String to Semantic Graph Alignment

by

Melanie Tosik

in partial fulfillment of the requirements for the degree of

Bachelor of Science in Computational Linguistics

Department of Linguistics
University of Potsdam

March 30, 2015

Thesis supervisors:

Prof. Dr. Alexander Koller
Dr. Christoph Teichmann

Abstract

String to Semantic Graph Alignment

Melanie Tosik

The aim of this thesis was to re-evaluate the tools and methods described in a recent publication on “Aligning English Strings with Abstract Meaning Representation Graphs” (Pourdamghani et al., 2014). Based on the IBM translation models, a novel algorithm for aligning pairs of English sentences and linearized Abstract Meaning Representation (AMR) graphs at token level is implemented and evaluated. After retraining the alignment model, we obtain 87.2% and 85.9% F1 score on development and test sets, thereby verifying the reproducibility of the original approach. In addition, an extended analysis of the theoretical preliminaries as well as further experimental studies on the preprocessing step are described in the course of this thesis.

Zusammenfassung

Ziel dieser Arbeit war eine umfassende und detaillierte Evaluierung einer veröffentlichten Arbeit zum Thema “Aligning English Strings with Abstract Meaning Representation Graphs” (Pourdamghani et al., 2014). Auf der Grundlage der Übersetzungsmodelle, die in den späten achtziger Jahren von IBM entwickelt worden, wird ein neuer Algorithmus zur Alignierung von englischen Sätzen und Graphen im Abstract Meaning Representation (AMR) Format vorgestellt. Um Reproduzierbarkeit nachzuvollziehen, wird das Originalmodell unverändert auf dem originalen Datensatz trainiert. Wir erzielen einen F1 Score von 87.2% und 85.9% auf Development und Test Set, und übertreffen damit die Ergebnisse aus dem Originalpapier. Zusätzlich enthält diese Arbeit eine ausführliche Beschreibung der mathematischen Grundlagen, die methodisch angewandt werden, und die Ergebnisse weiterer experimenteller Studien zum Vorverarbeitungsschritt.

Acknowledgements

I am most grateful to friends and family, whose continued support and encouragement were invaluable over the last weeks. Special gratitude extends to Christoph Teichmann, for his availability and his help in guiding this thesis on the right track. My personal thanks are directed towards Alexander Koller, for his mentorship and understanding.

Declaration of Authorship

I certify that the work presented here is, to the best of my knowledge and belief, original and the result of my own investigations, except as acknowledged, and has not been submitted, either in part or whole, for another degree at this or any other university.

(Signature)

Berlin, March 30, 2015

Contents

List of Figures

List of Tables

Introduction

"Le langage est source de malentendus."
Language is the source of misunderstandings.

– Antoine de Saint Exupéry

Who did what to whom? A question that every human can easily answer in a given context, but is very difficult for computers to analyze directly. For more than two decades, natural language processing heavily relied on syntactic treebanks in order to enable computers to derive meaning from language. Following the release of the first large-scale treebank, the Penn Treebank (Marcus et al., 1993), many more syntactic treebanks have been developed for a wide variety of languages, and used to build state-of-the-art natural language processing systems such as part-of-speech (POS) taggers, parsers, semantic analyzers, and machine translation (MT) systems.

Moving from the analysis of grammatical structure to sentence semantics, however, we find that statistical parsers are ill-suited for producing meaning representations. In semantic analysis, we often encounter complex structures that are impossible to capture within the limits of tree structures. For example, semantic nodes will often be the argument of more than one predicate, and it will often be useful to exclude semantically vacuous words (like particles or complementizers), i.e. leave nodes unattached which do not add further meaning to the resulting representation.

To overcome this fundamental limitation and enable a more direct semantic analysis of whole sentences, emerging research is shifting towards parsing with graph-structured representations. Just as syntactic treebanks were of significant importance to improve upon syntactic parsers, Banarescu et al. (2013) recently released the first large, broad-coverage sembank: a set of English sentences paired with simple, readable semantic representations. Specifically, they annotate the logical meaning of sentences in Abstract Meaning Representation (AMR) – single rooted, directed graphs, that incorporate semantic roles, coreference, questions, modality, negation, and further linguistic phenomena.

However, these annotations do not include alignment links between the words in the input sentences and the corresponding concepts in the AMR graphs – for each English sentence, one AMR graph is given, but without any information as to how one representation was derived from the other. Nevertheless, analogous alignment links are an essential prerequisite for the extraction of universal generation rules from existing alignments, as well as the construction of semantic parsers for AMR.

Here, semantic parsing refers to the task of transforming natural language strings into formal, machine-readable knowledge representations which support automated inference. Currently, semantic parsing entails domain dependence: example application domains include ATIS, the *Air Travel Information Service* (Price, 1990), the *Robocup Coach Language* CLang (Chen et al., 2003), and *GeoQuery: A Database Query Application* (Wong and Mooney, 2006). In order to train a semantic parser for AMR, we need to know which spans of words in the input sentence invoke which concepts in the corresponding graph, i.e. we need alignment links between each English token and its AMR representation.

