bragaglia/XHAIL

Algorithm : Constructing AMR Graph

```
1:  procedure AMR_GRAPH_CONSTRUCT(N_CURR, S_AMR)    ;   Constructing AMR graph from AMR
string S_AMR
2:      while M_R is not EOS
3:          if M_L is '('
4:              while M_R is not '(' and ')'
5:                  M_R +=1
6:              if M_R is '('
7:                  N_CURR is text between M_L and M_R
8:                  N_CURR.child = N_NEW
9:                  AMR_GRAPH_CONSTRUCT(N_NEW , S_AMR)
10:             else if M_R is ')'
11:                 N_CURR is text between M_L and M_R
12:                 return
13:         else if M_L is ')'
14:             M_R += 1
15:             M_L = M_R
16:             return
```

Figure 6. Algorithm : Constructing AMR Graph

- Using Stanford CoreNLP, background knowledge mentioned in Chapter 2 is extracted from the sentence.
- The AMR representation is stored in memory in a graph data structure using a recursive algorithm. The algorithm is shown in Figure 6.
- The graph is traversed to extract out all the concepts.
- The senses are removed from concepts. Example, in concept 'want-01', '-01' is removed.
- There are certain concepts that are enclosed in inverted commas, those commas are removed too.

A.2 Data Encoding

The preprocessed data is now encoded in ASP format using the extracted background knowledge and concepts in AMR representation. Following are its details:

- The common background knowledge for each data instance is
  - *position(I) :- token(S,I,L).*
  - *sentence(S) :- token(S,I,L).*
  - *lemmaList(L) :- token(S,I,L).*
- Then background knowledge for the category rule is being learnt is introduced.
  For example: Modal Concept Category
  modalConcepts(possible;likely;obligate;recommend;permit;prefer).
- From the concepts extracted in data processing, concepts related to this category are encoded as examples.
  For example :
  #example concept(possible,0).
- Concepts missing in this category for this example are encoded using not prefix.
  For example:
  #example not concept(obligate,0).
  .....
  .....
- Mode declarations are encoded for the category, rules are being learnt for.

Algorithm : Alignment

```
1:  procedure ALIGN(D_INS)   ;   Aligning data instance D_INS
2:     for each concept C in D_INS
3:        if C is not aligned already
4:           if isApplicableRule1(D_INS, C)
5:              assign alignment
6:           else if isApplicableRule2(D_INS, C)
7:              assign alignment
8:           .
9:           .
10:          .
```

Figure 7. Algorithm : Alignment

A.3 Rule Learning

The encoded data is given to XHAIL system for learning rules. Implementation details for rule learning are as follows:

- XHAIL system is not scalable for learning rules on large datasets. So rules are learnt on batches of 60 instances.
- The count of number of times every unique learnt rule is stored as this will decide the preference rule execution inside any category.
- The learnt rules are mentioned in Chapter 3.

A.4 Aligning

The learnt rules are then executed on every test data instance(see algorithm in Figure 7) in the following order in the implemented system on the development and test dataset:

- Concepts as Word Tokens

- Imperative
- Concepts Negated with Prefix
  - Negated polarity concept rule
  - Root word concept rule
- Modal Concepts
  - Concept *possible* invoked by *can*
  - Concept *obligate* invoked by *must*
  - Concept *obligate* invoked by *shall*
  - Concept *recommend* invoked by *should*
  - Concept *possible* invoked by *may*
  - Concept *obligate* invoked by *will*
  - Concept *possible* invoked by *would*
- Negation Concept
- Categorial Variation Concept
- Question Concept
  - *which* with *WDT* POS tag invokes *amr-unknown* concept
  - *what*, *who* or *whom* with *WP* POS tag invokes *amr-unknown* concept
  - *whose* with *WP$* POS tag invokes *amr-unknown* concept
  - *how*, *where*, *when* or *why* with *WRB* POS tag invokes *amr-unknown* concept
- Causal concept
  - *because* invokes *cause* concept
  - *since* invokes *cause* concept
- Imperative Concept