Abstract Meaning Representation for Paraphrase Detection

Fuad Issa[†] **Marco Damonte**[†] **Shay B. Cohen**[†] **Xiaohui Yan**[‡] **Yi Chang**[‡]
[†] School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh EH8 9AB, UK
[‡] Huawei Technologies, San Jose, CA 95050, USA
issa.fuad@gmail.com m.damonte@sms.ed.ac.uk
scohen@inf.ed.ac.uk {yanxiaohui2,yi.chang}@huawei.com

Abstract

Abstract Meaning Representation (AMR) parsing aims at abstracting away from the syntactic realization of a sentence, and denoting only its meaning in a canonical form. As such, it is ideal for paraphrase detection, a problem in which one is required to specify whether two sentences have the same meaning. We show that naïve use of AMR in paraphrase detection is not necessarily useful, and turn to describe a technique based on latent semantic analysis in combination with AMR parsing that significantly advances state-of-the-art results in paraphrase detection for the Microsoft Research Paraphrase Corpus. Our best results in the transductive setting are 86.6% for accuracy and 90.0% for F_1 measure.

1 Introduction

Abstract Meaning Representation (AMR) parsing focuses on the conversion of natural language sentences into AMR graphs, aimed at abstracting away from the surface realizations of the sentences while preserving their meaning.

We make a first step towards showing that AMR can be used in practice for a task that requires identifying the canonicalization of language: paraphrase detection. In a "perfect world" using AMR to test for paraphrasing relation of two sentences should be simple. It would require finding the two AMR parses for each of the sentences, and then checking whether they are identical. Since AMR is aimed at abstracting away from the surface form which is used to express meaning, two sentences should be paraphrases only if they have identical AMRs. For instance, the three sentences:

1. He described her as a curmudgeon,
2. His description of her: curmudgeon,
3. She was a curmudgeon, according to his description.

should result in the same AMR graph as shown in Figure 1.

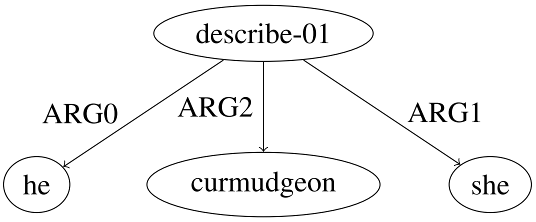


Figure 1: AMR graph for "He described her as a curmudgeon", "His description of her: curmudgeon" and "She was a curmudgeon, according to his description"

However, in practice, things are different. First, there are no known AMR parsers that really distil only the meaning in text. For example, predicates which have interchangeable meaning use different AMR concepts, and there are errors that exist because of the machine learning techniques that are used for learning the parsers from data. Finally, even human annotations do not yield perfect AMRs, as the interannotator agreement reported in the literature for AMR is around 80% (Banarescu et al., 2013).

Second, meaning is often contextual, and it is not fully possible to determine the corresponding AMR parse just by looking at a given sentence. Entity mentions denote different entities in different contexts, and similarly predicates and nouns are ambiguous and depend on context. As such, one cannot expect to use AMR in the transparent way mentioned above to identify paraphrase relations. However, we demonstrate in this paper that AMR can be used in a "softer" way to detect such relations.

Evaluation of AMR parsers is traditionally performed using the Smatch score (Cai and Knight, 2013). However, Damonte et al. (2017) argue that more ad-hoc metrics can be useful for advancing AMR research. Paraphrase detection can be seen

Proceedings of NAACL-HLT 2018, pages 442–452
New Orleans, Louisiana, June 1 - 6, 2018. ©2018 Association for Computational Linguistics

as a further benchmark for AMR parsers, highlighting their ability of abstracting away from syntax and representing the core concepts expressed in the sentence. In order to advance research in AMR and its applications, it is important to have metrics that reflect on the ability of AMR graphs to have impact on subsequent tasks. In this work we therefore use two different AMR parsers, comparing them throughout all experiments.

2 Background

AMRs are rooted, edge labeled, node labeled, directed graphs. They are biased towards the English language and rely on PropBank (Kingsbury and Palmer, 2002) for the definition of the main events in the sentence. Nodes in an AMR graph represent events and concepts, while edges represent the relationships between them. Banarescu et al. (2013) state that AMR are aimed at canonicalizing multiple ways of expressing the same idea, which could be of great assistance to solve the problem of paraphrase detection. However, this goal is not entirely achieved in practice, and it will take long for AMR parsers to mature and achieve such canonicalization. At the moment, for example, even a simple pair of sentences such as “the boy desires the cake” and the “the boy wants the cake” would not have the same canonical form by state-of-the-art AMR parsers.

While some researchers (Fodor, 1975) have doubted the practical possibility of canonicalizing language or finding identical paraphrases in English or otherwise, much work in NLP has been devoted to the problem of paraphrase identification (Mitchell and Lapata, 2010; Baroni and Lenci, 2010; Socher et al., 2011; Guo and Diab, 2012; Ji and Eisenstein, 2013) and more weakly, finding entailment between sentences and phrases (Dagan et al., 2006; Bos and Markert, 2005; Harabagiu and Hickl, 2006; Lewis and Steedman, 2013). In this work, we use the AMRs parsed for given sentences as a mean to extract useful information and train paraphrase detection classifiers on top of them.