This alignment problem is closely related to the problem of Statistical Machine Translation (SMT): the translation of text from one language to another, based on parameterized statistical models. Machine translation was one of the first applications envisioned for computers (Weaver, 1949/1955), and remains a commercially and academically interesting challenge in Natural Language Processing (NLP). The first MT system was introduced by IBM in 1954, and a basic word-for-word translation system. Since then, various approaches to machine translation have been implemented, ranging from word-to-word translation, syntactic transfer, interlingual approaches, controlled language, and example-based translation to statistical machine translation.

In SMT, the goal is to find the most probable sentence in any target language, given a sentence in a foreign source language. In order to arrive at the most probable translation, words and phrases within sentence pairs in a parallel corpus are first aligned automatically. The probabilities of the different possible translations are then determined by a statistical model that uses the parallel corpus to efficiently estimate its parameters. Instead of translating, for example, English strings to their French counterparts, we are now trying to translate English strings into their corresponding AMR representations.

Recent work on "Aligning English Strings with Abstract Meaning Representation Graphs" (Pourdamghani et al., 2014) provides a detailed description of how to align English/AMR pairs at token level, along with gold alignments and evaluation results. The work at hand is a detailed re-evaluation of the tools and methods presented in Pourdamghani et al. (2014), and as such introduces their precise methodology in what follows.

We start by reviewing related work in Chapter 2. Chapter 3 provides the necessary background information on the AMR language 3.1, along with an introduction to SMT and the IBM translation models 3.2. Subsequently, Chapter 4 details the approach taken, divided into the preprocessing 4.1, training 4.2, and postprocessing step 4.3. Chapter 5 describes the experimental evaluation 5.1, before we move on to discuss the results of our experiments 5.2. Chapter 6 provides conclusions and directions for future work.

Related work

To set the context for the work presented in this thesis, related work is reviewed in what follows. Generally, there are three different kinds of translation models that can be used to solve the alignment task: word-based translation models (Brown et al., 1993; Vogel et al., 1996; Knight, 1999), phrase-based translation models (Och and Ney, 2000, 2004; Marcu and Wong, 2002; Koehn, 2004; Chiang, 2007), and syntax-based translation models (Alshawi et al., 2000; Wu, 1997; Yamada and Knight, 2001).

As mentioned beforehand, Brown et al. (1993) originally approached SMT by implementing the five IBM translation models. These models were word-based, i.e. each source word is aligned to at most one target word. The currently best performing systems follow the phrase-based approach, enabling the models to produce more general alignments. However, phrase-based models often build upon the original word-based approach, in the sense that phrases are extracted from the alignments computed by word-based models. Phrase-based models that train phrases directly usually underperform these combined systems (DeNero et al., 2006).

Sentence alignment is a crucial preprocessing step for all sentence-internal alignment approaches. The standard algorithm was introduced by Gale and Church (1993), and was exclusively based on the character lengths of candidate segments. They exploited the fact that in 90% of alignments, no word reordering takes place.

Moving beyond the scope of natural language translation, Jones et al. (2012) recently described alignment of English string and semantic graphs, though they limited their evaluation to the *Geo-Query* domain (Wong and Mooney, 2006) and failed to provide adequate details on their methods. Subsequently, Flanigan et al. (2014) introduced the first approach to parsing English sentences into AMR. Their method is based on a heuristic aligner which uses a set of rules to align AMR concepts to their corresponding English spans. This method is restricted to the pre-defined set of alignment rules, and thus will not improve as more parallel AMR/English data becomes available. Finally, Pourdamghani et al. (2014) overcome this limitation by linearizing the AMR graphs and modifying the original word-based translation models to enable symmetrical parameter learning.

Prerequisites

This chapter contributes the fundamental knowledge necessary to comprehend how the alignments are derived. Since the ultimate goal is to discuss methods on the alignment of natural language input with corresponding semantic graphs written in AMR, Section 3.1 provides a detailed description of the format of the AMR language. Section 3.2 introduces the IBM translation models, which will later be used to compute the English/AMR alignments.

3.1 Abstract Meaning Representation

The Abstract Meaning Representation (AMR) language is presented in Banarescu et al. (2013), and described in more detail in the AMR annotation guidelines[1]. In a nutshell, AMR graphs are rooted, labeled graphs, comprising whole sentences. They are intended to abstract away from syntactic representations, in the sense that sentences which are similar in meaning should be assigned the same AMR, even if they are not equally worded. By nature, the AMR language is biased towards English – it is not meant to function as an international auxiliary language.

AMRs can be written in two different notations. For human reading and writing, the PENMAN notation (Matthiessen and Bateman, 1991) is adapted. For computer processing, a traditional graph notation is used. A basic example of both notations is illustrated in Figure 3.1.

English sentence:

"I do not understand", said the little prince.

