Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Extracting Biomolecular Interactions Using Semantic Parsing of Biomedical Text

Sahil Garg, Aram Galstyan, Ulf Hermjakob, and **Daniel Marcu**
USC Information Sciences Institute
Marina del Rey, CA 90292
{sahil, galstyan, ulf, marcu}@isi.edu

Abstract

We advance the state of the art in biomolecular interaction extraction with three contributions: (i) We show that deep, Abstract Meaning Representations (AMR) significantly improve the accuracy of a biomolecular interaction extraction system when compared to a baseline that relies solely on surface- and syntax-based features; (ii) In contrast with previous approaches that infer relations on a sentence-by-sentence basis, we expand our framework to enable consistent predictions over sets of sentences (documents); (iii) We further modify and expand a graph kernel learning framework to enable concurrent exploitation of automatically induced AMR (semantic) and dependency structure (syntactic) representations. Our experiments show that our approach yields interaction extraction systems that are more robust in environments where there is a significant mismatch between training and test conditions.

1 Introduction

Recent advances in genomics and proteomics have significantly accelerated the rate of uncovering and accumulating new biomedical knowledge. Most of this knowledge is available only via scientific publications, which necessitates the development of automated and semi-automated tools for extracting useful biomedical information from unstructured text. In particular, there has been a significant body of research on identifying biological entities (proteins, genes, chemical compounds) and interactions between those entities from bio-medical papers (Krallinger et al. 2008; Hakenberg et al. 2008; Tikk et al. 2010; Bunescu et al. 2005). Despite the recent progress, current methods for biomedical knowledge extraction suffer from a number of important shortcomings. First of all, existing methods rely heavily on shallow analysis techniques that severely limit their scope. For instance, most existing approaches focus on whether there is an interaction between a pair of proteins while ignoring the interaction types (Airola et al. 2008; Mooney and Bunescu 2005), whereas other more advanced approaches cover only a small subset of all possible interaction types (Hunter et al. 2008; McDonald et al. 2005; Demir et al. 2010). Second, most existing methods focus on single-sentence extraction, which makes them very susceptible to noise. And finally, owing to the enormous diversity of research topics in biomedical literature and the high cost of data annotation, there is often significant mismatch between training and testing corpora, which reflects poorly on generalization ability of existing methods (Tikk et al. 2010).

In this paper, we present a novel algorithm for extracting biomolecular interactions from unstructured text that addresses the above challenges. Contrary to the previous works, the extraction task considered here is less restricted and spans a much more diverse corpus of biomedical articles. These more realistic settings present some important technical problems for which we provide explicit solutions.

Our specific contributions are as follows:

- We propose a graph-kernel based algorithm for extracting biomolecular interactions from Abstract Meaning Representation, or AMR. To the best of our knowledge, this is the first attempt of using deep semantic parsing for biomedical knowledge extraction task.
- We provide a multi-sentence generalization of the algorithm by defining *Graph Distribution Kernels* (GDK), which enables us to perform document-level extraction.
- We suggest a hybrid extraction method that utilizes both AMRs and syntactic parses given by Stanford Dependency Graphs (SDGs). Toward this goal, we develop a linear algebraic formulation for *learning vector space embedding of edge labels* in AMRs and SDGs to define similarity measures between AMRs and SDGs.

We conduct an exhaustive empirical evaluation of the proposed extraction system on 45+ research articles on cancer (approximately 3k sentences), containing approximately 20,000 positive-negative labeled biomolecular interactions[1]. Our results indicate that the joint extraction method that leverages both AMRs and SDGs parses significantly improves the extraction accuracy, and is more robust to mismatch between training and test conditions.

2 Problem Statement

Consider the sentence *"As a result, mutant Ras proteins accumulate with elevated GTP-bound proportion"*, which de-

[1]The code and the data are available at https://github.com/sgarg87/big_mech_isi_gg

State change	inhibit, phosphorylate, signal, activate, transcript, regulate, apoptose, express, translocate, degrade, carboxymethylate, depalmitoylate, acetylate, nitrosylate, farnesylate, methylate, glycosylate, hydroxylate, ribosylate, sumoylate, ubiquitinate.
Bind	bind, heterodimerize, homodimerize, dissociate.

Table 1: Interaction type examples

scribes a "binding" interaction between a protein "Ras" and a small-molecule "GTP". We want to extract this interaction.

In our representation, which is motivated by BioPAX (Demir et al. 2010), an *interaction* refers to either i) an entity effecting state change of another entity; or ii) an entity binding/dissociating with another entity to form/break a complex while, optionally, also influenced by a third entity. An entity can be of any type existent in a bio pathway, such as protein, complex, enzyme, etc, although here we refer to an entity of all valid types simply as a protein. The change in state of an entity or binding type is simply termed as "interaction type" in this work. In some cases, entities are capable of changing their state on their own or bind to an instance of its own (self-interaction). Such special cases are also included. Some examples of interaction types are shown in Table 1.

Below we describe our approach for extracting above-defined interactions from natural language parses of sentences in a research document.

3 Extracting Interactions from an AMR

3.1 AMR Biomedical Corpus

Abstract Meaning Representation, or AMR, is a semantic annotation of single/multiple sentences (Banarescu et al. 2013). In contrast to syntactic parses, in AMR, entities are identified, typed and their semantic roles are annotated. AMR maps different syntactic constructs to same conceptual term. For instance, "binding", "bound", "bond" correspond to the same concept "bind-01". Because one AMR representation subsumes multiple syntactic representations, we hypothesize that AMRs have higher utility for extracting biomedical interactions.

We trained an English-to-AMR parser (Pust et al. 2015) on two manually annotated corpora: i) a corpus of *17k general domain sentences* including newswire and web text as published by the Linguistic Data Consortium; and ii) *3.4k systems biology sentences*, including in-domain PubMed-Central papers and the BEL BioCreative corpus. As part of building the bio-specific AMR corpus, we extended the PropBank-based framesets used in AMR by 45 bio-specific frames such as "phosphorylate-01", "immunoblot-01" and extended the list of AMR standard named entities by 15 types such as "enzyme", "pathway". It is important to note that these extensions are not specific to biomolecular interactions, and cover more general cancer biology concepts.