2.1 Latent Semantic Analysis

Our work falls under the category of distributional methods for paraphrase detection (Turney and Pantel, 2010; Mihalcea et al., 2006; Mitchell and Lapata, 2010; Guo and Diab, 2012; Ji and Eisenstein, 2013) such as with latent semantic analysis (LSA, Landauer et al., 1998). The main principle behind this approach is to detect semantic similarity through distributional representations for a given sentence and its potential paraphrase, where these representations are compared against each other according to some similarity metric or used as features with a discriminative classification method (Mihalcea et al., 2006; Guo and Diab, 2012; Ji and Eisenstein, 2013).

LSA is indeed one of the main tools in obtaining such distributional representations for the problem of paraphrase detection. Most often, TF-IDF weighting has been used for building the sentence-term matrix, but Ji and Eisenstein (2013) have shown that a significant improvement can be achieved in detecting similarity if one re-weights the sentence-term matrix differently. Indeed, this is one of our main contributions: we build on previous work on LSA for paraphrase detection and propose a technique to re-weight a *sentence-concept* matrix based on the AMR graphs for the given sentences. More details on the use of LSA for paraphrase detection appear in Section 4.

2.2 AMR Parsing

AMR parsing is the task of converting natural language sentences into AMR graphs, which are Directed Acyclic Graphs (DAGs) in all cases except a few rare controversial cases. This task embeds several common NLP problems together, such as named entity recognition, sentential-level coreference resolution, semantic role labeling and word-sense disambiguation. Several parsers for AMR have been recently developed (Flanigan et al., 2014; Wang et al., 2015; Peng et al., 2015; Pust et al., 2015; Goodman et al., 2016; Rao et al., 2015; Vanderwende et al., 2015; Artzi et al., 2015; Barzdins and Gosko, 2016a; Zhou et al., 2016; Damonte et al., 2017; Barzdins and Gosko, 2016b; Konstas et al., 2017). Shared tasks were also organized in order to push forward the state-of-the-art (May, 2016; May and Priyadarshi, 2017).

Meaning representations are usually evaluated based on their compositionality (construction of a representation based on parts of the text in a consistent way), verifiability (ability to check whether a meaning representation is true in a given model of the world), unambiguity (ability to full disambiguate text into the representation in a way that does not leave any ambiguity lingering), inference (the existence of a calculus that can be used to infer whether one meaning representation is logically implied by others) and canonicalization (the ability to map several surface forms, such as paraphrases, into a single unique meaning representation). In this paper, we evaluate AMR on its ability to canonicalize language through its assistance in deciding whether two sentences are paraphrases.

We note that this test is masked by the accuracy of the AMR parsers we use, which indeed do not give always fully correct predictions. These errors in our paraphrase detection due to the accuracy of the AMR parser are different than those which originate in an inherent difficulty of representing paraphrases using AMR because of the limitations of the formalism and the annotation guidelines that AMR follows.

We experiment with two AMR parsers for which a public version is available. The first is JAMR (Flanigan et al., 2014), which is a graph-based approach to AMR parsing. It works by performing two steps on the input sentence: concept identification and relation identification. The former discovers the concept fragments corresponding to span of words in the sentence, while the latter finds the optimal spanning connected subgraph from the concepts identified in the first step. The concept identification step has quadratic complexity and the relation identification step is $O(|V|^2 \log |V|)$, with $|V|$ being the set of nodes in the AMR graph.

The second is AMREager (Damonte et al., 2017), which is a transition-based parser that works by scanning the string left-to-right and building the graph as the scan proceeds. This transition-based system is akin to the dependency parsing transition-system ArcEager of Nivre (2004), only without constraints that ensure that the resulting structure is a tree. In addition, there are operations that make the system create additional non-projective structures by checking after transition step whether siblings should be connected together with an edge. The complexity of AMREager is linear in the length of the sentence. AMREager was extended to other languages (Damonte and Cohen, 2018), and we leave it for future work to test the utility of AMR for paraphrase detection in these languages.

3 Problem Formulation

Let $\mathcal{S}$ be a set of sentences. We are given input data in the form of $(x_1^{(i)}, x_2^{(i)}, b^{(i)})$ for $i \in [n]$ where n is the number of training examples, $x_j^{(i)} \in \mathcal{S}$, $j \in \{1,2\}$ and $b^{(i)} \in \{0,1\}$ is a binary indicator that tells whether $x_1^{(i)}$ is a paraphrase of $x_2^{(i)}$.

The goal is to learn a classifier

$$c \colon \mathcal{S} \times \mathcal{S} \to \{0,1\}$$

that tells for unseen instances whether the pair of sentences given as input are paraphrases of each other. We denote by $[n]$ the set $\{1, \ldots, n\}$.