Penman notation:

```
(s / say-01
   :arg0 (p / prince
            :mod (l / little))
   :arg1 (u / understand-01
            :arg0 p
            :polarity -))
```

Graph notation:

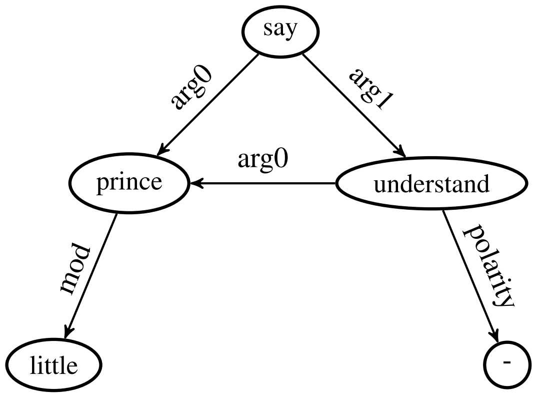


Figure 3.1: Example sentence and its annotated AMR graph[2]

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

As can be seen, AMR graphs are rooted, labeled, directed, acyclic graphs. Each node in the graph represents a semantic concept. These concepts can either be English words (*prince*), PropBank framesets (*say-01*) (Palmer et al., 2005), or special keywords. In PENMAN notation, *(p / prince)* refers to an instance *p* of the concept prince. Edge labels denote the relations that hold between entities. For example, *(p / prince :mod (l / little))* denotes the modality of our prince being little. These tokens, including *:mod* or *:arg0*, are referred to as AMR role tokens. It is worth noting that entities with multiple semantic roles are represented with only a single node in the graph. In such cases, variables are re-used and re-entrancies are annotated in the graph.

3.2 IBM Translation Models

The alignment phase is based on the IBM models, introduced by (Brown et al., 1993). They describe five statistical models of translation, as well as algorithms to efficiently estimate the parameters of these models given a set of translation pairs. For any such pair, the translation models also assign a probability to each of the possible word-by-word alignments. Instead of translating from one natural language to another, our task will be to translate natural language strings into semantic graph representations, i.e. translate from English into AMR.

We are given an English sentence e: e is a sequence of words e_1, e_2 … e_m, where m is the length of the sentence, and e_j for $j \in \{1 \dots m\}$ is the j'th word in the sentence. We would like to find the best AMR translation a, equal to a_1, a_2 … a_l, where l is the length of the AMR. We are also given a parallel English/AMR corpus, i.e. a set of gold English/AMR translations $(e^{(k)}, a^{(k)})$ for $k = 1 \dots n$, where $e^{(k)}$ is the k'th English sentence in the training corpus, $a^{(k)}$ is the k'th AMR string, and $a^{(k)}$ is a translation of $e^{(k)}$.

To solve the translation task, we can now build a generative model which will turn out to be an instance of a noisy channel model. A noisy channel is metaphorically used for many different engineering problems, most commonly to model actual noise in a speech recognition task. Figure 3.2 illustrates the main idea.

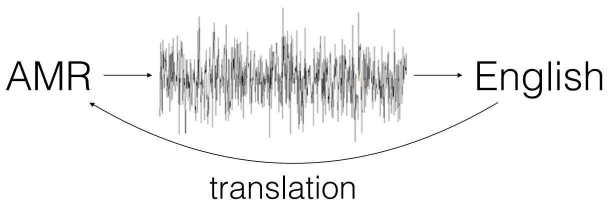


Figure 3.2: Noisy channel approach

We start with an AMR string, which gets corrupted by an abstract concept of noise and becomes English by the time it gets on to the printed page. We seek to recover the most likely AMR, given the observed English output.

[2]Taken from public Release 1.4 of the AMR Bank. (1,562 sentences from *The Little Prince*; November 14, 2014; http://amr.isi.edu/download/amr-bank-v1.4.txt)

To this end, we have to consider

a. how likely the utterance of the AMR sentence was in the first place: $P(a)$

b. how AMR can be translated into English: $P(e|a)$

Note that, although the task is to translate from English into AMR, we have to reason about a probability $P(e|a)$ – the probability of an English sentence given an AMR. This is commonly referred to as backwards reasoning, and intuitively explained in Knight (1999). Consider a sentence e as a set of medical symptoms. Naturally, there are many different diseases that could cause these symptoms. If we build a generative model like the one above, we can reason about the probability of any disease a occurring, as well as the probability that symptoms e will be caused by any concrete disease a. These probabilities are $P(a)$ and $P(e|a)$, and they are likely to conflict: a common disease might often cause symptoms e, and another, very rare disease might often cause symptoms e. Since biologists have knowledge of how diseases cause symptoms, they can build reliable computer models to predict $P(e|a)$. On the other hand, it is not obvious how to build a single model that can draw conclusions from symptoms to diseases, i.e. $P(a|e)$. Additionally, we also have independent sources of information about $P(a)$, like hospital records (or monolingual corpora).

By applying Bayes' theorem, we can use the generative model to define the best (most likely) translation $\hat{a}$:

$$\hat{a} = argmax_a\, P(a|e) = argmax_a\, \frac{P(a)P(e|a)}{P(e)}$$

Since e is fixed, we can write:

$$\hat{a} = \underbrace{\text{argmax}_a}_{\text{decoder}} \overbrace{P(a)}^{\text{language model}} \underbrace{P(e|a)}_{\text{translation model}}$$

Thus, we identify three components of the noisy channel model:

A **language model** assigns a probability $P(a)$ given the AMR input string a. Ideally, well-formed AMR strings get assigned a high probability, while ill-formed strings should receive a low probability. We can estimate the parameters for the language model from a monolingual corpus of linearized AMR graphs. Typically, this is done using a smoothed n-gram language model.