3.2 Extracting Interactions

Fig. 1 depicts a manual AMR annotation of a sentence, which has two highlighted entity nodes with labels "RAS"

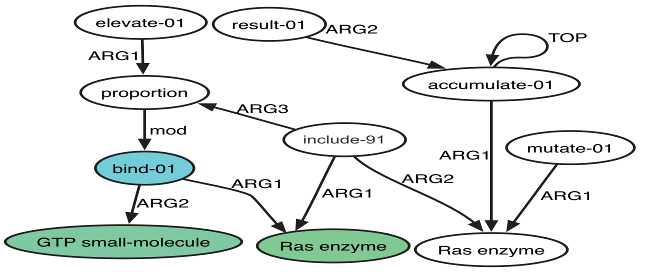


Figure 1: AMR of text "*As a result, mutant Ras proteins accumulate with elevated GTP-bound proportion.*"; interaction "*Ras binds to GTP*" is extracted from the colored sub-graph.

and "GTP". These nodes also have entity type annotations, "enzyme" and "small-molecule" respectively; the concept node with a node label "bind-01" corresponds to an interaction type "binding" (from the "GTP-bound" in the text). The interaction "RAS-binds-GTP" is extracted from the highlighted subgraph under the "bind" node. In the subgraph, relationship between the interaction node "bind-01" and the entity nodes, "Ras" and "GTP", is defined through two edges with edge labels "ARG1" and "ARG2" respectively. Additionally, in the subgraph, we assign roles "interaction-type", "protein", "protein" to the nodes "bind-01", "Ras", "GTP" respectively (roles presented with different colors in the subgraph).

Given an AMR graph, as in Fig. 1, we first identify potential entity nodes (proteins, molecules, etc) and interaction nodes (bind, activate, etc). Next, we consider all permutations to generate a set of potential interactions according to the format defined above. For each candidate interaction, we extract the corresponding shortest path subgraph. We then project the subgraph to a tree structure [2] with the interaction node as root and also possibly the protein nodes (entities involved in the interaction) as leaves.

Our training set consists of tuples $\{G_i^a, I_i, l_i\}_{i=1}^n$, where G_i^a is an AMR subgraph constructed such that it can represent an extracted candidate interaction I_i with interaction node as root and proteins nodes as leaves typically; and $l = \{0, 1\}$ is a binary label indicating whether this subgraph contains I_i or not. Given a training set, and a new sample AMR subgraph G_*^a for interaction I_*, we would like to infer whether I_* is valid or not. We address this problem by developing a graph-kernel based approach.

3.3 Semantic Embedding Based Graph Kernel

We propose an extension of the *contiguous subtree kernel* (Zelenko, Aone, and Richardella 2003; Culotta and Sorensen 2004) for mapping the extracted subgraphs (tree structure) to an implicit feature space. Originally, this kernel uses an identity function on two node labels when calculating the similarity between those two nodes. We instead propose to use vector space embedding of the node labels (Clark 2014; Mikolov et al. 2013), and then define a sparse RBF kernel on the node label vectors. Simi-

[2]This can be done via so called inverse edge labels; see (Banarescu et al. 2013, section 3).

lar extensions of convolution kernels have been been suggested previously(Mehdad, Moschitti, and Zanzotto 2010; Srivastava, Hovy, and Hovy 2013).

Consider two graphs G_i and G_j rooted at nodes $G_i.r$ and $G_j.r$, respectively, and let $G_i.c$ and $G_j.c$ be the children nodes of the corresponding root nodes. Then the kernel between G_i and G_j is defined as follows:

$$K(G_i, G_j) = \begin{cases} 0 & \text{if } k(i,j) = 0 \\ k(i,j) + K_c(G_i.c, G_j.c) & \text{otherwise} \end{cases},$$

where $k(i,j) \equiv k(G_i.r, G_j.r)$ is the similarity between the root nodes, whereas $K_c(G_i.c, G_j.c)$ is the recursive part of the kernel that measures the similarity of the children subgraphs. Furthermore, the similarity between root nodes x and y is defined as follows:

$$\begin{aligned} &k(x,y) = k_w(x,y)^2(k_w(x,y)^2 + k_e(x,y) + k_r(x,y)) \\ &k_w(x,y) = \exp((\boldsymbol{w}_x^T \boldsymbol{w}_y - 1)/\beta)((\boldsymbol{w}_x^T \boldsymbol{w}_y - \alpha)/(1-\alpha))_+ \\ &k_e(x,y) = \mathbb{I}(e_x = e_y),\ k_r(x,y) = \mathbb{I}(r_x = r_y)\ . \end{aligned} \tag{1}$$

Here $(\cdot)_+$ denotes the positive part; $\mathbb{I}(\cdot)$ is the indicator function; $\boldsymbol{w}_x, \boldsymbol{w}_y$ are unit vector embeddings of node labels [3]; e_x, e_y represent edge labels (label of an edge from a node's parent to it is the node's edge label); r_x, r_y are roles of nodes (such as protein, catalyst, concept, interaction-type); α is a threshold parameter on the cosine similarity ($\boldsymbol{w}_x^T \boldsymbol{w}_y$) to control sparsity (Gneiting 2002); and β is the bandwidth.

The recursive part of the kernel, K_c, is defined as follows:

$$K_c(G_i.c, G_j.c) = \sum_{\boldsymbol{i},\boldsymbol{j}: l(\boldsymbol{i})=l(\boldsymbol{j})} \lambda^{l(\boldsymbol{i})} \sum_{s=1,\cdots,l(\boldsymbol{i})} K(G_i[\boldsymbol{i}[s]], G_j[\boldsymbol{j}[s]]) \prod_{s=1,\cdots,l(\boldsymbol{i})} k(G_i[\boldsymbol{i}[s]].r, G_j[\boldsymbol{j}[s]].r),$$

where $\boldsymbol{i}, \boldsymbol{j}$ are contiguous children subsequences under the respective root nodes $G_i.r, G_j.r$; $\lambda \in (0,1)$ is a tuning parameter; and $l(\boldsymbol{i})$ is the length of sequence $\boldsymbol{i} = i_1, \cdots, i_l$; $G_i[\boldsymbol{i}[s]]$ is a sub-tree rooted at $\boldsymbol{i}[s]$ index child node of $G_i.r$. Here, we propose to sort children of a node based on the corresponding edge labels. This helps in distinguishing between two mirror image trees.

This extension is a valid kernel function (Zelenko, Aone, and Richardella, Theorem 3, p. 1090). Next, we generalize the dynamic programming approach of Zelenko, Aone, and Richardella for efficient calculation of this extended kernel.

Dynamic programming for computing convolution graph kernel In the convolution kernel presented above, the main computational cost is due to comparison of children sub-sequences. Since different children sub-sequences of a given root node partially overlap with each other, one can use dynamic programming to avoid redundant computations, thus reducing the cost. Toward this goal, we use the following decomposition of the kernel K_c :

$$K_c(G_i.c, G_j.c) = \sum_{p,q} C_{p,q}\ ,$$

[3]Learned using word2vec software (Mikolov et al. 2013) on over one million PubMed articles.

where $C_{p,q}$ refers to the similarity between sub-sequences starting at indices p, q respectively in $G_i.c$ and $G_j.c$.