4 Latent Semantic Analysis for Paraphrase Detection

The first step in our approach is the construction of lower-dimensional representations for the sentences in the training data. We use latent semantic analysis to get the sentence representations, which are then used to detect paraphrases using a classifier. More specifically, given a set of sentences $S = \{x_j^{(i)} \mid j \in \{1,2\}, i \in [n]\}$, we build a sentence-term matrix T such that $T_{k\ell}$ indicates the use of the ℓth word in the kth sentence in S. The number of rows is the number of sentences in the dataset and the number of columns is the vocabulary size. This follows previous work with the use of LSA for paraphrasing (Guo and Diab, 2012; Ji and Eisenstein, 2013).

As a baseline, we experiment with two ways of assigning the values to the matrix:

- $T_{k\ell}$ is the count of the ℓth word in the kth sentence:

$$T_{k\ell} = \text{count}(\ell, k)$$

- $T_{k\ell}$ is the term frequency-inverse document frequency (TF-IDF) for the kth sentence with respect to the ℓth word. TF-IDF is commonly used in Information Retrieval to score words in a document and combines the frequency of the words in a document with the rarity of the term across documents. With TF-IDF, in order to have a high score, concepts must appear in this sentence and not in many others. In that case, we define:

$$T_{k\ell} = \text{count}(\ell, k) \times \frac{n}{\text{csent}(\ell, k)}$$

where $\text{count}(\ell, k)$ gives the count of the ℓth word in the kth sentence and csent is the number of sentences which contain the ℓth word:

$$\mathrm{csent}(\ell, k) = |\{k \in [|S|] \colon \mathrm{count}(k, \ell) > 0\}|.$$

The AMR-based systems of Section 5 build upon this by re-weighting $T_{k\ell}$ with terms depending on the AMRs of the sentences.

For paraphrasing, previous work (Ji and Eisenstein, 2013) has also considered the *transductive* setting (Gammerman et al., 1998), which we also use in our experiments. In the transductive setting, S also includes the sentences on which we expect to perform the final evaluation for the purpose of learning the latent representations. Note that, in this case, the labels $b^{(i)}$ are not used in the process of constructing word representations. In the inductive setting, on the other hand, the sentences in the testing set are not included in training and we project them instead using the LSA projection matrices onto the latent space learned to find their representations.

Once we constructed the matrix T, we perform truncated singular value decomposition (SVD) on it, such that:

$$T \approx U \Sigma V^\top$$

where $U \in \mathbb{R}^{k \times m}$, $V \in \mathbb{R}^{\ell \times m}$ and $\Sigma \in \mathbb{R}^{m \times m}$ is a diagonal matrix of singular values. The final sentence representations are the rows of the U matrix which range over the sentences and have m dimensions.

The output of this process is a function $f \colon \mathcal{S} \to \mathbb{R}^m$ which attaches to each sentence a representation. The idea behind LSA is that this matrix decomposition will make semantically similar sentences to appear close in the latent space, hence alleviating the problem of data sparsity and making it easier to detect when two sentences are paraphrases of each other.

Once we construct the sentence representations from the training data (either in the inductive or the transductive setting) we use the function f to map each pair of sentences from the training data $(x_1^{(i)}, x_2^{(i)})$ to two vectors $f(x_1^{(i)}) + f(x_2^{(i)})$ and $|f(x_1^{(i)}) - f(x_2^{(i)})|$ (where the absolute value is taken coordinate-wise) and then concatenate them into a feature vector $\phi(x_1^{(i)}, x_2^{(i)})$, which is then used as input to a support vector machine (SVM) classifier (Ji and Eisenstein, 2013).[1]

5 Abstract Meaning Representation Features

The main hypothesis tested in this work is that AMR can be useful in deciding whether two sentences are paraphrases of each other. We investigate two ways to use AMR information to better inform the classifier: similarity-based and LSA-based.

5.1 Graph Similarity and Bag of AMR Concepts

An obvious way to use AMR information is to just compute the similarity between the two graphs and use the score as an additional feature. As a score we use Smatch, which computes the overlap in terms of recall, precision and F-score between two unaligned graphs by finding the alignments between the graphs that maximizes the overlap. The alignment step is necessary because in AMR multiple nodes can have the same labels and arbitrary variable names are used to distinguish between them. Smatch is the standard metric to evaluate the overlap between AMR graphs. The score returned by Smatch is used as a single additional feature for the SVM.

The amount of overlap in the AMR nodes of the two graphs can be a good indicator of whether the sentences are paraphrases of each other. To test this hypothesis, we extract the unordered sets of AMR nodes and use the Jaccard similarity coefficient as a feature. This is directly related to the concept identification step of the AMR parsing process, which is concerned with generating and labeling the nodes of the AMR graph. Concept identification is arguably one of the most challenging part of AMR parsing as the mapping between word spans and AMR nodes is not trivial (Werling et al., 2015). It is often considered as the first stage in the AMR parsing pipeline and it is therefore reasonable to attempt using its intermediate results. We choose Jaccard as a metric for bag of concepts overlap following previous work in paraphrase detection (Achananuparp et al., 2008; Be-

[1] We note that while the NLP community has largely switched to the use of neural networks for classification problems, in our case support vector machines prove to be a simpler and more efficient solution. They also tend to generalize better than neural networks, as the number of features we use is not large.

rant and Liang, 2014).