A **translation model** models the lexical correspondences between English and AMR and assigns $P(e|a)$ given an English/AMR pair (e,a). The more likely (e,a) is a translation, the higher is $P(e|a)$. The parameters of the translation model are estimated from the translation examples in our parallel English/AMR corpus.

A **decoder**, an effective and efficient search technique to find $\hat{a}$. Since the search space is infinite and rather unstructured, a heuristic search will be applied.

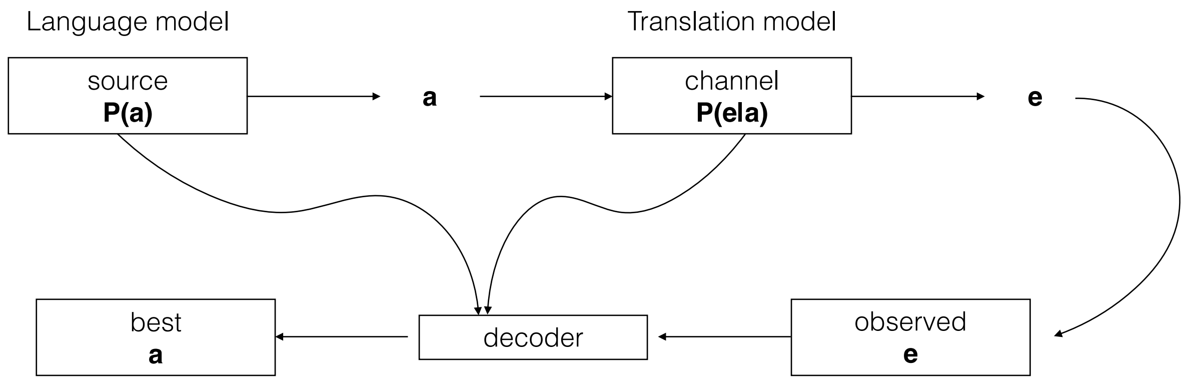
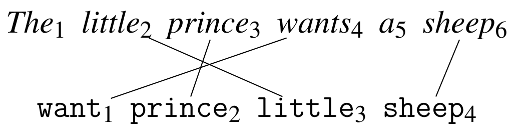


Figure 3.3: Generative noisy channel model

Figure 3.3 shows how the three components are connected within the noisy channel model. An intuition about this model architecture is very useful, because the translation model $P(e|a)$ can be factored to identify the alignments (the lexical correspondences) between the words in the English sentence $e = e_1 \dots e_m$ and the words in the AMR $a = a_1 \dots a_l$. For example, here are the alignments for the English sentence *The little prince wants a sheep*, and the corresponding nodes in the output AMR:


The_1 $little_2$ $prince_3$ $wants_4$ a_5 $sheep_6$
want₁ prince₂ little₃ sheep₄

We can formalize alignment with an alignment function g, which maps an AMR target token at position j to the English source word at position i:

$$g : j \rightarrow i$$

For the example above, we obtain the following mapping:

$$g : \{1 \rightarrow 4,\ 2 \rightarrow 3,\ 3 \rightarrow 2,\ 4 \rightarrow 6\}$$

Note that in this example, there are only one-to-one alignments. Quite possibly, we could encounter one-to-many or many-to-many alignments for English/AMR pairs. Generally, words can be reordered, dropped, or inserted during translation.

For every word in the English input sequence, the translation model needs to incorporate its translation, the number of necessary words in the AMR, the position of the translation within the sentence, and the number of words that need to be generated from scratch. Therefore, the IBM models characterize $P(e|a)$ with four corresponding parameters t, n, d, and p_1, denoting the following:

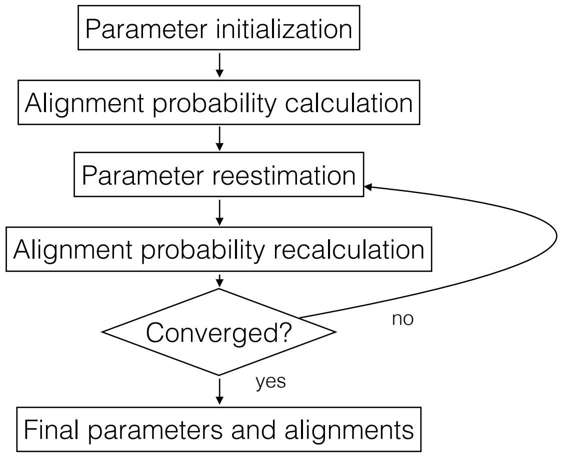

Parameter initialization
Alignment probability calculation
Parameter reestimation
Alignment probability recalculation
Converged?
no
yes
Final parameters and alignments

Figure 3.4: EM algorithm

The **lexical probability** t, the probability that the English token *wants* is translated into the AMR token `want`: $t(\texttt{want}|wants)$.