To calculate $C_{p,q}$ via dynamic programming, let us introduce

$$L_{p,q} = \max_l \left(\prod_{s=0}^{l} k(G_i[\boldsymbol{i}[p+s]].r, G_j[\boldsymbol{j}[q+s]].r) \neq 0 \right).$$

Furthermore, let us denote $k_{p,q} = k(G_i[\boldsymbol{i}[p]].r, G_j[\boldsymbol{j}[q]].r)$, and $K_{p,q} = K(G_i[\boldsymbol{i}[p]], G_j[\boldsymbol{j}[q]])$. We then evaluate $C_{p,q}$ in a recursive manner using the following equations.

$$L_{p,q} = \begin{cases} 0 & \text{if } k_{p,q} = 0 \\ L_{p+1,q+1} + 1 & \text{otherwise} \end{cases} \tag{2}$$

$$C_{p,q} = \begin{cases} 0 & \text{if } k_{p,q} = 0 \\ \frac{\lambda(1-\lambda^{L(p,q)})}{1-\lambda} K_{p,q} k_{p,q} + \lambda C_{p+1,q+1} & \text{otherwise} \end{cases} \tag{3}$$

$$\begin{aligned} L_{m+1,n+1} = 0, L_{m+1,n} = 0, L_{m,n+1} = 0 \\ C_{m+1,n+1} = 0, C_{m+1,n} = 0, C_{m,n+1} = 0\ , \end{aligned} \tag{4}$$

where m, n are number of children under the root nodes $G_i.r$ and $G_j.r$ respectively.

Note that for graphs with cycles, the above dynamic program can be transformed into a linear program.

There are a couple of practical considerations during the kernel computations. First of all, the kernel depends on two tunable parameters λ and α. Intuitively, decreasing λ discounts the contributions of longer child sub-sequences. The parameter α, on the other hand, controls the tradeoff between computational cost and accuracy. Based on some prior tuning we found that our results are not very sensitive to the parameters. In the experiments below we set $\lambda = 0.99$ and $\alpha = 0.4$. Also, consistent with previous studies, we normalize the graph kernel (e.g., kernel similarity $K(G_i, G_j)$ is divided by the normalization term $\sqrt{K(G_i, G_i)K(G_j, G_j)}$) to increase accuracy.

4 Graph Distribution Kernel- GDK

Often an interaction is mentioned more than once in the same research paper, which justifies a document-level extraction, where one combines evidence from multiple sentences. The prevailing approach to document-level extraction is to first perform inference at sentence level, and then combine those inferences using some type of an aggregation function for a final document-level inference (Skounakis and Craven 2003; Bunescu et al. 2006). For instance, in (Bunescu et al. 2006), the inference with the maximum score is chosen. We term this baseline approach as "Maximum Score Inference", or MSI. Here we advocate a different approach, where one uses the evidences from multiple sentences jointly, for a collective inference.

Let us assume an interaction I_m is supported by k_m sentences, and let $\{G_{m1}, \cdots, G_{mk_m}\}$ be the set of relevant AMR subgraphs extracted from those sentences. We can view the elements of this set as samples from some distribution over the graphs, which, with a slight abuse of notation, we denote as $\mathcal{G}_m$. Consider now interactions $I_1, \cdots, I_p$, and let $\mathcal{G}_1, \cdots, \mathcal{G}_p$ be graph distributions representing these interactions.

The graph distribution kernel (GDK), $\mathcal{K}(\mathcal{G}_i, \mathcal{G}_j)$, for a pair $\mathcal{G}_i, \mathcal{G}_j$ is defined as follows:

$$\mathcal{K}(\mathcal{G}_i, \mathcal{G}_j) = \exp(-\mathcal{D}_{mm}(\mathcal{G}_i, \mathcal{G}_j)); \;\; \mathcal{D}_{mm}(\mathcal{G}_i, \mathcal{G}_j) =$$
$$\sum_{r,s=1}^{k_i} \frac{K(G_{ir}, G_{is})}{k_i^2} + \sum_{r,s=1}^{k_j} \frac{K(G_{jr}, G_{js})}{k_j^2} - 2\sum_{r,s=1}^{k_i,k_j} \frac{K(G_{ir}, G_{js})}{k_i k_j}$$

Here $\mathcal{D}_{mm}$ is the *Maximum Mean Discrepancy* (MMD), a valid l_2 norm, between a pair of distributions $\mathcal{G}_i, \mathcal{G}_j$ (Gretton et al. 2012); $K(.,.)$ is the graph kernel defined in Section 3.3 (though, not restricted to this specific kernel). As the term suggests, *maximum mean discrepancy represents the discrepancy between the mean of graph kernel features (features implied by kernels) in samples of distributions $\mathcal{G}_i$ and $\mathcal{G}_j$. Now, since $\mathcal{D}_{mm}$ is the l_2 norm on the mean feature vectors, $\mathcal{K}(\mathcal{G}_p, \mathcal{G}_q)$ is a valid kernel function.*

We note that MMD metric has attracted a considerable attention in the machine learning community recently (Gretton et al. 2012; Kim and Pineau 2013; Pan, Kwok, and Yang 2008; Borgwardt et al. 2006). For our purpose, we prefer using this divergence metric over others (such as KL-D divergence) for the following reasons: i) $\mathcal{D}_{mm}(.,.)$ is a "kernel trick" based formulation, nicely fitting with our settings since we do not have explicit features representation of the graphs but only kernel density on the graph samples. Same is true for KL-D estimation with kernel density method. ii) Empirical estimate of $\mathcal{D}_{mm}(.,.)$ is a valid l_2 norm distance. Therefore, it is straightforward to derive the graph distribution kernel $\mathcal{K}(\mathcal{G}_i, \mathcal{G}_j)$ from $\mathcal{D}_{mm}(\mathcal{G}_i, \mathcal{G}_j)$ using a function such as RBF. This is not true for divergence metrics such as KL-D, Renyi (Sutherland et al. 2012); iii) It is suitable for compactly supported distributions (small number of samples) whereas methods, such as k-nearest neighbor estimation of KL-D, are not suitable if the number of samples in a distribution is too small (Wang, Kulkarni, and Verdú 2009); iv) We have seen the most consistent results in our extraction experiments using this metric as opposed to the others.

For the above mentioned reasons, here we focus on MMD as our primary metric for computing similarities between graph distributions. The proposed GDK framework, however, is very general and not limited to a specific metric. Next, we briefly describe two other metrics that can be used with GDK.

GDK with Kullback-Leibler divergence While MMD represents maximum discrepancy between the mean features of two distributions, the Kullback-Leibler divergence (KL-D) is a more comprehensive (and fundamental) measure of distance between two distributions[4]. For defining kernel $\mathcal{K}_{KL}$ in terms of KL-D, however, we have two challenges. First of all, KL-D is not a symmetric function. This problem can be addressed by using a symmetric version of the distance in the RBF kernel,

$$\mathcal{K}_{KL}(\mathcal{G}_i, \mathcal{G}_j) = \exp(-[\mathcal{D}_{KL}(\mathcal{G}_i||\mathcal{G}_j) + \mathcal{D}_{KL}(\mathcal{G}_j||\mathcal{G}_i)])$$

where $\mathcal{D}_{KL}(\mathcal{G}_i||\mathcal{G}_j)$ is the KL distance of the distribution $\mathcal{G}_i$ w.r.t. the distribution $\mathcal{G}_j$. And second, even the symmetric combination of the divergences is not a valid Euclidian distance. Hence, $\mathcal{K}_{KL}$ is not guaranteed to be a positive semi-definite function. This issue can be dealt in a practical manner as nicely discussed in (Sutherland et al. 2012). Namely, having computed the Gram matrix using $\mathcal{K}_{KL}$, we can project it onto a positive semi-definite one by using linear algebraic techniques, e.g., by discarding negative eigenvalues from the spectrum.