We note that while this approach of using AMR to detect paraphrase may sound plausible, it does not perform very well. As such, we compare and contrast this as an AMR baseline with the approach that makes use of PageRank with TF-IDF reweighting for LSA, as described next.

5.2 PageRank and TF-IDF Reweighting for LSA

The main idea is to re-weight the LSA sentence-term matrix T (Section 4) according to a probability distribution over the AMR nodes, which we accomplish by means of PageRank (Page et al., 1999). The utility of re-weighting terms in the sentence-term matrix has been previously proved (Turney and Pantel, 2010). PageRank is a method, originally developed for web pages, for ranking nodes in a graph according to their impact on other nodes. The algorithm works iteratively by adjusting at each iteration the score of each node based on the number and scores of nearby nodes that is connected to it, until convergence. Prior to applying PageRank, we merge the two graphs by collapsing the concepts in the two graphs that have the same labels, similarly to Liu et al. (2015), as shown in Figure 2. We then compute the PageRank score for each node in the merged graph and multiply them by the corresponding frequency count of that concept in the sentence-term matrix. The graph merging step is necessary in order to ensure that overlapping concepts obtain high PageRank scores. The PageRank step applied to the merged graph ensures that this importance propagates to nearby nodes.

For a given graph $G = (V, E)$, PageRank takes as input a list of edges between nodes:

$$E = \{(n_i, m_i)\}, \forall i = 0, \ldots, n$$
$$n = |E|$$

and outputs a PageRank score for each node by solving the following equations with respect to $\mathrm{PG}(\cdot)$:

$$\mathrm{PG}(n) = \sum_{m \in I(n)} \frac{\mathrm{PG}(m)}{l(m)}$$

where $I(n)$ are the input edges to node n and $l(m)$ is the number of edges coming out of m.

For each concept of the merged AMR graph, we compute $T_{k\ell}$, the weight for the LSA matrix introduced in Section 4, as follows:

$$T_{k\ell} = \mathrm{PG}(l, k) \times \mathrm{count}(l, k)$$

where $\mathrm{PG}(l, k)$ is the PageRank of ℓth concept for the kth sentence.

As a baseline for the PageRank system, the TF-IDF re-weighting scheme, as described in Section 4, is also used to re-weight the AMR concepts.

6 Experiments

We now describe the experiments that we devised to discover whether AMR is useful for paraphrase detection. For AMR parsing, we used the JAMR[2] version published for SemEval 2016 (Flanigan et al., 2016), reporting 0.67 Smatch score on LDC2015E86 and the first and only version available for AMREager,[3] obtaining 0.64 Smatch score on the same dataset. First, we discuss experiments where the AMRs are used as a mean to extract additional sparse features for a SVM classifier. Then we turn to LSA to construct a representation of the sentence based on the reweighting on the AMR nodes achieved through either PageRank or TF-IDF. Results show how the latter, which builds on state-of-the-art systems for this task, is a much more promising approach. Finally, we analyze how performance changes as a function of the number of dimensions used in the truncated matrix.

For evaluation, we use the Microsoft Research Paraphrase Corpus (Dolan et al., 2004). We use 70% of the dataset as training data and 30% as a test set. The total number of sentence pairs in the corpus is 5,801.

6.1 Graph Similarity and Bag of AMR Concepts

The Bag of words (BOW) baseline consists of a SVM that takes into account one single feature: the Jaccard score between the BOW representations for the two sentences, i.e., one-hot vectors indicating whether each word in the vocabulary is used or not. The use of the single Jaccard feature means that for the linear kernel we just learn a threshold on the score.

We note that the addition of the similarity-based features does not suffice to outperform the BOW baseline, as described in Table 1. Unlike Smatch, the bag of concepts feature does not need to find a, possibly wrong, alignment between the two graphs

[2] JAMR is available from https://github.com/jflanigan/jamr.

[3] AMREager is available from http://cohort.inf.ed.ac.uk/amreager.html.

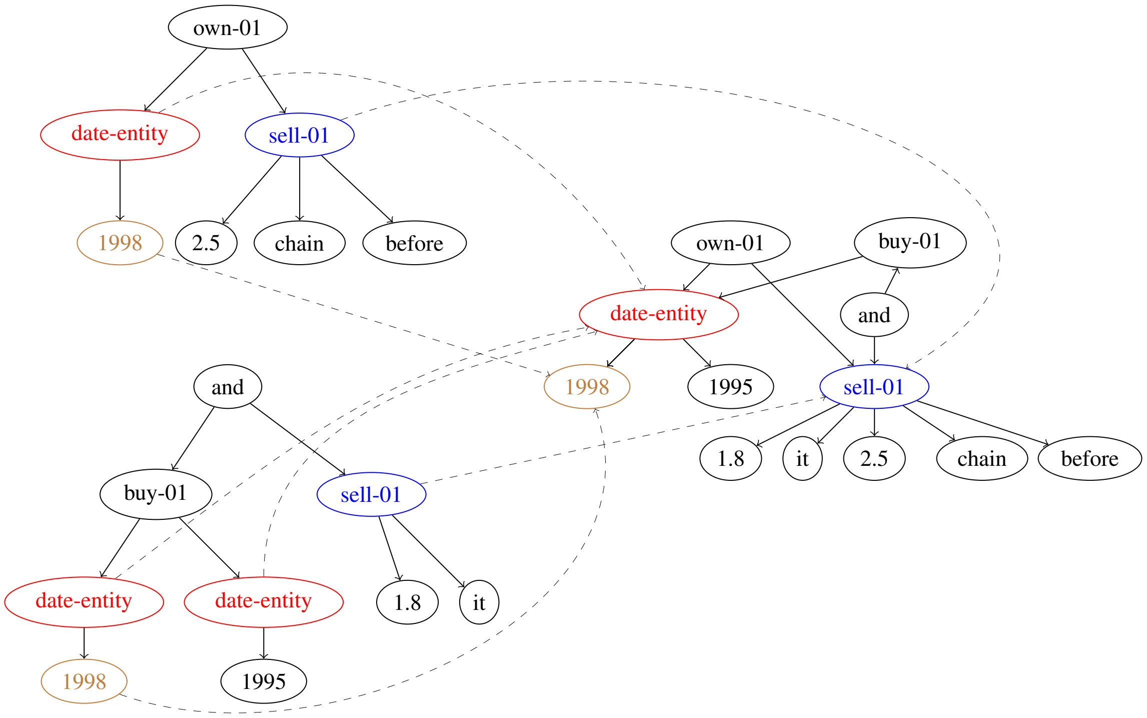


Figure 2: Visualization of the graph merging procedure for the sentence *Yucaipa owned Dominick's before selling the chain to Safeway in 1998 for $2.5 billion.* (above) and *Yucaipa bought Dominick's in 1995 for $693 million and sold it to Safeway for $1.8 billion in 1998.* (below). The "date-entity", "sell-01" and "1998" nodes in the two AMR graphs on the left are merged in the resulting graph on the right.

because it considers the node labels only. Interestingly, the addition of the bag of concepts feature is beneficial only for AMREager. It is indeed worth noting the different behaviors of the two parsers: when using the Smatch score only, JAMR reports slightly higher numbers than AMREager. However, when using the bag of concepts features too, AMREager is considerably better than JAMR, which is unexpected as the concept identification performance of the two parsers is reported to be identical (Damonte et al., 2017).

There is also some variability with the kernel used for the SVM classifier. The polynomial kernel does consistently better than the RBF and linear kernel. This means that a low-level interaction between the sentence representations does exist (when trying to determine whether they are paraphrases), but a higher order interaction, such as implied with RBF, is not necessary to be modeled.

6.2 PageRank and TF-IDF Reweighting for LSA

We now turn to experiments involving LSA as a mean to represent the candidate paraphrases. In this set of experiments, the baseline consists of using TF-IDF to weight the bag of words in the sentence-term matrix.

We first try to replace the bag of words with the bag of concepts from the AMR graphs, also re-weighted by TF-IDF. Then, we also replace the TF-IDF with PageRank as it is more appropriate to re-weight graph structures than TF-IDF. We report experiments for both inductive setting and transductive setting (Table 3). Our first finding is that, regardless of the parser, AMR is very helpful in the tranductive setting while it is harmful in the inductive setting. When using bag of words, it is easy to project sentences of the test set into the latent space learned on the training set only. However, our experiments indicate that this is not as easy with the AMR concepts produced by the two parsers. On the other hand, when the latent space is learned using also the sentences in the test set, the abstractive power of AMRs is helpful for this task. In the inductive setting, PageRank fails to improve over the TF-IDF scheme and neither of them outperform the BOW baseline. AMREager outperforms JAMR in this case. In the transduc-

		kernel	acc.	P	R	F$_1$
Smatch + BOC	AMREager	linear	65.8	81.8	62.9	71.1
		poly	**68.5**	75.4	78.3	76.8
		rbf	65.0	84.0	58.8	69.2
	JAMR	linear	63.4	79.4	61.0	69.0
		poly	64.9	76.0	69.3	72.5
		rbf	61.8	82.0	55.3	65.9
Smatch	AMREager	linear	63.8	79.4	61.7	69.5
		poly	67.8	72.0	84.7	77.8
		rbf	61.6	81.2	55.3	65.8
	JAMR	linear	63.5	77.8	63.5	69.9
		poly	68.1	71.6	**86.5**	78.3
		rbf	62.0	0.80	57.4	66.9
	BOW	linear	68.1	**84.1**	64.3	72.9
		poly	72.7	77.8	82.7	**80.2**
		rbf	68.1	**84.1**	64.3	72.9

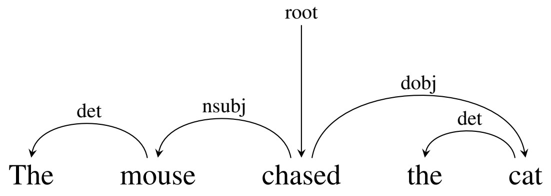
Table 1: Baseline results for paraphrase detection with AMR and with bag-of-words (BOW). "linear," "poly" and "rbf" denote the kernel which is used with a support vector machine classifier. "Smatch" denotes the use of the additional graph similarity feature and "BOC" the use of the additional Jaccard score on the bag of concept. Best result in each column is in bold.

tive case, the AMRs provided by JAMR are helpful with both TF-IDF and PageRank, while the graphs provided by AMREager give good results only for the PageRank scheme. The best result is achieved with JAMR, PageRank and a linear kernel for the SVM classifier.

We wanted to test in our experiments whether the same gains that are achieved with AMR parsing can also be achieved with just a syntactic parser. To test that, we parsed the paraphrase dataset with a dependency parser and reduced the syntactic parse trees to AMR graphs (meaning, we represented the dependency trees as graphs by representing each word as a node and labeled dependency relations as edges). Figure 3 gives an example of such conversion. As can be seen, the AMR-like representation for the dependency trees retains words such as determiners ("the"). It also uses a different set of relations, as reflected by the edge labels that the dependency parser returns.

We chose to do this reduction instead of directly building a classifier that makes use of the dependency trees to ensure we are conducting a controlled experiment in which we precisely compare the use of syntax for paraphrase against the use of semantics. Once the syntactic trees are converted

(a)



(b)

```
(c / chased
    nsubj (m / mouse
        det (t1 / The))
    dobj (c / cat
        det (t2 / the))
)
```

Figure 3: An example of a dependency tree (a) converted to an AMR graph (b).

to AMR graphs, the same code is used to run the experiments as in the case of AMR parsing, with both the PageRank and TF-IDF reweighting settings. We used the dependency parser from the Stanford CoreNLP (Manning et al., 2014). The results are given in Table 3, under "dep." As can be seen, these results lag behind the bag-of-words model in the inductive case and the AMR models in the transductive case. This could be attributed to AMR parsers better abstracting away from the surface form than dependency parsers.

System	acc.	F$_1$
Most common class	66.5	79.9
Mitchell and Lapata (2010)	73.0	82.3
Baroni and Lenci (2010)	73.5	82.2
Socher et al. (2011)	76.8	83.6
Guo and Diab (2012)	71.5	NR
Ji and Eisenstein (2013) (ind.)	80.0	85.4
Ji and Eisenstein (2013) (trans.)	80.4	86.0
Our paper (inductive)	68.7	80.9
Our paper (transductive)	86.6	90.0

Table 2: Comparison of our results with previous work ("NR" stands for "not reported"). All work mentioned above was done in the inductive setting, except for Ji and Eisenstein (2013), which, like us, was done in both settings.

6.3 Dimensionality of the Truncated Matrix

Figure 4 shows how performance changes as a function of the number of dimensions used in the

			inductive				transductive			
		kernel	acc.	P	R	F_1	acc.	P	R	F_1
PageRank	AMREager	linear	68.7	73.1	84.0	78.2	79.4	79.9	92.6	85.7
		poly	67.6	67.9	97.6	80.1	67.0	67.3	**98.6**	80.0
		rbf	68.0	75.0	78.1	76.5	79.6	84.8	84.5	84.7
	JAMR	linear	59.3	70.4	67.4	68.9	**86.6**	88.3	92.0	**90.0**
		poly	66.9	67.6	96.7	79.6	80.0	69.6	98.1	81.5
		rbf	67.4	73.5	79.9	76.6	**86.6**	**90.4**	89.5	89.9
	dep.	linear	62.4	71.6	72.5	72.1	79.0	83.2	85.0	84.1
		poly	68.3	69.0	95.3	80.1	74.0	77.6	85.1	81.2
		rbf	68.8	71.0	90.0	79.4	77.0	89.4	73.9	80.9
TFIDF	AMREager	linear	68.9	73.4	83.7	78.2	73.0	75.8	87.1	81.1
		poly	68.7	68.7	98.2	80.9	68.0	68.2	98.5	80.6
		rbf	68.3	77.9	73.4	75.6	72.0	81.2	75.6	78.3
	JAMR	linear	57.6	68.8	66.8	67.8	82.0	84.7	88.5	86.5
		poly	67.4	67.8	97.4	79.9	82.0	84.7	88.5	86.5
		rbf	69.1	76.7	77.2	76.8	85.0	88.0	88.9	88.4
	dep.	linear	70.3	74.2	85.1	79.3	69.0	73.8	82.8	78.1
		poly	68.8	68.7	97.8	80.7	68.0	68.2	98.2	80.5
		rbf	**70.6**	79.1	76.0	77.5	70.0	79.3	75.5	77.4
	BOW	linear	71.9	75.4	86.0	80.4	73.0	75.8	87.7	81.3
		poly	70.5	69.8	**98.3**	**81.6**	71.0	69.9	98.1	81.7
		rbf	70.5	**81.3**	72.5	76.6	73.0	82.5	75.7	79.0

Table 3: LSA experiments in the inductive and transductive settings, with two different reweighting schema: "PageRank" and "TF-IDF". "linear," "poly" and "rbf" denote the kernel for the SVM. "dep." denotes the use of syntactic parsing instead of semantic parsing.

truncated matrix U (Section 4). More specifically, on the x axis of the plots we have m/l, where m is the number of columns in the truncated matrix and l the number of words in the vocabulary. The plot shows that the performance stays stable for inductive inference. With transductive inference, however, performance peaks when m is very close to the vocabulary size. This shows that, in order to achieve good results, it is not necessary to remove a large number of columns from the original sentence-term matrix. The plot gives us more evidence on how the inductive setting is not ideal for the AMR-based approach. For the TF-IDF reweighting, the systems that show a considerably different behavior are JAMR with linear and RBF kernels, where we show clear peaks for the transductive case. For PageRank also the AMREager systems with linear and RBF kernel follow this trend. In general the polynomial kernel is the one less affected by this variable.

Table 2 shows that our best result for the transductive case, which we obtain with JAMR and PageRank, outperforms the current state of the art for paraphrase detection in the transductive setting. This is not true for the inductive case, proving the preference of the AMR-based LSA approach for the former setting.

7 Conclusion

We described an approach to incorporate an AMR parser output into the detection of paraphrases. Our method works by merging two graphs that need to be tested for a paraphrase relation, and then re-weighting a sentence-term matrix by the PageRank values of the nodes in the merged graph. We find that our method gives significant improvements over state of the art in paraphrase detection in the transductive setting, showing that AMR is indeed helpful for this task. We further show that the inductive settings is instead not ideal for this type of approach.

We are encouraged by the results, and believe that paraphrase detection can also be used as a proxy test for the performance of an AMR parser: if an AMR parser is close to canonicalizing language, it should be of significant help in detecting

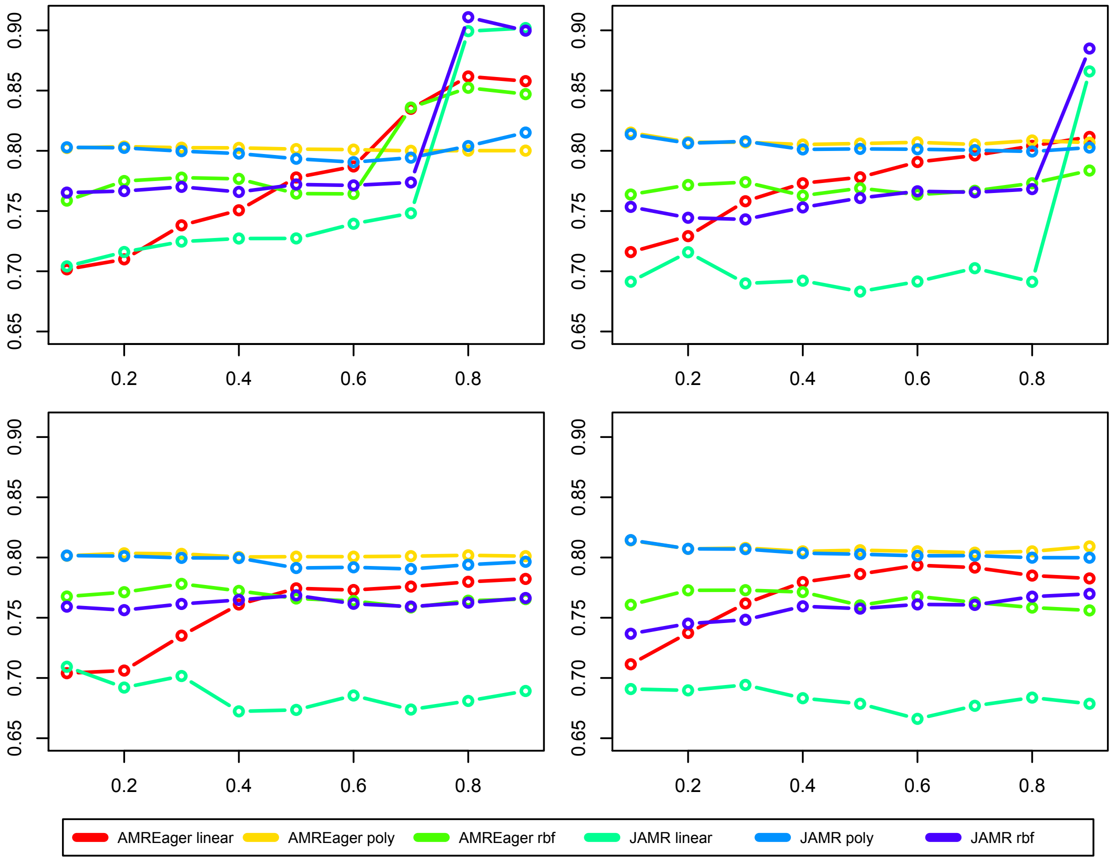


Figure 4: Plots of F_1 measure as a function of m/l, where m is the number of columns in the truncated matrix and l the number of words in the vocabulary. The top plots are in the transductive setting (with the left using PageRank and the right using TF-IDF weighting) while the bottom plots are in the inductive setting.

paraphrase relations. In our experiments, the overall best result was achieved by JAMR. More generally, our results show that JAMR has been more helpful in the transductive setting and in the first set of experiment when using the Smatch score only, while AMREager wins the comparison in the inductive case as well as in the first set of experiments when using both the Smatch score and the bag of concepts score as additional features.

Acknowledgments

The authors would like to thank the three anonymous reviewers for their helpful comments. This research was supported by a grant from Bloomberg, a grant from Huawei Technologies and by the EU H2020 project SUMMA, under grant agreement 688139.

References

Palakorn Achananuparp, Xiaohua Hu, and Xiajiong Shen. 2008. The evaluation of sentence similarity measures. In *Proceedings of DaWaK*.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. *Proceedings of EMNLP* .

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. *Linguistic Annotation Workshop* .

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics* 36(4):673–721.

Guntis Barzdins and Didzis Gosko. 2016a. RIGA at SemEval-2016 task 8: Impact of smatch extensions and character-level neural translation on AMR parsing accuracy. *arXiv preprint arXiv:1604.01278* .

Guntis Barzdins and DIdzis Gosko. 2016b. Riga: Impact of smatch extensions and character-level neural translation on AMR parsing accuracy. *Proceedings of SemEval* .

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of ACL*.

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of HLT/EMNLP*.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. *Proceedings of ACL* .

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, Springer, pages 177–190.

Marco Damonte and Shay B. Cohen. 2018. Cross-lingual abstract meaning representation parsing. *Proceedings of NAACL* .

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of EACL*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of COLING*.

Jeffrey Flanigan, Chris Dyer, Noah A Smith, and Jaime Carbonell. 2016. CMU at SemEval-2016 task 8: Graph-based AMR parsing with infinite ramp loss. *Proceedings of SemEval* .

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

Jerry A Fodor. 1975. *The language of thought*, volume 5. Harvard University Press.

Alexander Gammerman, Volodya Vovk, and Vladimir Vapnik. 1998. Learning by transduction. In *Proceedings of AUAI*.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Noise reduction and targeted exploration in imitation learning for abstract meaning representation parsing. *Proceedings of ACL* .

Weiwei Guo and Mona Diab. 2012. Modeling sentences in the latent space. In *Proceedings of ACL*.

Sanda Harabagiu and Andrew Hickl. 2006. Methods for using textual entailment in open-domain question answering. In *Proceedings of COLING/ACL*.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *Proceedings of EMNLP*.

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. *Proceedings of LREC* .

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. *arXiv preprint arXiv:1704.08381* .

Thomas K. Landauer, Peter W. Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse Processes* 25:259–284.

Mike Lewis and Mark Steedman. 2013. Combining distributional and logical semantics. *Transactions of the Association for Computational Linguistics* 1:179–192.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A Smith. 2015. Toward abstractive summarization using semantic representations. *Proceedings of NAACL* .

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of ACL*.

Jonathan May. 2016. Semeval-2016 task 8: Meaning representation parsing. In *Proceedings of SemEval*.

Jonathan May and Jay Priyadarshi. 2017. Semeval-2017 task 9: Abstract meaning representation parsing and generation. In *Proceedings of SemEval*.

Rada Mihalcea, Courtney Corley, Carlo Strapparava, et al. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of AAAI*.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive science* 34(8):1388–1429.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. *Workshop on Incremental Parsing: Bringing Engineering and Cognition Together* .

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for AMR parsing. *Proceedings of CoNLL* .

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Using syntax-based machine translation to parse english into abstract meaning representation. *arXiv preprint arXiv:1504.06665* .

Sudh Rao, Yogarshi Vyas, Hal Daume III, and Philip Resnik. 2015. Parser for abstract meaning representation using learning to search. *arXiv:1510.07586* .

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP*.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research* 37:141–188.

Lucy Vanderwende, Arul Menezes, and Chris Quirk. 2015. An AMR parser for English, French, German, Spanish and Japanese and a new AMR-annotated corpus. *Proceedings of NAACL-HLT* .

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. Boosting transition-based AMR parsing with refined actions and auxiliary analyzers. *Proceedings of ACL* .

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

Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang QU, Ran Li, and Yanhui Gu. 2016. AMR parsing with an incremental joint model. In *Proceedings of EMNLP*.