The **fertility** n, the probability that *wants* generates one AMR token: $n(1|wants)$. In the example, the words *the* and *a* are zero fertility words, which do not get translated at all.

The **distortion** d, the probability that the English word in the i'th position generates the AMR token in the j'th position, given the lengths of the source and target strings, m and l: $d(i|j,m,l)$

The **probability** p_1, the probability that spurious words (words which did not occur in the source string, but are present in the translation) are generated from *NULL*.

We can estimate these parameters together with the word alignments using the expectation-maximization (EM) algorithm. EM is an iterative method which maximizes the likelihood estimates of parameters in statistical models, where the model depends on unobserved (latent) variables. In the case of word alignments, we are trying to maximize the conditional likelihood of the data over the above set of parameters θ:

$$\theta_{e|a} = argmax\ L_{\theta_{e|a}}(e|a)$$

Note that it is always possible to add the inverse translation model to the generative one:

$$\theta_{a|e} = argmax\ L_{\theta_{a|e}}(a|e)$$

Figure 3.4, adapted from Espanã-Bonet (2010), illustrates how the algorithm works. In the beginning, all model parameters are initialized uniformly (all alignments are equally likely). Afterwards, each EM iteration alternates between an expectation (E) and a maximization (M) step. In the expectation step, a function for the expectation of the log-likelihood is created and evaluated using the current estimate for the parameters. The model is applied to the data, and the hidden alignments are assigned probabilities to possible values. The maximization step computes parameters maximizing the expected log-likelihood found on the E step – the model is estimated from the data. The assigned values are taken as a fact, and weighted word counts are collected from the parallel AMR/English corpus. These parameter-estimates are then used to determine the distribution of the latent variables in the next E step. The two steps are iterated until convergence is reached.

Typically, the IBM models are applied hierarchically, breaking up the translation process into smaller steps. Since, by definition, only IBM Model 1 has a unique global maximum that the EM algorithm is guaranteed to find, the training of any higher model always builds on the previous one. In a nutshell, the five different models can be characterized as follows:

Model 1 only uses the lexical translation probabilities estimated from the parallel corpus.

Model 2 adds the absolute reordering model.

Model 3 adds the fertility model, introducing a sampling method over high probability alignments instead of directly collecting words counts from the data.

Model 4 is the relative reordering model.

Model 5 fixes deficiency.

In addition, we can define a Hidden Markov Model (HMM) based alignment model. Since words do not usually move independently of each other, we condition each word movement on the previous word and define an alignment model:

$$p(g(i)|i, g(i-1), m)$$

By the definition of fertility, the IBM models can only create one-to-many alignments, and therefore learn the alignments asymmetrically. The alignment function g may return the same value for different input (one-to-many mapping), but never the other way around (no many-to-one mapping).

Methodology

Pourdamghani et al. (2014) introduce the first algorithm for automatically aligning English and AMR that will improve with more data. Just like in SMT, they define a generative model from AMR graphs to strings, and apply EM training to uncover the hidden alignments.

Here is another example English/AMR pair:

```
It was a picture of a boa constrictor digesting an elephant.

(p / picture-01
   :arg0 (i / it)
   :arg1 (b2 / boa
            :mod (c / constrictor)
            :arg-of (d / digest-01
                       :arg1 (e / elephant))))
```

For this example, we would obtain the following alignments:

$\texttt{it}_0$ $\texttt{was}_1$ $\texttt{a}_2$ $\texttt{picture}_3$ $\texttt{of}_4$ $\texttt{a}_5$ $\texttt{boa}_6$ $\texttt{constrictor}_7$ $\texttt{digesting}_8$ $\texttt{an}_9$ $\texttt{elephant}_{10}$ $\texttt{.}_{11}$

```
(p / picture-01~e.3
   :arg0 (i / it~e.0)
   :arg1 (b2 / boa~e.6
            :mod (c / constrictor~e.7)
            :arg-of (d / digest-01~e.8
                       :arg1 (e / elephant~e.10))))
```

In practice, alignments are additionally stored in a more machine-readable format, for example:

```
3-1 0-1.1 6-1.2 7-1.2.1 8-1.2.2 10-1.2.2.1
```

In this format, alignment elements are dashed pairs of English token indices (starting at 0) and the indexed position in the AMR graph, where 1 always denotes the root node, where *x.i* is the *i*'th sub-node of node *x* (starting at 1), and where *x.r* refers to the role of AMR node *x*. For example, 7-1.2.1 refers to the English word with index 7 (*constrictor*), which gets aligned to the first sub-node (*constrictor*) of the second sub-node (*boa*) of the root node (*picture*).

The methodology of Pourdamghani et al. (2014) is described in this section. It is divided into three subsections, detailing the preprocessing 4.1, the training 4.2, and the postprocessing step 4.3.