Since we do not know the true divergence, we approximate it with its empirical estimate from the data, $\mathcal{D}_{KL}(\mathcal{G}_i||\mathcal{G}_j) \approx \hat{\mathcal{D}}_{KL}(\mathcal{G}_i||\mathcal{G}_j)$. While there are different approaches for estimating divergences from samples (Wang, Kulkarni, and Verdú 2009), here we use kernel density estimator as shown below:

$$\hat{\mathcal{D}}_{KL}(\mathcal{G}_i||\mathcal{G}_j) = \frac{1}{k_i}\sum_{r=1}^{k_i} \log \frac{\frac{1}{k_i}\sum_{s=1}^{k_i} K(G_{ir}, G_{is})}{\frac{1}{k_j}\sum_{s=1}^{k_j} K(G_{ir}, G_{js})}$$

GDK with cross kernels Another simple way to evaluate similarity between two distributions is to take the mean of cross-kernel similarities between the corresponding two sample sets:

$$\mathcal{K}(\mathcal{G}_i, \mathcal{G}_j) = \sum_{r,s=1}^{k_i,k_j} \frac{K(G_{ir}, G_{js})}{k_i k_j}$$

Note that this metric looks quite similar to the MMD. As demonstrated in our experiments, however, MMD does better, presumably because it accounts for the mean kernel similarity between samples of the same distribution.

Having defined the graph distribution kernel-GDK, $\mathcal{K}(.,.)$, our revised training set consists of tuples $\{\mathcal{G}_i, I_i, l_i\}_{i=1}^n$ with $G_{i1}^a, \cdots, G_{ik_i}^a$ sample sub-graphs in $\mathcal{G}_i$. For inferring an interaction I_*, we evaluate GDK between a test distribution $\mathcal{G}_*$ and the train distributions $\{\mathcal{G}_1, \cdots, \mathcal{G}_n\}$, from their corresponding sample sets. Then, one can apply any "kernel trick" based classifier.

5 Cross Representation Similarity

In the previous section, we proposed a novel algorithm for document-level extraction of interactions from AMRs. Looking forward, we will see in our experiments (Section 6) that AMRs yield better extraction accuracy compared to SDGs. This result suggests that using deep semantic features is very useful for the extraction task. On the other hand, the accuracy of semantic (AMR) parsing is not as good as the accuracy of shallow parsers like SDGs (Pust et al. 2015; Flanigan et al. 2014; Wang, Berant, and Liang 2015; Andreas, Vlachos, and Clark 2013; Chen and Manning 2014). Thus, one can ask whether the joint use of semantic (AMRs)

[4]Recall that the KL divergence between distributions p and q is defined as $\mathcal{D}_{KL}(p||q) = \mathbb{E}_{p(x)}[\log \frac{p(x)}{q(x)}]$

Abstract Meaning Representation

```
(a / activate-01
   :ARG0 (s / protein :name (n1 / name :op1 "RAS"))
   :ARG1 (s / protein :name (n2 / name :op1 "B-RAF"))
```

Stanford Typed Dependency

```
nsubj(activates-2, RAS-1)
root(ROOT-0, activates-2)
acomp(activates-2, B-RAF-3)
```

Figure 2: AMR and SDG parses of "*RAS activates B-RAF.*"

and syntactic (SDGs) parses can improve extraction accuracy further.

There are some intuitive observations that justify the joint approach: i) shallow syntactic parses may be sufficient for correctly extracting a subset of interactions; ii) semantic parsers might make mistakes that are avoidable in syntactic ones. For instance, in machine translation based semantic parsers (Pust et al. 2015; Andreas, Vlachos, and Clark 2013), hallucinating phrasal translations may introduce an interaction/protein in a parse that is non-existent in true semantics; iii) over fit of syntactic/semantic parsers can vary from each other in a test corpus depending upon the data used in their independent trainings.

In this setting, in each evidence sentence, a candidate interaction I_i is represented by a tuple $\Sigma_i = \{G_i^a, G_i^s\}$ of sub-graphs G_i^a and G_i^s which are constructed from AMR and SDG parses of a sentence respectively. Our problem is to classify the interaction jointly on features of both sub-graphs. This can be further extended for the use of multiple evidence sentences. We now argue that the graph-kernel framework outlined above can be applied to this setting as well, with some modifications.

Let Σ_i and Σ_j be two sets of points. To apply the framework above, we need a valid kernel $K(\Sigma_i, \Sigma_j)$ defined on the joint space. One way of defining this kernel would be using similarity measures between AMRs and SDGs separately, and then combining them e.g., via linear combination. However, here we advocate a different approach, where we *flatten* the joint representation. Each candidate interaction is represented as a set of two points in the same space. This projection is a valid operation as long as we have a similarity measure between G_i^a and G_i^s (correlation between the two original dimensions). This is rather problematic since AMRs and SDGs have non-overlapping edge labels (although the space of node labels of both representations coincide). To address this issue, for inducing this similarity measure, we next develop our approach for edge-label vector space embedding.

Let us understand what we mean by vector space embedding of edge-labels. In Fig. 2, we have an AMR and a SDG parse of "RAS activates B-RAF". "ARG0" in the AMR and "nsubj" in SDG are conveying that "RAS" is a catalyst of the interaction "activation"; "ARG1" and "acomp" are meaning that "B-RAF" is activated. In this sentence, "ARG0" and "nsubj" are playing the same role though their higher dimensional roles, across a diversity set of sentences, would vary. Along these lines, we propose to *embed these high dimensional roles in a vector space*, termed as "edge label vectors".

5.1 Consistency Equations for Edge Vectors

We now describe our unsupervised algorithm that learns vector space embedding of edge labels. The algorithm works by imposing linear consistency conditions on the word vector embeddings of node labels. While we describe the algorithm using AMRs, it is directly applicable to SDGs as well.

Linear algebraic formulation In our formulation, we first learn subspace embedding of edge labels (edge label matrices) and then transform it into vectors by flattening. Let us see the AMR in Fig. 2 again. We already have word vectors embedding for terms "activate", "RAS", "B-RAF", denoted as $\boldsymbol{w}_{activate}, \boldsymbol{w}_{ras}, \boldsymbol{w}_{braf}$ respectively; a word vector $\boldsymbol{w}_i \in \mathbb{R}^{m \times 1}$. Let embedding for edge labels "ARG0" and "ARG1" be $\boldsymbol{A}_{arg0}, \boldsymbol{A}_{arg1}$; $\boldsymbol{A}_i \in \mathbb{R}^{m \times m}$. In this AMR, we define following linear algebraic equations.

$$\boldsymbol{w}_{activate} = \boldsymbol{A}_{arg0}^T \boldsymbol{w}_{ras}, \boldsymbol{w}_{activate} = \boldsymbol{A}_{arg1}^T \boldsymbol{w}_{braf}$$
$$\boldsymbol{A}_{arg0}^T \boldsymbol{w}_{ras} = \boldsymbol{A}_{arg1}^T \boldsymbol{w}_{braf}$$

The edge label matrices $\boldsymbol{A}_{arg0}^T$, $\boldsymbol{A}_{arg1}^T$ are linear transformations on the word vectors $\boldsymbol{w}_{ras}, \boldsymbol{w}_{braf}$, establishing linear consistencies between the word vectors along the edges. One can define such a set of equations in each parent-children nodes sub-graph in a given set of manually annotated AMRs (and so applies to SDGs independent of AMRs). Along these lines, for a pair of edge labels i, j in AMRs, we have generalized equations as below.

$$\boldsymbol{Y}_i = \boldsymbol{X}_i \boldsymbol{A}_i,\ \boldsymbol{Y}_j = \boldsymbol{X}_j \boldsymbol{A}_j,\ \boldsymbol{Z}_i^{ij} \boldsymbol{A}_i = \boldsymbol{Z}_j^{ij} \boldsymbol{A}_j$$

Here $\boldsymbol{A}_i, \boldsymbol{A}_j$ are edge labels matrices. Considering n_i occurrences of edge labels i, we correspondingly have word vectors from the n_i child node labels stacked as rows in matrix $\boldsymbol{X}_i \in \mathbb{R}^{n_i \times m}$; and $\boldsymbol{Y}_i \in \mathbb{R}^{n_i \times m}$ from the parent node labels. There would be a subset of instances, $n_{ij} <= n_i, n_j$ where edge labels i and j has same parent node (occurrence of pairwise relationship between i and j). This gives $\boldsymbol{Z}_i^{ij} \in \mathbb{R}^{n_{ij} \times m}$ and $\boldsymbol{Z}_j^{ij} \in \mathbb{R}^{n_{ij} \times m}$, subsets of word vectors in $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$ respectively (along rows). Along these lines, neighborhood of edge label i is defined to be: $\mathcal{N}(i)$: $j \in \mathcal{N}(i)\ s.t.\ n_{ij} > 0$. From the above pairwise linear consistencies, we derive linear dependencies of an $\boldsymbol{A}_i$ with its neighbors $\boldsymbol{A}_j : j \in \mathcal{N}(i)$, while also applying least square approximation.

$$\boldsymbol{X}_i^T \boldsymbol{Y}_i + \sum_{j \in \mathcal{N}(i)} \boldsymbol{Z}_i^{ij^T} \boldsymbol{Z}_j^{ij} \boldsymbol{A}_j = (\boldsymbol{X}_i^T \boldsymbol{X}_i + \sum_{j \in \mathcal{N}(i)} \boldsymbol{Z}_i^{ij^T} \boldsymbol{Z}_i^{ij}) \boldsymbol{A}_i$$

Exploiting the block structure in the linear program, we propose an algorithm that is a variant of "Gauss-Seidel" method (Demmel 1997; Niethammer, De Pillis, and Varga 1984).

Algorithm 1 *(a) Initialize:*

$$\boldsymbol{A}_i^{(0)} = (\boldsymbol{X}_i^T \boldsymbol{X}_i)^{-1} \boldsymbol{X}_i^T \boldsymbol{Y}_i.$$

(b) Iteratively update $\boldsymbol{A}_i^{(t+1)}$ *until convergence:*

$$\boldsymbol{A}_i^{(t+1)} = \boldsymbol{B}[\boldsymbol{X}_i^T \boldsymbol{Y}_i + \sum_{j \in \mathcal{N}(i)} \boldsymbol{Z}_i^{ij^T} \boldsymbol{Z}_j^{ij} \boldsymbol{A}_j^{(t)}]$$

$$\boldsymbol{B} = [\boldsymbol{X}_i^T \boldsymbol{X}_i + \sum_{j \in \mathcal{N}(i)} \boldsymbol{Z}_i^{ij^T} \boldsymbol{Z}_i^{ij}]^{-1}$$

(c) Set the inverse edge label matrices:

$$A_{i_{inv}} = A_i^{-1}.$$

Theorem 6.2 in (Demmel 1997)[p. 287, chapter 6] states that the Gauss-Seidel method converges if the linear transformation matrix in a linear program is strictly row diagonal dominant (Niethammer, De Pillis, and Varga 1984). In our formulation, diagonal blocks dominate the non-diagonal ones row-wise. Thus, Algorithm 1 should converge to an optimum.

Using Algorithm 1, we learned edge label matrices in AMRs and SDGs independently on corresponding AMRs and SDGs annotations from 2500 bio-sentences (high accuracy auto-parse for SDGs). Convergence was fast for both AMRs and SDGs (log error drops from 10.14 to 10.02 for AMRs, and from 30 to approx. 10 for SDGs).

Next, we flatten an edge label matrix $\boldsymbol{A}_i \in \mathbb{R}^{m \times m}$ to a corresponding edge label vector [5] $\boldsymbol{e}_i \in \mathbb{R}^{m^2 \times 1}$, and then redefine $k_e(x, y)$ in (1) using the sparse RBF kernel.

$$k_e(x, y) = \exp\left((\boldsymbol{e}_x^T \boldsymbol{e}_y - 1)/\beta\right) \left((\boldsymbol{e}_x^T \boldsymbol{e}_y - \alpha)/(1 - \alpha)\right)_+$$

This redefinition enables to define kernel similarity between AMRs and SDGs. One can either use our original formulation where a single AMR/SDG sub-graph is classified using training sub-graphs from both AMRs and SDGs, and then the inference with maximum score-MSI (Bunescu et al. 2006) is chosen. Another option, preferable, is to consider the set $\{G_i^a, G_i^s\}$ as samples of a graph distribution $\mathcal{G}_i$ representing an interaction I_i. Generalizing it further, $\mathcal{G}_i$ has samples set $\{G_{i1}^a, \cdots, G_{ik_i^a}^a, G_{i1}^s, \cdots, G_{ik_i^s}^s\}$, containing k_i^a, k_i^s number of sub-graphs in AMRs and SDGs respectively from multiple sentences in a document, all for classifying I_i. With this graph distribution representation, we can apply our GDK from Section 4 and then infer using a “kernel trick” based classifier. This final formulation gives the best results in our experiments discussed next.

6 Experimental Evaluation

We evaluated the proposed algorithm on two data sets.

PubMed45: This dataset has 400 manual and 3k auto parses of AMRs (and 3.4k auto parses of SDGs)[6]; AMRs auto-parses are from 45 PubMed articles on cancer. From the 3.4k AMRs, we extract 25k subgraphs representing 20k interactions (valid/invalid); same applies to SDGs. This is our primary data for the evaluation.

We found that for both AMR and SGD based methods, a part of the extraction error can be attributed to poor recognition of named entities. To minimize this effect, and to isolate errors that are specific to the extraction methods themselves, we follow the footsteps of the previous studies, and take a filtered subset of the interactions (approx. 10k out of 20k). We refer to this data subset as “PubMed45” and the super set as “PubMed45-ERN” (for entity recognition noise).

AIMed: This is a publicly available dataset[7], which contains about 2000 sentences from 225 abstracts. In contrast to PubMed45, this dataset is very limited as it describes only whether a given pair of proteins interact or not, without specifying the interaction type. Nevertheless, we find it useful to include this dataset in our evaluation since it enables us to compare our results with other reported methods.

Evaluation settings In a typical evaluation scenario, validation is performed by random sub-sampling of labeled interactions (at sentence level) for a test subset, and using the rest as a training set. This sentence-level validation approach is not always appropriate for extracting protein interactions (Tikk et al. 2010), since interactions from a single/multiple sentences in a document can be correlated. Such correlations can lead to information leakage between training and test sets (artificial match, not encountered in real settings). For instance, in (Mooney and Bunescu 2005), the reported F1 score from the random validation in the AIMed data is approx. 0.5. Our algorithm, even using SDGs, gives 0.66 F1 score in those settings. However, the performance drops significantly when an independent test document is processed. Therefore, for a realistic evaluation, we divide data sets at documents level into approx. 10 subsets such that there is minimal match between a subset, chosen as test set, and the rest of sub sets used for training a kernel classifier. In the PubMed45 data sets, the 45 articles are clustered into 11 subsets by clustering PubMed-Ids (training data also includes gold annotations). In AIMed, abstracts are clustered into 10 subsets on abstract-ids. In each of 25 independent test runs (5 for AIMed data) on a single test subset, 80% interactions are randomly sub sampled from the test subset and same percent from the train data.

For the classification, we use the LIBSVM implementation of Kernel Support Vector Machines (Chang and Lin 2011) with the sklearn python wrapper [8]. Specifically, we used settings $\{$ $probability = True$, $C = 1$, $class\_weight = auto\}$. In our data, we have a class “swap” in addition to the two binary classes (“valid”, “invalid”). The “swap” class means that an interaction is invalid as such but swapping of entity roles in the interaction makes it valid. For the analysis purpose however, we focus on F1 scores only for the positive class, i.e. class “valid”.

[5]alternatives for kernel directly on the matrices instead of the flattening can be more accurate, that we plan to explore in the future

[6]not the same 2.5k sentences used in learning edge label vectors

[7]http://corpora.informatik.hu-berlin.de

[8]http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Methods	**PubMed45-ERN**	**PubMed45**	**AIMed**
SDG (SLI)	0.25 ± 0.16 $(0.42, 0.29)$	0.32 ± 0.18 $(0.50, 0.35)$	0.27 ± 0.12 $(0.54, 0.22)$
AMR (SLI)	0.33 ± 0.16 $(0.33, 0.45)$	0.45 ± 0.25 $(0.58, 0.43)$	0.39 ± 0.05 $(0.53, 0.33)$
SDG (MSI)	0.24 ± 0.14 $(0.39, 0.28)$	0.33 ± 0.17 $(0.50, 0.34)$	0.39 ± 0.09 $(0.51, 0.38)$
AMR (MSI)	0.32 ± 0.14 $(0.30, 0.45)$	0.45 ± 0.24 $(0.56, 0.44)$	0.51 ± 0.11 $(0.49, 0.56)$
SDG (GDK)	0.25 ± 0.16 $(0.33, 0.31)$	0.38 ± 0.15 $(0.32, 0.61)$	0.47 ± 0.08 $(0.41, 0.58)$
AMR (GDK)	0.35 ± 0.16 $(0.31, 0.51)$	0.51 ± 0.23 $(0.59, 0.49)$	0.51 ± 0.11 $(0.43, 0.65)$
AMR-SDG (MSI)	0.33 ± 0.18 $(0.29, 0.54)$	0.47 ± 0.24 $(0.50, 0.53)$	$\mathbf{0.55 \pm 0.09}$ $(0.46, 0.73)$
AMR-SDG (GDK)	$\mathbf{0.38 \pm 0.16}$ $(0.33, 0.55)$	$\mathbf{0.57 \pm 0.23}$ $(0.63, 0.54)$	0.52 ± 0.09 $(0.43, 0.67)$
Data Statistics			
Positive ratio	0.07 ± 0.04	0.19 ± 0.14	0.37 ± 0.11
Train-Test Div.	0.014 ± 0.019	0.041 ± 0.069	0.005 ± 0.002

Table 2: F1 score statistics. "SLI" is sentence level inference; "MSI" refers to maximum score inference at document level; "GDK" denotes Graph distribution kernel based inference at document level. Precision, recall statistics are presented as (mean-precision, mean-recall) tuples.

6.1 Evaluation Results

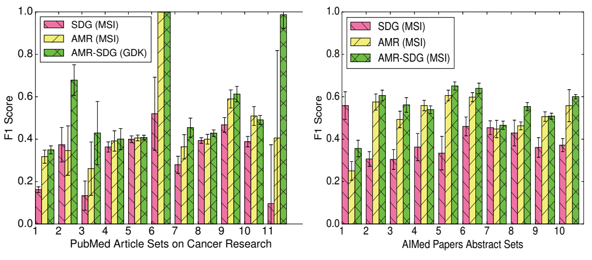
We categorize all methods evaluated below as follows: i) *Sentence Level Inference*-SLI [9]; ii) document level using *Maximum Score Inference*-MSI (Bunescu et al. 2006); and iii) document-level inference on all the subgraphs using our *Graph Distribution Kernel* (GDK). In each of the categories, AMRs, SDGs are used independently, and then jointly. Edge label vectors are used only when AMRs and SDGs are jointly used, referred as "AMR-SDG".

Table 2 shows the F1 score statistics for all the experiments. In addition, the mean of precision and recall values are presented as (precision, recall) tuples in the same table. For most of the following discussion, we focus on F1 scores only to keep the exposition simple.

Before going into detailed discussion of the results, we make the following two observations. First, we can see that, in all methods (including our GDK and baselines), we obtain much better accuracy using AMRs compared to SDGs. This result is remarkable, especially taking into account the fact that the accuracy of semantic parsing is still significantly lower when compared to syntactic parsing. And second, observe that the overall accuracy numbers are considerably lower for the PubMed45-ERN data, compared to the filtered data PubMed45.

Let us focus on document-level extraction using MSI. We do not see much improvement in numbers compared to SLI for our PubMed45 data. On the other hand, even this simple MSI technique works for the restricted extraction settings in the AIMed data. MSI works for AIMed data probably because there are multiple sub-graph evidences with varying interaction types (root node in subgraphs), even in a single sentence, all representing same protein-protein pair interaction. This high number of evidences at document level, should give a boost in performance even using MSI.

[9]Note that even for the sentence level inference, the training/test division is done on document level.



(a) PubMed45 (b) AIMed

Figure 3: Comparison of extraction accuracy (F1 score)

Next, we consider document-level extraction using the proposed GDK method with the MMD metric. Comparing against the baseline SLI, we see a significant improvement for all data sets and in both AMRs and SDGs (although the improvement in PubMed45-ERN is relatively small). The effect of the noise in entity recognition can be a possible reason why GDK does not work so well in this data compared to the other two data sets. Here, we also see that: a) GDK method performs better than the document level baseline MSI; and b) AMRs perform better than SDGs with GDK method also.

Let us now consider the results of extraction using both AMRs and SDGs jointly. Here we evaluate MSI and GDK, both using our edge label vectors. Our primary observation here is that the joint inference using both AMRs and SDGs improves the extraction accuracy across all datasets. Furthremore, in both PubMed45 datasets, the proposed GDK method is a more suitable choice for the joint inference on AMRs and SDGs. As we can see, comparing to GDK for AMRs only, F1 points increment from 0.35 to 0.38 for the PubMed45-ERN data, and from 0.51 to 0.57 for the PubMed45 data. For the AIMed dataset, on the other hand, the best result (F1 score of 0.55) is obtained when one uses the baseline MSI for the joint inference on AMRs and SDGs.

To get more insights, we now consider (mean-precision, mean-recall) tuples shown in the Table 2. The general trend is that the AMRs lead to higher recall compared to the SDGs. In the PubMed45-ERN data set, this increase in the recall is at cost of a drop in the precision values. Since the entity types are noisy in this data set, this drop in the precision numbers is not completely surprising (note that the F1 scores still increase). With the use of the GDK method in the same data set, however, the precision drop (SDGs to AMRs) becomes negligible, while the recall still increases significantly. In the data set PubMed45 (the one without noise in the entity types), both the precision and recall are generally higher for the AMRs compared to the SDGs. Again, there is an exception for the GDK approach, for which the recall decreases slightly. However, the corresponding precision almost doubles.

	MMD	KL-D	CK
SDG	0.25 ± 0.16 $(0.33, 0.31)$	0.21 ± 0.17 $(0.59, 0.21)$	0.26 ± 0.13 $(0.29, 0.38)$
AMR	0.35 ± 0.16 $(0.31, 0.51)$	0.37 ± 0.17 $(0.50, 0.41)$	0.29 ± 0.13 $(0.28, 0.39)$

Table 3: Comparison of F1 scores for different divergence metrics used with GDK. The evaluation is on PubMed45-ERN dataset. "KL-D" and "CK" stand for Kullback-Leibler divergence and Cross Kernels, respectively.

For a more fine-grained comparison between the methods, we plot F1 score for each individual test set in Fig. 3. Here, we compare the baselines, "AMR (MSI)", "SDG (MSI)" against the "AMR-SDG (GDK)" in our data (and, "AMR-SDG (MSI)" for "AIMed"). We see a general trend, across all test subsets, of AMRs being more accurate than SDGs and the joint use of two improving even upon AMRs. Though, there are some exceptions where the difference is marginal between the three. In our cross checking, we find that such exceptions are when there is relatively more information leakage between train-test, i.e. less train-test divergence. We use Maximum Mean Discrepancy-MMD for evaluating this train-test divergence (originally used for defining GDK in Section 4. We find that our GDK technique is more suitable when $MMD > 0.01$ (MMD is normalized metric for a normalized graph kernel).

The results for the GDK method described above are specific to the MMD metric. We also evaluated GDK using two other metrics (KL-D and cross kernels), specifically on "PubMed45-ERN" dataset, as presented in Table 3. Here, as in Table 2, we also present (mean-precision, mean-recall) tuples. We can see that MMD and KL-D metrics, both, perform equally well for AMR whereas MMD does better in case of SDG. CK (cross kernels), which is a relatively naive approach, also performs reasonably well, although for the AMRs it performs worse compared to MMD and KL-D. For the precision and recall numbers in the Table 3, we see similar trends as reported in Table 2. We observe that the recall numbers increase for the AMRs compared to the SDGs (the metric CK is an exception with negligible increase). Also, comparing KL-D against MMD, we see the former favors (significantly) higher precision, albeit at the expense of lower recall values.

7 Related Work

There have been different lines of work for extracting protein extractions. Pattern-matching based systems (either manual or semi-automated) usually yield high precision but low recall (Hunter et al. 2008; Krallinger et al. 2008; Hakenberg et al. 2008; Hahn and Surdeanu 2015). Kernel-based methods based on various convolution kernels have also been developed for the extraction task (Tikk et al. 2010; Airola et al. 2008; Mooney and Bunescu 2005). Some approaches work on string rather than parses (Mooney and Bunescu 2005). The above mentioned works either rely on text or its shallow parses, none using semantic parsing for the extraction task. Also, most works consider only protein-protein interactions while ignoring interaction types. Some recent works used distant supervision to obtain a large data set of protein-protein pairs for their experiments (Mallory et al. 2015).

Document-level extraction has been explored in the past (Skounakis and Craven 2003; Bunescu et al. 2006). These works classify at sentence level and then combine the inferences whereas we propose to infer jointly on all the sentences at document level.

Previously, the idea of linear relational embedding has been explored in (Paccanaro and Hinton 2000), where triples of concepts and relation types between those concepts are (jointly) embedded in some latent space. Neural networks have also been employed for joint embedding (Bordes et al. 2014). Here we advocate for a factored embedding where concepts (node labels) are embedded first using plain text, and then relations (edge labels) are embedded in a linear sub-space.

8 Conclusion

In summary, we have developed and validated a method for extracting biomolecular interactions that, for the first time, uses *deep semantic parses* of biomedical text (AMRs). We have presented a novel algorithm, which relies on *Graph Distribution Kernels* (GDK) for document-level extraction of interactions from a set of AMRs in a document. GDK can operate on both AMR and SDG parses of sentences jointly. The rationale behind this hybrid approach is that while neither parsing is perfect, their combination can yield superior results. Indeed, our experimental results suggest that the proposed approach outperforms the baselines, especially in the practically relevant scenario when there is a noticeable mismatch between the training and test sets.

To facilitate the joint approach, we have proposed a novel edge vector space embedding method to assess similarity between different types of parses. We believe this notion of edge-similarly is quite general and will have applicability for a wider class of problems involving graph kernels. As a future work, we intend to validate this framework on a number of problems such as improving accuracy in AMRs parsing with SDGs.

9 Acknowledgements

This work was sponsored by the DARPA Big Mechanism program (W911NF-14-1-0364). It is our pleasure to acknowledge fruitful discussions with Michael Pust, Kevin Knight, Shuyang Gao, Linhong Zhu, Neal Lawton, Emilio Ferrara, and Greg Ver Steeg. We are also grateful to anonymous reviewers for their valuable feedback.

References

Airola, A.; Pyysalo, S.; Björne, J.; Pahikkala, T.; Ginter, F.; and Salakoski, T. 2008. All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. *BMC Bioinformatics* 9:S2.

Andreas, J.; Vlachos, A.; and Clark, S. 2013. Semantic parsing as machine translation. In *Proc. of ACL*, 47–52.

Banarescu, L.; Bonial, C.; Cai, S.; Georgescu, M.; Griffitt, K.; Hermjakob, U.; Knight, K.; Koehn, P.; Palmer, M.; and Schneider, N. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

Bordes, A.; Glorot, X.; Weston, J.; and Bengio, Y. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning* 94:233–259.

Borgwardt, K. M.; Gretton, A.; Rasch, M. J.; Kriegel, H.-P.; Schölkopf, B.; and Smola, A. J. 2006. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics* 22:e49–e57.

Bunescu, R.; Ge, R.; Kate, R. J.; Marcotte, E. M.; Mooney, R. J.; Ramani, A. K.; and Wong, Y. W. 2005. Comparative experiments on learning information extractors for proteins and their interactions. *Artificial Intelligence in Medicine* 139–155.

Bunescu, R.; Mooney, R.; Ramani, A.; and Marcotte, E. 2006. Integrating co-occurrence statistics with information extraction for robust retrieval of protein interactions from medline. In *Proc. of the Workshop on Linking Natural Language Processing and Biology: Towards Deeper Biological Literature Analysis*.

Chang, C.-C., and Lin, C.-J. 2011. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2:27.

Chen, D., and Manning, C. D. 2014. A fast and accurate dependency parser using neural networks. In *Proc. of EMNLP*, 740–750.

Clark, S. 2014. Vector space models of lexical meaning. *Handbook of Contemporary Semantics, 2nd Ed. Blackwell*.

Culotta, A., and Sorensen, J. 2004. Dependency tree kernels for relation extraction. In *Proc. of ACL*, 423.

Demir, E.; Cary, M. P.; Paley, S.; Fukuda, K.; Lemer, C.; Vastrik, I.; Wu, G.; D'Eustachio, P.; Schaefer, C.; Luciano, J.; et al. 2010. The biopax community standard for pathway data sharing. *Nature Biotechnology* 28:935–942.

Demmel, J. W. 1997. *Applied numerical linear algebra*. Siam.

Flanigan, J.; Thomson, S.; Carbonell, J.; Dyer, C.; and Smith, N. A. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proc. of ACL*.

Gneiting, T. 2002. Compactly supported correlation functions. *Journal of Multivariate Analysis* 83:493–508.

Gretton, A.; Borgwardt, K. M.; Rasch, M. J.; Schölkopf, B.; and Smola, A. 2012. A kernel two-sample test. *JMLR* 13:723–773.

Hahn, M. A. V.-E. G., and Surdeanu, P. T. H. M. 2015. A domain-independent rule-based framework for event extraction. *ACL-IJCNLP 2015* 127.

Hakenberg, J.; Plake, C.; Royer, L.; Strobelt, H.; Leser, U.; and Schroeder, M. 2008. Gene mention normalization and interaction extraction with context models and sentence motifs. *Genome Biol* 9:S14.

Hunter, L.; Lu, Z.; Firby, J.; Baumgartner, W. A.; Johnson, H. L.; Ogren, P. V.; and Cohen, K. B. 2008. Opendmap: an open source, ontology-driven concept analysis engine, with applications to capturing knowledge regarding protein transport, protein interactions and cell-type-specific gene expression. *BMC Bioinformatics* 9:78.

Kim, B., and Pineau, J. 2013. Maximum mean discrepancy imitation learning. In *Proc. of RSS*.

Krallinger, M.; Leitner, F.; Rodriguez-Penagos, C.; and Valencia, A. 2008. Overview of the protein-protein interaction annotation extraction task of biocreative ii. *Genome biology* 9:S4.

Mallory, E. K.; Zhang, C.; Ré, C.; and Altman, R. B. 2015. Large-scale extraction of gene interactions from full-text literature using deepdive. *Bioinformatics* btv476.

McDonald, R.; Pereira, F.; Kulick, S.; Winters, S.; Jin, Y.; and White, P. 2005. Simple algorithms for complex relation extraction with applications to biomedical ie. In *Proc. of ACL*.

Mehdad, Y.; Moschitti, A.; and Zanzotto, F. M. 2010. Syntactic/semantic structures for textual entailment recognition. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 1020–1028.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*.

Mooney, R. J., and Bunescu, R. C. 2005. Subsequence kernels for relation extraction. In *Proc. of NIPS*, 171–178.

Niethammer, W.; De Pillis, J.; and Varga, R. 1984. Convergence of block iterative methods applied to sparse least-squares problems. *Linear Algebra and its Applications* 58:327–341.

Paccanaro, A., and Hinton, G. E. 2000. Learning distributed representations by mapping concepts and relations into a linear space. In *Proc. of ICML*, 711–718.

Pan, S. J.; Kwok, J. T.; and Yang, Q. 2008. Transfer learning via dimensionality reduction. In *Proc. of AAAI*.

Pust, M.; Hermjakob, U.; Knight, K.; Marcu, D.; and May, J. 2015. Using syntax-based machine translation to parse english into abstract meaning representation. In *Proc. of EMNLP*.

Skounakis, M., and Craven, M. 2003. Evidence combination in biomedical natural-language processing. In *Proc. of BIOKDD*.

Srivastava, S.; Hovy, D.; and Hovy, E. H. 2013. A walk-based semantically enriched tree kernel over distributed word representations. In *Proc. of EMNLP*, 1411–1416.

Sutherland, D. J.; Xiong, L.; Póczos, B.; and Schneider, J. 2012. Kernels on sample sets via nonparametric divergence estimates. *arXiv preprint arXiv:1202.0302*.

Tikk, D.; Thomas, P.; Palaga, P.; Hakenberg, J.; and Leser, U. 2010. A comprehensive benchmark of kernel methods to extract protein–protein interactions from literature. *PLoS Comput Biol*.

Wang, Y.; Berant, J.; and Liang, P. 2015. Building a semantic parser overnight. In *Proc. of ACL*.

Wang, Q.; Kulkarni, S. R.; and Verdú, S. 2009. Divergence estimation for multidimensional densities via-nearest-neighbor distances. *IEEE Transactions on Information Theory* 2392–2405.

Zelenko, D.; Aone, C.; and Richardella, A. 2003. Kernel methods for relation extraction. *JMLR* 3:1083–1106.