4.1 Preprocessing

During preprocessing, the AMR graph structures are first linearized into strings by employing a depth first search from the root node of the graph, and printing the subsequent nodes in the order they are visited. For example, the example AMR graph from above changes into:

```
p / picture-01 :arg0 i / it :arg1 b2 / boa :mod c / constrictor :arg-of
d / digest-01 :arg1 e / elephant
```

The original graph structure is recorded to make it possible to rebuild the AMR graph once the training is complete. All English and AMR tokens are lowercased, before specific stop words are removed from both sides of the input:

```
it picture of boa constrictor digesting elephant

picture it boa constrictor digest elephant
```

The stop word lists are build from existing resources, taking into account how English/AMR typically align. Articles, relative pronouns, *to be* verbs, and punctuation marks only rarely align to AMR tokens, and are therefore removed from the English sentences. On the AMR side, it is mostly role tokens like *:arg0* or *:name* which are filtered, in addition to the *instance-of* token */* and tokens like *age-quantity* or *date-entity*. Again, the positions of all tokens are stored and to be recovered during postprocessing. On top of stop word removal, word sense indicators and quotation marks included in AMR tokens are deleted (for example, *picture-01* changes into *picture*). Except for role tokens in AMR, all tokens are then stemmed into their first four letters:

```
it pict of boa cons dige elep

pict it boa cons dige elep
```

After stemming, English words and their corresponding AMR concepts are often spelled the same. All English/AMR pairs for which this holds true are added to the corpus to help the training phase by incorporation lexical information. Moreover, there are English tokens which can be translated into multiple AMR tokens. For example, *short* could be translated into *shorter* or *short :degree more*. A pre-built set of such English words is extracted, and corresponding pairs are added to the corpus as well.

4.2 Training

Recall that during the training phase, we want to use the EM algorithm to estimate the parameters of our generative model together with the word alignments. More formally, we are trying to maximize the conditional likelihood L of the data over set of parameters θ:

$$\theta_{a|e} = argmax\, L_{\theta_{a|e}}(a|e)$$

$$\theta_{e|a} = argmax\, L_{\theta_{e|a}}(e|a)$$

Recall also that the IBM models can only create one-to-many alignments, and therefore can only learn the alignments asymmetrically. This is not ideal, since word alignment is a symmetric problem, and we would want many-to-many mappings. Therefore, Pourdamghani et al. (2014) modify the objective function of the training phase to estimate the parameters symmetrically:

$$\theta_{a|e},\ \theta_{e|a} = argmax\, L_{\theta_{a|e}}(a|e) + L_{\theta_{e|a}}(e|a)$$

This modification is subject to:

$$\theta_{a|e}\theta_e = \theta_{e|a}\theta_a = \theta_{a,e}$$

After the constraint is relaxed to $\theta_{a|e} = \theta_{e|a}$, the following iterative process is applied:

1. Using EM, optimize the first part of the objective function: $\theta_{a|e} = argmax\, L_{\theta_{a|e}}(a|e)$
2. Satisfy the constraint: set $\theta_{e|a} \propto \theta_{a|e}$
3. Using EM, optimize the second part of the objective function: $\theta_{e|a} = argmax\, L_{\theta_{e|a}}(e|a)$
4. Satisfy the constraint: set $\theta_{a|e} \propto \theta_{e|a}$
5. Iterate

Steps 1 and 3 simply apply the IBM models. Steps 2 and 4 initialize the EM parameters, based on the parameters estimated in previous iterations. Since we wish to compute the alignments symmetrically, Steps 2 and 4 initialize the parameters which account for both directions of the alignments: the translation table and the distortion table. For each English word **e** and each AMR token **a**, the translation is initialized as follows:

$$t_{\mathbf{e}|\mathbf{a}}(\mathbf{e}|\mathbf{a}) = t_{\mathbf{a}|\mathbf{e}}(\mathbf{a}|\mathbf{e})$$

The distortion table is initialized similarly. In each iteration, the fertility table is initialized with its estimated value from the previous iteration. After estimating the parameters, alignments are computed using the Viterbi algorithm in both directions of the IBM models.

4.3 Postprocessing

The postprocessing step aims at rebuilding the aligned AMR. Since we recorded the original graph structure, we can easily revert the AMR structure after we reinsert the previously filtered stop words.

Experiments

This section provides a detailed description of the data and the evaluation method 5.1, followed by a discussion of the experimental results 5.2.

5.1 Evaluation

We can evaluate the alignment model by comparing the model output on different data sets to a manually annotated gold standard. The latest release of the AMR bank is publicly available and consists of 13050 AMR/English pairs. For 200 of these pairs, a hand-aligned gold standard has been provided by Pourdamghani et al. (2014). We will use 100 of these sentences as development set, and the remaining 100 sentences as test set. The model is trained on all the data combined.

A detailed description of the data is presented in Table 5.1, taken from Pourdamghani et al. (2014). In addition to the mere counts, the percentages of aligned tokens in the gold annotation is specified in brackets.

	Training	Development	Test
Sentence pairs	13050	100	100
AMR tokens	465K	3.8K (52%)	2.3K (55%)
AMR role tokens	226K	1.9K (23%)	1.1K (22%)
English tokens	248K	2.3K (76%)	1.7K (74%)

Table 5.1: Description of the English/AMR corpus

Both the IBM models and the HMM model are implemented in MGIZA++ (Gao and Vogel, 2008), a word alignment tool based on GIZA++, which has been extended to support multi-threading, resume training, and incremental training. To compute the alignments, IBM Model 1 and HMM are run for 5 iterations each, followed by the training algorithm on Model 4 for 4 iterations.

We evaluate the quality of the alignments intrinsically, by comparing them to the corresponding gold alignments. Given a set of computed alignments A and a set of gold alignments G, we can define the standard measures of precision, recall, and F1 score.

Assume that, for our example sentence, *It was a picture of a boa constrictor digesting an elephant*, we compute the following alignments A:

$$A = \{\texttt{3-1, 6-1.2.1, 7-1.2, 8-1.2.2, 10-1.2.2.1}\}$$

We want to evaluate them against the gold standard alignments G:

$$G = \{\texttt{3-1, 0-1.1, 6-1.2, 7-1.2.1, 8-1.2.2, 10-1.2.2.1}\}$$

We define:

$$Prec(A,G) = \frac{|G \cap A|}{|A|} = \frac{3}{5}$$

$$Rec(A,G) = \frac{|G \cap A|}{|G|} = \frac{3}{6}$$

$$F1(Prec,Rec) = \frac{2 \times Prec \times Rec}{Prec + Rec} = \frac{2 \times \frac{3}{5} \times \frac{3}{6}}{\frac{3}{5} + \frac{3}{6}} = \frac{6}{11} \approx 0.55$$

5.2 Results

Table 5.2 shows the results reported in the original paper by Pourdamghani et al. (2014) compared to the results we achieve by simply rerunning their candidate model, in addition to one experiment which reintegrates specific stopwords. Since Pourdamghani et al. (2014) already show that their training method outperforms IBM Model 1, HMM, and IBM Model 4, we limit the evaluation of our results to their best-performing model IBM Model 4+, referred to as *Original*. We reproduce their results on the same data and without any changes to the source code[1] (*Remake*). After retraining, we obtain slight improvements on the development set and a recall gain of 4% on the test set, resulting in a 2.8% increase on F1 score.

These improvements might be due to differing implementations of the evaluation metrics. It might for instance be desirable to assign partial alignment scores for partially correct alignments, e.g. for a correctly aligned node with a wrong sub-node alignment. However, the comparable results on the development set suggest otherwise, and the test set is additionally stated to be an intrinsically harder set. Tentatively, it remains unclear what causes the gain in accuracy. In order to verify the different procedures applied during the preprocessing steps, we conduct additional experiments concerning stop word filtering and stemming.

As Table 5.1 shows, only about 25% of AMR role tokens (such as *:arg0* or *:extent*) are aligned in the gold annotation. Based on this observation, Pourdamghani et al. (2014) decide to exclude

[1]Available from `http://www.isi.edu/~damghani/`

	[%]	Precision	Recall	F1 Score
Development set	Original	94.1	80.0	86.5
	Remake	94.3	81.1	87.2
	+Stopwords	74.5	83.1	78.6
Test set	Original	92.4	75.6	83.1
	Remake	93.3	79.6	85.9
	+Stopwords	81.0	83.1	82.0

Table 5.2: Experimental results on IBM Model 4+

a number of specific role tokens from the linearized AMR strings. Moreover, they also remove a handful of English stop words, including articles and relative pronouns. Indeed, if these stop words are not removed from the input sequences (*+Stopwords*), precision accuracy drops by 19.8% compared to the *Remake* baseline. On the other hand, we see a 2% increase in recall, which is expected given that some of the tokens which had previously been excluded are now included in the training phase. However, with respect to the overall quality of the alignments, it seems reasonable to trade the relatively small recall cost for the large gain in precision.

Secondly, it is not obvious why the best results should be achieved by stemming all tokens into their first four letters. We conduct an empirical study and compute alignments for stemming tokens into their first three letters (*Stem 3*), as well stemming tokens into their first five letters (*Stem 5*). The results are presented in Table 5.3.

	[%]	Precision	Recall	F1 Score
Development set	Stem 3	94.2	80.4	86.7
	Stem 5	87.2	78.6	82.7
Test set	Stem 3	92.3	77.2	84.1
	Stem 5	87.1	77.3	81.9

Table 5.3: Results of the stemming experiments on Model 4+

Somewhat surprisingly, the *Stem 3* model performs just slightly worse than the *Remake* baseline, and still better than the *Original* model. This is surprising, since the average length of English words is usually fixed at just above five, and stemming into only three letters could easily appear to be too substantial to be effective. On the contrary, we achieve the lowest F1 scores on both development and test set with the *Stem 5* model. On the test set, the recall score improves minimally compared to *Stem 3*. In general, our results verify the initial hypothesis that stemming all tokens into their first four letters maximally improves the overall alignment scores.

Conclusion and Future Work

The aim of this work was to investigate and extend prior work on aligning English strings with semantic graphs written in Abstract Meaning Representation (AMR). Before the individual English/AMR pairs are aligned at token level, a series of preprocessing steps needs to be applied. Most importantly, this includes linearizing the AMR graphs into strings and stemming all tokens into their first four letters.

The training phase is based on the IBM translation models and the definition of a generative model from AMR graphs to strings. Previously hidden alignment links are subsequently uncovered using Expectation-Maximization (EM) algorithm. Pourdamghani et al. (2014) developed a novel algorithm for word-to-word alignment by modifying the objective function of the IBM models in order to allow for symmetrical parameter learning during EM training.

They use MGIZA++ as existing implementation of the IBM models. By reproducing their experiments, we arrive at 87.2% and 85.9% F1 score on the original development and test sets. These results are higher than the original baseline, but the improvements might be due to minor differences in the implementation of the evaluation metrics.

We conducted three additional experiments on the preprocessing step, affecting stop word removal and stemming. We verify that the settings reported in the original paper result in the best model performance by providing an extensive insight in the properties of string-graph alignment.

Since stemming English and AMR tokens into their *n* first letters proves to have a large impact on the alignment accuracy, it could be worth investigating how stemming all tokens with an off-the-shelf system would affect the results. Furthermore, computing alignments between English sentences and AMR graphs is essentially groundwork for attempting the downstream extraction of English-AMR generation rules and subsequently whole-sentence semantic representations at large. Therefore, the next logical step would be to use the alignment algorithm to automatically parse English into AMR and vice versa.

References

Hiyan Alshawi, Shona Douglas, and Srinivas Bangalore. Learning dependency translation models as collections of finite-state head transducers. *Computational Linguistics*, 26(1):45–60, 2000.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract Meaning Representation for Sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria, 2013. Association for Computational Linguistics.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311, June 1993.

Mao Chen, Klaus Dorer, Ehsan Foroughi, Fredrick Heintz, ZhanXiang Huang, Spiros Kapetanakis, Kostas Kostiadis, Johan Kummeneje, Jan Murray, Itsuki Noda, Oliver Obst, Pat Riley, Timo Steffens, Yi Wang, and Xiang Yin. *Users Manual: RoboCup Soccer Server — for Soccer Server Version 7.07 and Later*. The RoboCup Federation, Februar 2003.

David Chiang. Hierarchical Phrase-Based Translation. *Computational Linguistics*, 33(2):201–228, June 2007.

John DeNero, Dan Gillick, James Zhang, and Dan Klein. Why Generative Phrase Models Underperform Surface Heuristics. In *Proceedings of the Workshop on Statistical Machine Translation*, StatMT '06, pages 31–38, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

Cristina Espanã-Bonet. Statistical Machine Translation – A practical tutorial, March 2010. Tutorial at MOLTO kick-off meeting.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and A. Noah Smith. A Discriminative Graph–Based Parser for the Abstract Meaning Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436. Association for Computational Linguistics, 2014.

William A. Gale and Kenneth W. Church. A Program for Aligning Sentences in Bilingual Corpora. *Computational Linguistics*, 19(1):75–102, March 1993. ISSN 0891-2017.

Qin Gao and Stephan Vogel. Parallel implementations of word alignment tool. In *Proceedings of the ACL 2008 Software Engineering, Testing, and Quality Assurance Workshop*, 2008.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. Semantics-Based Machine Translation with Hyperedge Replacement Grammars. In *Proceedings of COLING 2012*, pages 1359–1376, December 2012.

Kevin Knight. A Statistical MT Tutorial Workbook, April 1999. Prepared in connection with the JHU summer workshop.

Philipp Koehn. Pharaoh: A Beam Search Decoder for Phrase-Based Statistical Machine Translation Models. In *Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 115–124, Washington, DC, USA, 2004.

Daniel Marcu and William Wong. A Phrase-Based, Joint Probability Model for Statistical Machine Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 133–139, Philadelphia, Pennsylvania, 2002.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, June 1993.

Christian Matthiessen and John A Bateman. *Text generation and systemic-functional linguistics: experiences from English and Japanese*. Pinter Publishers, 1991.

Franz Josef Och and Hermann Ney. Improved Statistical Alignment Models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447, 2000.

Franz Josef Och and Hermann Ney. The Alignment Template Approach to Statistical Machine Translation. *Computational Linguistics*, 30(4):417–449, December 2004.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106, March 2005.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. Aligning English Strings with Abstract Meaning Representation Graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 425–429, Doha, Qatar, October 2014. Association for Computational Linguistics.

P. J. Price. Evaluation of Spoken Language Systems: The ATIS Domain. In *Proceedings of the Speech and Natural Language Workshop*, pages 91–95, Hidden Valley, PA, 1990.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. HMM-based Word Alignment in Statistical Translation. In *Proceedings of the 16th Conference on Computational Linguistics – Volume 2*, COLING '96, pages 836–841, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics.

Warren Weaver. Translation. In William N. Locke and A. Donald Boothe, editors, *Machine Translation of Languages*, pages 15–23, Cambridge, MA, 1949/1955. MIT Press. Reprinted from a memorandum written by Weaver in 1949.

Yuk Wah Wong and Raymond J. Mooney. Learning for Semantic Parsing with Statistical Machine Translation. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 439–446, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

Dekai Wu. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, 23(3):377–403, September 1997.

Kenji Yamada and Kevin Knight. A Syntax-based Statistical Translation Model. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 523–530, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics.