PIDGIN: Ontology Alignment using Web Text as Interlingua

Derry Wijaya
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA
dwijaya@cs.cmu.edu

Partha Pratim Talukdar
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA
ppt@cs.cmu.edu

Tom Mitchell
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA
tom.mitchell@cs.cmu.edu

ABSTRACT

The problem of aligning ontologies and database schemas across different knowledge bases and databases is fundamental to knowledge management problems, including the problem of integrating the disparate knowledge sources that form the semantic web's Linked Data [5].

We present a novel approach to this ontology alignment problem that employs a very large natural language text corpus as an interlingua to relate different knowledge bases (KBs). The result is a scalable and robust method (PIDGIN[1]) that aligns relations and categories across different KBs by analyzing both (1) shared relation instances across these KBs, and (2) the verb phrases in the text instantiations of these relation instances. Experiments with PIDGIN demonstrate its superior performance when aligning ontologies across large existing KBs including NELL, Yago and Freebase. Furthermore, we show that in addition to aligning ontologies, PIDGIN can automatically learn from text, the verb phrases to identify relations, and can also type the arguments of relations of different KBs.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods; I.2.6 [**Artificial Intelligence**]: Learning; I.2.7 [**Artificial Intelligence**]: Natural Language Processing

General Terms

Algorithms, Experimentation

Keywords

Ontology Alignment, Knowledge Bases, Graph-based Self-Supervised Learning, Label Propagation, Natural Language Processing.

[1]PIDGIN: the source code and relevant datasets will be available at http://rtw.ml.cmu.edu/cikm2013_pidgin/

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.

http://dx.doi.org/10.1145/2505515.2505559.

1. INTRODUCTION

Over the last few years, several large, publicly available Knowledge Bases (KBs) have been constructed, such as DBPedia [3], Freebase [6], NELL [8], and Yago [18]. These KBs consist of both an ontology that defines a set of categories (e.g., *Athlete, Sports*) and relations (e.g., *playerPlaysSport(Athlete, Sport)*), and the data entries which instantiate these categories (e.g., *Tiger Woods* is an *Athlete*) and relations (e.g., *playerPlaysSport(Tiger Woods, Golf)*). The growth of the Semantic Web [4] has contributed significantly to the construction and availability of such KBs. However, these KBs are often independently developed, using different terminologies, coverage, and ontological structure of categories and relations. Therefore, the need for automatic alignment of categories and relations across these and many other heterogeneous KBs is now greater than ever, and this remains one of the core unresolved challenges of the Linked Data movement [5].

Research within the Ontology Matching community has addressed different aspects of this Ontology Alignment problem, with recently proposed PARIS [17] being the current state-of-the-art in this large body of work (see [16] for a recent survey). PARIS is a probabilistic ontology matcher which uses the overlap of instances between two relations (or categories) from a pair of KBs as one of the primary cues to determine whether an equivalence or subsumption relationship exists between those two relations (or categories). PARIS, and the instance overlap principle which it shares with most previous ontology alignment systems, has been found to be very effective in discovering alignments when applied to KBs such as DBPedia [3], Yago [18], and IMDB[2].

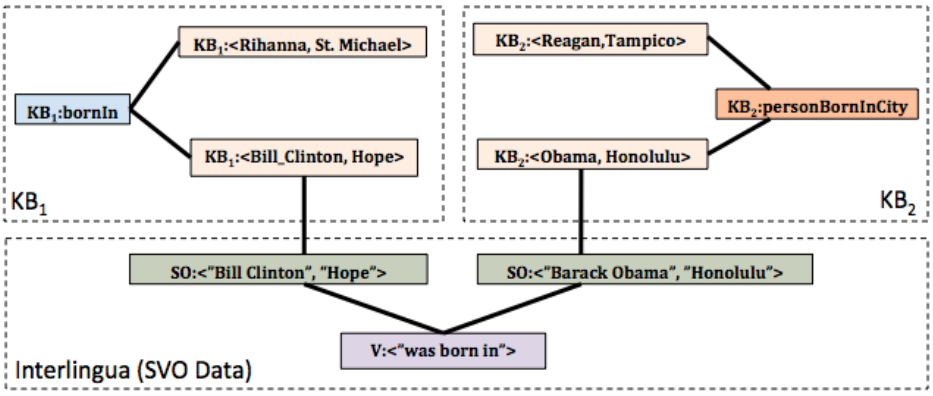
Despite this recent progress, the current state-of-the-art remains insufficient to align ontologies across many practical KB's and databases, especially when they share few or no data entries in common. To overcome this shortcoming, we introduce a new approach that is capable of matching the categories and relations across multiple KB's even in the extreme case where they share no data entries in common. The key idea is to introduce side information in the form of a very large text corpus (in our case, 500 million dependency-parsed web pages). Our approach, called PIDGIN, effectively grounds each KB relation instance (e.g., *playerPlaysSport(Rodriguez, baseball)*) by its mentions in this text, then represents the relation in terms of the verbs that connect its arguments (e.g., the relation *playerPlaysSport(x,y)* might frequently be expressed in text by verbs such as *"x plays y"* or *"x mastered y"*). The distribution of verbs as-

[2]http://www.imdb.com/

Knowledge Base 1 (KB_1):
(Rihanna, bornIn, St. Michael)
(Bill_Clinton, bornIn, Hope)

Knowledge Base 2 (KB_2):
(Reagan, personBornInCity, Tampico)
(Obama, personBornInCity, Honolulu)

Interlingua (Subject-Verb-Object):
(Bill Clinton, was born in, Hope)
(Barack Obama, was born in, Honolulu)
...



(a) (b)

Figure 1: **(a) Inputs to PIDGIN** include KB_1 and KB_2, each consisting of two relation instances (e.g., (*Bill_Clinton, bornIn, Hope*)). Another input is a set of Subject-Verb-Object (SVO) triples (the interlingua) extracted from a natural language text corpus. **(b) Graph constructed from the input** (see Section 4.1). PIDGIN performs inference over this graph to determine that KB_1:bornIn is equivalent to KB_2:personBornInCity (see Section 4.2). Note since there is no overlap between relation instances from these two KBs, algorithms based on instance overlap will be unable to align these two ontologies. PIDGIN overcomes this limitation through use of the SVO-based interlingua and inference over the graph.

sociated with instances of any given relation forms a KB-independent representation of that relation's semantics, which can then be aligned with relations from other KBs. In essence, the verb distributions associated with relations provide an *interlingua* that forms the basis for aligning ontologies across arbitrary KBs, even when their actual data entries fail to overlap. PIDGIN integrates this text information with information about overlapping relation instances across the two KB's using a graph-based self-supervised learning strategy, to determine their final ontology alignment.

In particular, we make the following contributions:

- We present PIDGIN, a novel, scalable, and flexible graph-based ontology aligner. PIDGIN uses natural language text as an interlingua to align the ontologies of different KBs. To the best of our knowledge, this is the first approach to aligning ontologies that makes use of unlabeled web-scale text corpora.
- PIDGIN is self-supervised, and does not require human labeled data. PIDGIN can be easily parallelized and implemented in MapReduce, making it suitable even for aligning ontologies from very large KBs.
- Through extensive experimentation on real-world datasets, we demonstrate effectiveness of PIDGIN, and find that PIDGIN significantly outperforms a state-of-the-art ontology alignment system (PARIS).
- In addition to aligning ontologies, PIDGIN learns verb phrases to identify relations and can type the arguments of relations of different KBs.

2. MOTIVATING EXAMPLE

We first illustrate the core ideas behind PIDGIN through a motivating example. Let us consider the alignment problem involving two KBs as shown in Figure 1. In this case, KB_1 contains the relation (*bornIn*) with two instances, while KB_2 contains the relation *personBornInCity* with two other instances. In this case, we would like to discover the alignment KB_1*:bornIn* $\equiv$ KB_2*:personBornInCity* despite the fact that the relation names are different, and their instances do not overlap. Note that previous ontology alignment algorithms (such as PARIS [18]) will not be able to discover the desired alignment.

Need for Interlingua: In order to overcome such overlap sparsity, PIDGIN analyzes a large natural language text corpus to determine the expression pattern of instances from different relations. For example, in the text corpus, PIDGIN finds that instances of the KB_1*:bornIn* relation is often expressed using the verb phrase *"was born in"*, e.g., *(Bill_Clinton, bornIn, Hope)* is expressed using this verb phrase in the sentence *"Former President Bill Clinton was born in Hope, AK"*. PIDGIN also finds that KB_2 relation instance *(Barack_Obama, personBornInCity, Honolulu)* is expressed using the same verb phrase in sentences such as *"President Barack Obama was born in Honolulu, HI"*. So, even though there is no direct overlap of instances between the relations KB_1*:bornIn* and KB_2*:personBornInCity*, PIDGIN might be able to discover the equivalence KB_1*:bornIn* $\equiv$ KB_2*:personBornInCity* by exploiting overlapping expression patterns (i.e., verbs) of these two relations in natural language text, the **interlingua**.

3. PROBLEM DEFINITION

In this section, we outline the terminology used and present the problem definition. We define a Knowledge Base (KB) K to be a 6-tuple $(C, O_C, I_C, R, O_R, I_R)$, where C is the set of categories (e.g., *athlete*, *sports*), O_C is the category ontology, which specifies the subset/superset hierarchical relations among categories (e.g., that *athlete* is a subset of *person*), I_C is the set of entity-category pairs (e.g., (*Tiger Woods*, *athlete*)) for categories in C, R is the set of relations (e.g., *athletePlaysSport(athlete, sports)*), O_R is the relation ontology, which specifies the hierarchy of relations (e.g., *ceoOf(person,company)* is a special case of the relation *worksFor(person, company)*), and I_R is the set of entity-

relation-entity triples for relations in R (e.g., *(Tiger Woods, athletePlaysSport, Golf)*). We allow O_C and O_R to be empty, i.e., the KB may have a flat category and relation structure. Each instance of a relation $r \in R$ is a 3-tuple $(e_1, r, e_2) \in I_R$, where $(e_1, c_1) \in I_C$ and $(e_2, c_2) \in I_C$ for some $c_1, c_2 \in C$. Note that each entity can be referred to by one or more Noun Phrases (NP). For example, the entity *Tiger Woods*, can be instantiated in text using either the NP *Tiger Woods* or the NP *Eldrick Tont Woods*. Let $N(e)$ be the set of NPs corresponding to entity e.

Let D be the Subject-Verb-Object (SVO) based interlingua consisting of tuples of the form (np_1, v, np_2, w), where np_1 and np_2 are noun phrases (NP) corresponding to subject and object, respectively, v is a verb, and $w \in \mathbb{R}_+$[3] is the normalized count of this tuple in a large text corpus.

Given two sets of relations R_1 and R_2, we define the alignment between them to be the set $A(R_1, R_2) = \{(r_1, a, r_2, w) \mid r_1 \in R_1, a \in \{\equiv, \subseteq\}, r_2 \in R_2, w \in \mathbb{R}\}$, where $\equiv$ signifies relation equivalence, $\subseteq$ less general, and $\mathbb{R}$ is the set of real numbers. In other words, $(r_1, \subseteq, r_2, w)$ signifies that relation $r_1 \in R_1$ is less general than relation $r_2 \in R_2$, with $w \in \mathbb{R}$ representing the confidence in this alignment. Similarly for the equivalence alignment ($\equiv$). We similarly define the category alignments $A(C_1, C_2) = \{(c_1, a, c_2, w) \mid c_1 \in C_1, a \in \{\equiv, \subseteq\}, c_2 \in C_2, w \in \mathbb{R}\}$.

Problem Definition: Given two knowledge bases (KBs) $K_1(C_1, O_{C_1}, I_{C_1}, R_1, O_{R_1}, I_{R_1})$ and $K_2(C_2, O_{C_2}, I_{C_2}, R_2, O_{R_2}, I_{R_2})$, and a syntactically-parsed text corpus D, we would like to discover the category and relation alignments, $A(C_1, C_2)$ and $A(R_1, R_2)$ respectively.

4. PIDGIN

PIDGIN is able to exploit large web text as interlingua by posing the ontology alignment problem as a classification problem over an appropriately constructed graph. The system consists of two stages:

1. **Graph Construction**: Given two KBs and a Subject-Verb-Object (SVO) based interlingua, PIDGIN first represents this data as a graph. An example graph constructed from the the input in Figure 1(a) is shown in Figure 1(b). See Section 4.1 for details.

2. **Alignment as Classification over Graph**: Once the graph is constructed in Stage 1, PIDGIN poses ontology alignment as node classification over this graph. For the graph in Figure 1(b), PIDGIN first associates two labels with the node *KB_1:bornIn*, one for equivalence and the other for subsumption, both specific to this node. Starting with this initial seed information and graph structure, PIDGIN will use the graph-based semi-supervised learning (SSL) described in [19] to classify the rest of the nodes in the graph, including the node corresponding to the relation in KB_2. Based on the assignment of scores of these labels on the KB_2 relation node, PIDGIN will determine the alignments between ontologies from these two KBs. PIDGIN starts out by attempting to align relations from the two KBs, and produces category alignment as an important by product. Please see Section 4.2 for details.

[3] $\mathbb{R}_+$ is the set of positive reals.

For ease of explanation and readability, we present all examples and descriptions involving two ontologies. However, please note that PIDGIN is capable of handling multiple ontologies simultaneously.

We now turn to describing PIDGIN's two stages in detail.

4.1 Stage 1: Graph Construction

Given $\text{KB}_1(C_1, O_{C_1}, I_{C_1}, R_1, O_{R_1}, I_{R_1})$, $\text{KB}_2(C_2, O_{C_2}, I_{C_2}, R_2, O_{R_2}, I_{R_2})$, and a SVO-based interlingua D[4], PIDGIN first constructs a graph $G = (V, E, W)$, where V is the set of vertices, E is the set of edges, and $W_{i,j}$ representing the weight of the edge $(i, j) \in E$. We describe the construction of this graph below.

We first initialize $V = \phi$, $E = \phi$, and W is an all zero matrix, with edge weight 0 indicating absence of the edge. All edges in G are undirected and untyped.

- **Relation-Entity Pair edge**: For each $(e_1, r, e_2) \in I_{R_k}$ $\forall k \in \{1, 2\}$, add vertices a = KB_k:r and b = KB_k:<e1,e2> to V, add the edge (a, b) to E, and set $W_{a,b} = W_{b,a} = 1.0$. In Figure 1(b), *(KB_1:bornIn, KB_1:<Bill_Clinton, Hope>)* is an example of such an edge.

- **Entity Pair-NP Pair edge**: For each $k \in \{1, 2\}$, we define,

$$Q^k = \{\text{SO}:<np_1, np_2> \mid np_1 \in N(e_1), \\ np_2 \in N(e_2), \text{KB}_k:<e_1, e_2> \in V\}$$

  where $N(e)$ returns the set of NPs corresponding to entity e (see Section 3). Now, for each b = KB_k:<e_1, e_2> $\in V$ $\forall k \in \{1, 2\}$ we define,

$$Q_b = \{\text{SO}:<np_1, np_2> \mid np_1 \in N(e_1), np_2 \in N(e_2) \wedge \\ (\text{SO}:<np_1, np_2> \in Q^1 \cap Q^2 \vee \exists\ v \text{ s.t. } (np_1, v, np_2) \in D)\}$$

  In other words, Q_b is the cross product of NPs used to express the two entities in b, with the requirement that the NP pair is either present in D, or they also correspond to some entity pair from the other KB. We set $V = V \cup Q_b$, and add the edges $\{(b, q) \mid q \in Q_b\}$ to E, with edge weight set to 1.0. In Figure 1(b), (KB_1:<Bill_Clinton, Hope>, SO:<"Bill Clinton", "Hope">) is an example of such an edge.

- **NP Pair-Verb edge**: Let Q be the union of all Q_b sets defined above. In other words, Q is the set of all NP pair nodes in graph G, with each node named SO:<np_1, np_2> for some NPs np_1 and np_2. We define $T = \{v \mid \text{SO}:<np_1, np_2> \in Q, (np_1, v, np_2, w) \in D\}$. We now set $V = V \cup T$, and $E = E \cup \{(q, v) \mid q \in Q, v \in T\}$ with the edge weight set to w. In Figure 1(b), (SO:<"Bill Clinton", "Hope">, "was born in") is an example of such an edge.

At the end of this stage, we end up with a graph $G = (V, E, W)$ as shown in Figure 1(b) when given Figure 1(a) as input to PIDGIN.

[4] For the experiments in this paper, we collected about 600 million SVO triples from the entire ClueWeb [7] corpus of about 230 billion tokens. To the best of our knowledge, this is the largest such resource used for this line of study.

4.2 Stage 2: Alignment as Classification over Graph

At this point, we have a graph $G = (V, E, W)$ with $n = |V|$ and $m = |E|$. PIDGIN poses the ontology alignment problem as one of classification of nodes in G. It starts out by attempting to align relations from the two KBs, and produces category alignment as a by-product. So, we shall first look at how PIDGIN solves the relation alignment problem.

For each relation $r_1 \in R_1$, PIDGIN generates two labels and injects them as seed labels into nodes of the graph G as follows:

- l_{r_1}: This label is node-specific, and is injected *only* on the node named $\text{KB}_1{:}r_1$ which corresponds to relation r_1 in V, i.e., this is self injection. This label will be used to establish equivalence with other relations in KB_2.

- $l_{\vec{r}_1}$: This label is injected as seed to the set of nodes $\{\text{KB}_1{:}s \in V \mid s \in \text{childrenOf}(O_1, r_1)\}$. In other words, $l_{\vec{r}_1}$ is injected into nodes corresponding to children of relation r_1 as determined by ontology O_1. However, if no such child exists, then this label is effectively discarded. This label will be used to identify subsumption relations in KB_2 which are subsumed by, i.e., less general than, r_1.

Let L_1 be the union of these labels, with $|L_1| \leq 2 \times |R_1|$. Starting with this seed information, PIDGIN now applies Modified Adsorption (MAD) [19], a graph-based self-supervised learning (SSL) algorithm, to classify the rest of the nodes in the graph taking transitivity in account. Since nodes corresponding to R_1 were injected, let $\hat{Y}_1 \in \mathbb{R}_{n \times |L_1|}$ be the estimated label score matrix generated by MAD, where $\hat{Y}_1(r, l)$ is the score of label $l \in L_1$ on node $r \in V$. In the current setting, MAD will solve the following optimization problem to estimate $\hat{Y}_1$:

$$\arg\min_{\hat{Y}_1} \sum_{l \in L_1'} \Bigg[\mu_1 \sum_{v \in V} S_v \ (Y_1(v,l) - \hat{Y}_1(v,l))^2 + \mu_2 \sum_{u,v \in V} M_{u,v} (\hat{Y}_1(u,l) - \hat{Y}_1(v,l))^2 + \mu_3 \sum_{v \in V} (\hat{Y}_1(v,l) - F(v,l))^2 \Bigg]$$

where $\mu_1, \mu_2,$ and μ_3 are hyperparameters; $L_1' = L_1 \cup \{\perp\}$ with $\perp$ as the none-of-the-above label; S is a seed node selection vector with $S_v = 1$ if $v \in R_1$ and 0 otherwise; $Y_1(v, l)$ is the score of seed label l on node v (if any); M is a modified version of edge weight matrix W; and F is a regularization matrix. In the MAD objective above, the first and third terms encourage the algorithm to match seed scores (if any) and regularization targets in a soft way, respectively, while the second term encourage smooth label score variation over graph. This is essentially soft enforcement of transitivity, making MAD a suitable inference scheme for PIDGIN. MAD's objective is convex which it solves exactly by iteratively updating scores of labels on the nodes. This takes the form of propagating labels over the graph. These updates can be easily implemented in MapReduce, thereby making MAD, and hence PIDGIN, suitable for large ontology alignment problems[5]. We refer the reader to [19] for further details on MAD.

After estimating $\hat{Y}_1$ as described above, PIDGIN repeats the same process, but in the reverse direction, i.e., it now injects the nodes corresponding to $r \in R_2$ with label set L_2 and propagates those labels to rest of the nodes using MAD. As before, we end up with an estimated label score matrix $\hat{Y}_2 \in \mathbb{R}_{n \times |L_2|}$.

The final set of relation alignments, $A(R_1, R_2)$, discovered by PIDGIN can be divided into the following two subsets:

$$A(R_1, R_2) = A_{\equiv}(R_1, R_2, \hat{Y}_1, \hat{Y}_2) \cup A_{\subseteq}(R_1, R_2, \hat{Y}_1, \hat{Y}_2)$$

4.2.1 Equivalence Alignment

The equivalence alignments between relation sets R_1 and R_2 are estimated as follows:

$$A_{\equiv}(R_1, R_2, \hat{Y}_1, \hat{Y}_2) = \{(r_1, \equiv, r_2, \hat{Y}_1(r_2, l_{r_1}) \times \hat{Y}_2(r_1, l_{r_2})) \mid r_1 \in R_1, r_2 \in R_2, l_{r_1} \in L_1, l_{r_2} \in L_2\}$$

where $\hat{Y}_1$ and $\hat{Y}_2$ are the label score matrices estimated by MAD as described in previous section. In other words, to establish the equivalence $r_1 \equiv r_2$, we want to make sure that relation node r_2 is assigned the label l_{r_1} specific to relation r_1 with a high score by MAD, and vice versa. We define final equivalence alignment score as $\hat{Y}_{\equiv}(r_1, r_2) = \hat{Y}_1(r_2, l_{r_1}) \times \hat{Y}_2(r_1, l_{r_2})$.

4.2.2 Subsumption Alignment

Subsumption alignments between relation sets R_1 and R_2 are estimated as follows:

$$A_{\subseteq}(R_1, R_2, \hat{Y}_1, \hat{Y}_2) = \{(r_2, \subseteq, r_1, \hat{Y}_1(r_2, l_{\vec{r}_1}) \mid r_1 \in R_1, r_2 \in R_2, l_{\vec{r}_1} \in L_1\}$$

In other words, PIDGIN infers alignment $r_2, \subseteq, r_1$ if the subsumption label $l_{\vec{r}_1}$ specific to node $r_1 \in R_1$ is assigned by MAD with higher score than the label l_{r_1} to node corresponding to relation $r_2 \in R_2$. We shall call this score for $r_2, \subseteq, r_1$ by $\hat{Y}_{\subseteq}(r_1, r_2) = \hat{Y}_1(r_2, l_{\vec{r}_1})$.

4.2.3 Concept Alignment

From Section 4.2.1, we have $\hat{Y}_{\equiv}(r_1, r_2)$ as the relation equivalence score estimated by PIDGIN for relations $r_1 \in R_1$ and $r_2 \in R_2$. PIDGIN uses this estimate to establish category equivalence alignments as follows. Given a relation r, let $\text{Dom}(r)$ and $\text{Ran}(r)$ be its domain and range categories, i.e., categories of the two entities connected by this relation. We define,

$$H_{\equiv}^{\text{Dom}}(c_1, c_2) = \sum_{\substack{r_1 \in R_1, c_1 = \text{Dom}(r_1), \\ r_2 \in R_2, c_2 = \text{Dom}(r_2)}} \hat{Y}_{\equiv}(r_1, r_2)$$

$$H_{\equiv}^{\text{Ran}}(c_1, c_2) = \sum_{\substack{r_1 \in R_1, c_1 = \text{Ran}(r_1), \\ r_2 \in R_2, c_2 = \text{Ran}(r_2)}} \hat{Y}_{\equiv}(r_1, r_2)$$

We define the final category equivalence alignments as,

$$A(C_1, C_2) = \{(c_1, \equiv, c_2, H_{\equiv}^{\text{Dom}}(c_1, c_2) + H_{\equiv}^{\text{Ran}}(c_1, c_2)) \mid c_1 \in C_1, \ c_2 \in C_2\}$$

[5]For the experiments in this paper, we use the MAD implementation provided by the Junto toolkit (https://code.google.com/p/junto/), which also includes Hadoop-based implementations of MAD.

KB	Relations	Relation Instances
Freebase	79	7,450,452
NELL	499	3,235,218
Yago2	23	1,770,163
KBP	17	1,727

Table 1: Statistics of KBs used in experiments. We use NELL as a common target to align other KBs to, and consider only those relations in other KBs that have alignments to NELL relations (as decided by human annotators).

5. EXPERIMENTS

5.1 Experimental Setup

We conduct our experiments on several large scale open-domain publicly available real-world KBs, namely NELL [8] (a large scale KB extracted automatically from web text), Yago2 [11] (a large scale KB extracted automatically from semi-structured text of Wikipedia infoboxes), Freebase [6] (a large scale KB created collaboratively and manually by humans), and KB Population (KBP) dataset (a smaller scale, manually constructed dataset used in the 2012 Text Analysis Conference for entity-linking, slot-filling and KB population tasks[6]). Table 1 shows the statistics of the KBs used in our experiments.

In each experiment, we are given two KBs to align: KB_1 and KB_2 with sets of relations R_1 and R_2. For equivalence alignments, **PIDGIN** returns for each relation $r_1 \in R_1$, a list of $r_2 \in R_2$ ranked by $\hat{Y}_{\equiv}(r_1, r_2)$ (see Section 4.2.1). For each r_1, we compare our ranked list of r_2 with that of **PARIS**. We infer PARIS equivalence alignments from its output subsumption alignments P_{12} and P_{21}, i.e., $P_{12} = \{(r_1, \subseteq, r_2, score_{p_{12}})\}$ and $P_{21} = \{(r_2, \subseteq, r_1, score_{p_{21}})\}$ where $score_{p_{12}}$ is PARIS confidence measure that $r_1 \subseteq r_2$ and $score_{p_{21}}$ is PARIS confidence measure that $r_2 \subseteq r_1$. We compute PARIS equivalence alignments $P = \{(r_1, \equiv, r_2, score_{p_{12}} * score_{p_{21}})\}$, i.e., we define *equivalence* ($\equiv$) relation between r_1 and r_2 if both $r_1 \subseteq r_2$ and $r_2 \subseteq r_1$. We also compare against a baseline that computes the equivalence of r_1 and r_2 using *Jaccard* similarity measures based on the number of overlap instances that r_1 and r_2 have, i.e., $Jaccard(r_1, r_2) = \frac{|I_{r_1} \cap I_{r_2}|}{|I_{r_1} \cup I_{r_2}|}$, where I_r is the set of instances of relation r. We call this overlap-based alignment **JACCARD (inst)**.

For subsumption alignments, **PIDGIN** returns for each relation $r_1 \in R_1$, a list of $r_2 \in R_2$ ranked by $\hat{Y}_{\subseteq}(r_1, r_2)$ (see Section 4.2.1).

The list of the systems evaluated include:

- **JACCARD (inst)**: baseline alignment that uses Jaccard Similarity measures based on instance overlap
- **JACCARD (inst + NPs + verb)**: baseline alignment that uses Jaccard Similarity measures based on instance, NP pair and verb overlap
- **PARIS**: a recently proposed state-of-the-art ontology alignment system [17]
- **PIDGIN**: our approach run on a graph that includes relation, instance, NP pair and verb nodes. Also called **PIDGIN (inst + NPs + verb)** or **PIDGIN (binary)**
- **PIDGIN (inst)**: our approach run on a graph that only includes relation and instance nodes
- **PIDGIN (inst +NPs)**: our approach run on a graph that only includes relation, instance, and NP pair nodes
- **PIDGIN (binary)**: our approach run on a graph constructed from SVO dataset and two KBs
- **PIDGIN (multiple)**: our approach run on a graph constructed from SVO dataset and more than two KBs

In the experiments, we want to answer the following:

- Whether PIDGIN improves precision, recall and F1-score of relation and category alignments. We evaluate **PIDGIN**, **PARIS**, **JACCARD (inst)** for relation alignments and **PIDGIN** and **PARIS**, for category alignments (Section 5.3)
- Whether adding more resources from text and more KBs as background knowledge in the graph improves alignment accuracy. We evaluate **PIDGIN (inst)**, **PIDGIN (inst +NPs)**, **PIDGIN (inst + NPs + verb)** that have different sets of resources included in the graph. We also evaluate **PIDGIN (binary)** against **PIDGIN (multiple)** that have additional KB(s) added to its graph (Section 5.4)
- Whether using Label Propagation for alignment is useful. We evaluate the overlap-based approach **JACCARD (inst + NPs + verb)** against our label propagation approach **PIDGIN (inst + NPs + verb)** where both systems use the same sets of resources (Section 5.5)
- Whether PIDGIN is tolerant to noise. We evaluate **PIDGIN** and **PARIS** for tolerance to different fractions of noisy facts in the KB (Section 5.6)
- What are the useful by-products of PIDGIN? We evaluate and analyze various by-products of PIDGIN (Section 5.7)

5.2 Performance Measures

When KB_1 and KB_2 are aligned, for each relation $r_1 \in R_1$, a list of relations $r_2 \in R_2$ that align to r_1 is returned, ranked by some scores. We treat this ranked list as a returned "document" for r_1. The document is considered *relevant* at top-k, if any of the relations r_2 in its top-k matches the gold standard r_2^{gold} for r_1. The gold standard r_2^{gold} for r_1, $r_2^{gold} \subseteq R_2$, is a set of relations that are deemed equivalent to (or subsume) r_1 by human annotators for the task of equivalence (or subsumption) alignment between R_1 and R_2. In the case that $|r_2^{gold}| > 1$, we consider a document relevant if it returns at least one of the relations in this set. We measure *precision* of a system as the number of *relevant* documents returned by the system, divided by the total number of documents returned by the system. We measure *recall* of a system as the number of relevant documents returned by the system over the number of relations r_1 for which there is a gold standard alignment i.e., $|r_2^{gold}| > 0$. The *F1*-score measures the harmonic mean of the precision and recall. We report these precision, recall, and F1-scores at various values of k. Precision of a 100% at $k = 1$ means that for every relation $r_1 \in R_1$, its gold standard mapping can be found at the top

[6] http://www.nist.gov/tac/2012/KBP/index.html

KB Pair	System	Prec	Recall	F1
Freebase & NELL	JACCARD (inst)	0.61	0.51	0.56
	PARIS	0.47	0.09	0.15
	PIDGIN	**0.65**	**0.61**	**0.63**
Yago2 & NELL	JACCARD (inst)	0.56	0.43	0.49
	PARIS	**0.67**	0.09	0.15
	PIDGIN	0.52	**0.52**	**0.52**
KBP & NELL	JACCARD (inst)	0.0	0.0	0.0
	PARIS	0.0	0.0	0.0
	PIDGIN	**0.07**	**0.06**	**0.06**

Table 2: Precision, Recall and F1 scores @k=1 of Relation equivalence alignments comparing overlap based approach such as JACCARD and PARIS with PIDGIN. For each KB pair, best performance is marked in bold (See Section 5.3.1 for details)

KB Pair	System	Prec	Recall	F1
Freebase & NELL	PARIS	0.36	0.08	0.13
	PIDGIN	**0.8**	**0.77**	**0.79**
Yago2 & NELL	PARIS	0.33	0.06	0.09
	PIDGIN	**0.65**	**0.61**	**0.63**
KBP & NELL	PARIS	1.0	0.13	0.24
	PIDGIN	**0.8**	**0.8**	**0.8**

Table 3: Precision, Recall and F1 scores @k=1 of relation subsumption alignments comparing PARIS with PIDGIN. For each KB pair, best performance is marked in bold. (See Section 5.3.2 for details)

of the list of alignments for r_1. Precision of 100% at $k = 5$ means that for every relation r_1, its gold standard mapping can be found somewhere within the top 5 relations in the output list of alignments for r_1.

5.3 Relation and Category Alignment

We find alignments between Freebase and NELL, Yago2 and NELL, and KBP and NELL. We evaluate precision, recall and F1 scores of resulting alignments against the gold standard alignments produced by human annotators.

5.3.1 Relation Equivalence Alignment

In this set of experiments, we compare the performance of PIDGIN, PARIS, and JACCARD (inst) for relation equivalence alignment. We report precision, recall and F1-scores at $k = 1$ of the equivalence alignments returned. We observe in Table 2 that PIDGIN always has much higher recall (without sacrificing precision) than PARIS or JACCARD (inst). In two of the three experiments, precision of PIDGIN is also highest. We conjecture that recall improves due to the improved coverage of the alignments by the use of interlingua and transitivity of inference in PIDGIN. For example, the relation */medicine/medical_treatment/side_effects* in Freebase has no instance overlap with any relation in NELL; thus PARIS and JACCARD (inst) are not able to find alignments for this relation. However, its instances co-occur with some similar verbs in the SVO such as "*may promote*", "*can cause*", "*exacerbate*" as instances of the relation *drughassideeffect* in NELL. Thus PIDGIN is able to map this Freebase relation to the appropriate NELL relation. Another example is the reified relation */sports/league/arena_stadium* in Freebase that has no instance overlap with relations in NELL. However, its instances are represented by some similar NP pairs as instances of the relation *leaguestadiums* in NELL. Thus PIDGIN is able to find a mapping of this relation.

System	F1 @k=1	F1 @k=3	F1 @k=5
PARIS	0.0	0.47	0.68
PIDGIN	**0.53**	**0.68**	**0.79**

Table 4: F1 scores @k = 1, 3 and 5 of Category Equivalence Alignments between Yago2 and NELL, comparing PARIS and PIDGIN. For each k, best performance is marked in bold. (See Section 5.3.3 for details)

The effect of over-reliance on instance overlap to predict alignments is worse when the KB is small. In the KBP dataset, PARIS only returns equivalence alignments for 2 out of the 17 relations in KBP. No correct alignments are returned by either PARIS or JACCARD at $k = 1$. We also observe that precision, recall and F1-scores are highest when aligning Freebase and NELL. This maybe because Freebase has the largest number of facts and is probably the cleanest KB (since it is created manually). The scores are the lowest when aligning KBP and NELL, probably because the KBP is very small and have sparse edges to both NELL and the SVO. In future, for such small and domain-specific KB, we can expand our interlingua to include more targeted natural language text, for example by using web search to find more documents that mention entities in the KB.

5.3.2 Relation Subsumption Alignment

In these experiments, we compare the performance of PIDGIN and PARIS for relation subsumption alignment. We report precision, recall and F1-scores at $k = 1$ of the subsumption alignments returned. [7] We observe in Table 3 that, similar to the equivalence alignment results, PIDGIN always has much higher recall (without sacrificing precision) than PARIS. In two of the three experiments, precision of PIDGIN is highest. We conjecture that the improved recall (and subsequently F1) is due to the use of interlingua and transitivity of inference in PIDGIN.

5.3.3 Category Equivalence Alignment

We compare the performance of PIDGIN and PARIS for inferring category equivalence alignment between Yago2 and NELL. PARIS infers category alignment from its instance alignment [17]. We report the F1-scores @k=1, 3, and 5 of the returned alignments against the gold standard alignments. We observe that PIDGIN category alignments have highest F1-scores at different values of k (Table 4). Some examples of PIDGIN alignments include Yago2 category *wordnet_actor_109765278* aligned to NELL category *actor*, Yago2 category *yagoURL* aligned to NELL category *website*, and Yago2 category *yagoLegalActor* aligned to NELL category *agent*.

5.4 Effect of Interlingua Size

In these experiments, we evaluate whether adding more resources extracted from text to the graph improves alignment performance. We construct three graphs: the first contains only relation and instance nodes (PIDGIN (inst)). The second has added NP pair nodes representing instances (PIDGIN (inst + NPs)), while the third contains relation, instance, NP pair, and verb nodes (PIDGIN (inst + NPs + verbs)). We report F1-scores obtained by PIDGIN on these

[7] In this evaluation, we do not count alignment to NELL's generic relation *relatedto* as correct, since it trivially subsumes all others.

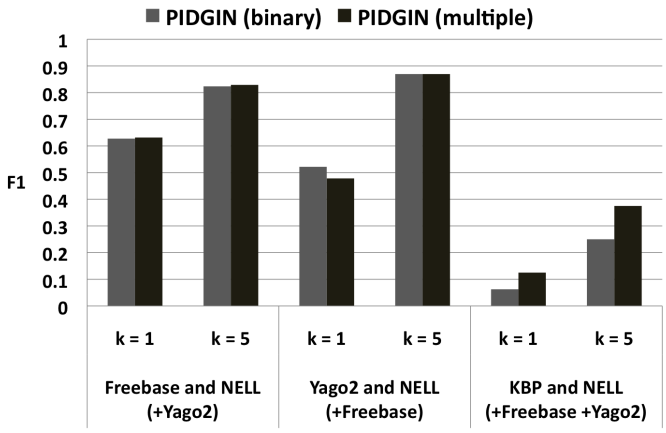


Figure 2: F1 scores @$k = 1$ and 5 of relation equivalence alignments comparing the performance of PIDGIN when only two KBs are used in the graph (binary, grey) compared to the setting when multiple KBs are used in the graph (multiple, black). (See Section 5.4 for details)

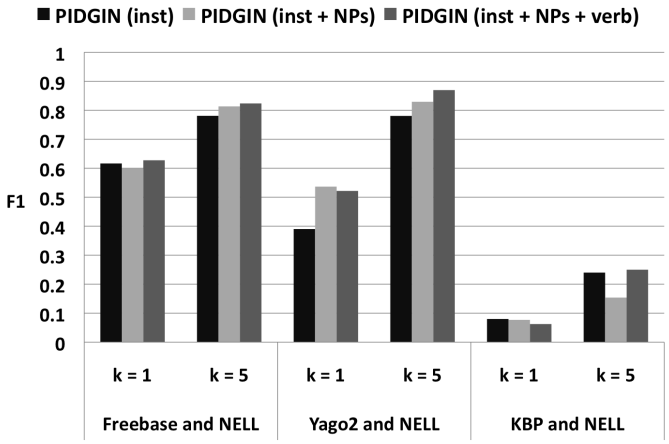


Figure 3: F1 scores @$k = 1$ and 5 of relation equivalence alignments when varying amount of interlingua text is used: PIDGIN using just relation instances (black), PIDGIN using both instances and NPs (light grey) and PIDGIN using instances, NPs and verbs (grey). (See Section 5.4 for details)

three graphs. We observe in Figure 3 that adding more resources to the graph improves performance. Using all the resources available seems to result in the best performance than using relations and instances alone.

Previous works have suggested that adding more ontologies as background knowledge improves the resulting alignments [1], the results of our experiments here seem to confirm this. We construct a graph containing the SVO data and the relations and instances from more than two KBs. We propagate relation labels as before to align Freebase and NELL (with added Yago2 in the graph), to align Yago2 and NELL (with added Freebase in the graph), and to align KBP and NELL (with added Freebase and Yago2 in the graph). The transitivity of inference in our approach allows two relations with no instance overlap to be aligned through an *interlingua*, which can take the form free text or structured ones. In this case, the additional KBs are another *interlingua*. We observe in Figure 2 that adding data from more KBs (PIDGIN (multiple)) results in a comparable or improved performance of alignment compared to using data from only two KBs (PIDGIN (binary)). When the KB is small (e.g., KBP), adding more data to the graph approximately doubles the F1-scores. We believe this is due to the increased connectivity of the graph (more paths for propagating NELL labels to corresponding KBP labels). This increased connectivity combined with transitivity of inference improves performance. Adding more KBs to the already large graph (Freebase and NELL or Yago2 and NELL) does not seem to improve performance. This begs the question that we can explore in future of how much background knowledge is necessary to improve alignment performance.

5.5 Effect of Transitivity of Inference

In these experiments, we evaluate whether the transitivity of inference and joint inference in label propagation (PIDGIN) improve alignment performance when compared to the overlap-based approach (JACCARD). We observe in Figure 4 that using PIDGIN improves performance (F1-scores) at various values of k compared to the overlap-based approach even when the same set of resources are being used in both. We also experiment with using cosine based similarity on the same set of resources to compute alignment. We observe but do not report here that cosine-based approach gives comparable or worse performance than using Jaccard similarity.

We also notice in separate experiments that in some cases, adding NP pairs and verbs to the overlap-based approach decreases performance when compared to using instance overlap alone. This maybe due to the amount of noise that is inherent in the natural language corpus from which the SVO dataset is obtained. However, we have observed previously in Section 5.4 that adding these resources does not hurt PIDGIN performance. The joint inference framework of PIDGIN that jointly optimizes similarities coming from instances, NP pairs and verbs, may be what makes it more tolerant to noise in the interlingua than a simple overlap-based similarity.

5.6 Effect of Noise

In these experiments, we compare the performance of PIDGIN and PARIS when different amounts of noise are added to the KB. We randomly pick some instances in a KB and randomly switch them to other relations in the same KB - adding *noise* to the KB. Note that the original KB may already contain noise due to noisy extractions or wrongly added facts, for example. We observe in Figure 5 that the performance (F1-scores at $k = 5$) of PIDGIN decreases only gradually with the amount of noise added, while the performance of PARIS decreases rapidly and in some cases drops to zero, which means no correct alignments are returned by PARIS. Surprisingly, adding 20% of noise to KBP improves performance. Due to the randomness with which we introduce *noise*, some relations in KBP that are noisy actually become cleaner when the random process coincidentally removes these instances and assigns them to other relations.

5.7 Practical By-Products by PIDGIN

In this section we consider capabilities of PIDGIN to perform tasks beyond aligning relations across ontologies.

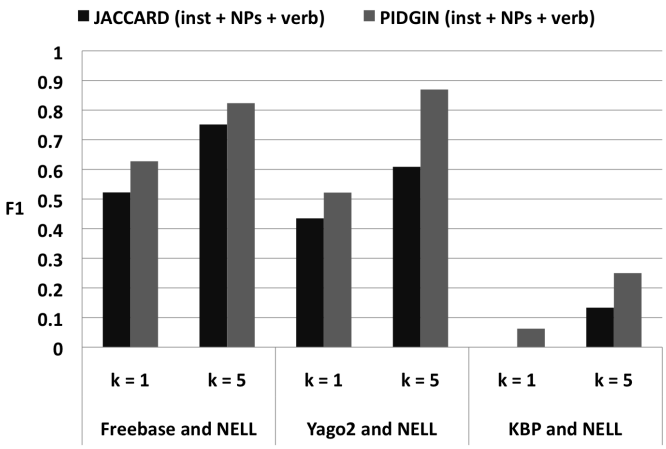


Figure 4: F1 scores @$k = 1$ and 5 of relation equivalence alignments comparing overlap-based approach (black) and PIDGIN (grey) both using the same set of resources. This demonstrates the benefit of transitivity of inference which is exploited by PIDGIN. (See Section 5.5 for details)

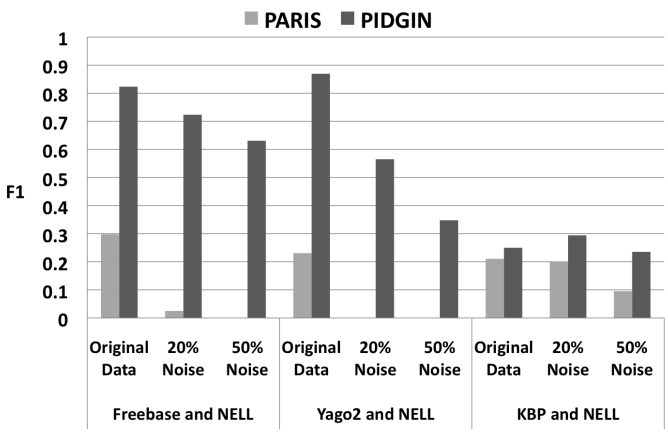


Figure 5: F1 scores @$k = 5$ of relation equivalence alignments when varying amount of noise is introduced into the KB, comparing performance of PARIS (light grey) and PIDGIN (grey). PARIS performance drops drastically to zero as more noise is added, while PIDGIN is more robust to noise. (See Section 5.6 for details)

5.7.1 Argument Typing of Relations with NELL Categories

For each Freebase, Yago2 and KBP relation, we convert the ranked list of NELL relations aligned to it into a ranked list of the corresponding NELL ⟨domain,range⟩ pairs, summing up the scores of similar ⟨domain,range⟩ pairs in the list. For each Freebase, Yago2 and KBP relation, we then have a ranked list of NELL ⟨domain,range⟩ as candidate argument types for the relation. For example, KBP relation *per:city_of_birth* has NELL ⟨*person*,*city*⟩ type. We evaluate F1-scores of these ranked candidate types against the gold standard argument types. We observe in Figure 6 that PIDGIN types have higher F1-scores than PARIS for typing relations in different KBs. Some examples of typing produced by PIDGIN include Yago2 *isPoliticianOf* assigned NELL ⟨*person*, *geoPoliticalLocation*⟩ type, and Freebase */business/industry/name* (i.e., the type of industry a company operates in) being assigned NELL category pair ⟨ *company*, *economicsector*⟩.

5.7.2 Learning Verbs for Relations

As a by product of label propagation on the graph, each verb and NP-pair node in the graph will be assigned scores for each relation label. Exploiting these scores, we can estimate the probability that a verb v represents a relation r as $P(v|r) \approx \frac{\hat{Y}(v,r)}{\sum_{v'} \hat{Y}(v',r)}$, where $\hat{Y}(v,r)$ (from Section 4.2) is the score of label r assigned to verb node v. Since a verb may represent different relations depending on the NP-pair with which it co-occurs e.g., the verb *enter* has different meaning when it appears with an NP-pair ⟨*Paul*, *room*⟩ from when it appears with an NP-pair ⟨*John*, *American Idol*⟩; when estimating $P(v|r)$ we also take into account the scores of r on the NP-pair nodes $\langle NP_1, NP_2\rangle$ with which verb v co-occurs. Similar as before, $P(v|r) \approx \frac{\hat{Y}(v,r)}{\sum_{v'} \hat{Y}(v',r)}$. But now we measure $\hat{Y}(v,r) = \sum_{T_v} \hat{Y}(T_v, r)$, where T_v is a SVO triple $\langle np_1, v, np_2\rangle$, and where $\hat{Y}(T_v, r) = \hat{Y}(\langle np_1, np_2\rangle, r) * \hat{Y}(v, r)$. We multiply this estimate with the tf-idf score of the verb, which is proportional to the number of times a verb appears for a relation, but is offset by the total number of times the verb appears with all the relations. This helps to reduce the effect of common verbs such as *is*, *become* that represent many relations.

Using this scoring, for each relation we can return a ranked list of verbs that represent the relation. Some of the verbs returned are shown in Table 5. As we can see in Table 5, the system is able to distinguish verbs representing the relation */medicine/medical_treatment/side_effects*: "*exacerbate*", "*can cause*" from the verbs representing the antonym relation *drugPossiblyTreatsPhysiologicalCondition*: "*relieve*", "*can help alleviate*" even when the two relations have the same domain (*drug*) and range (*physiological condition*). The system is also able to recognize the directionality of the relation. For example, for the relation *_acquired*, which represents the inverse of the relation *acquired* (as in company X acquiring company Y); the system is able to return the correct verbs: *_bought* and *_purchase*, which are the inverse forms of the verbs *bought* and *purchase* (as in *is bought by* and *is purchased by*). Of practical importance is the fact that PIDGIN can learn verbs representing relations in Freebase and Yago2 whose facts are created manually or extracted via carefully constructed regular-expression matching. We can now use these verbs to automate an extraction process for the ontologies used by Freebase and Yago2.

5.7.3 Learning New Relation Instances

Here we examine the accuracy of automatically extracting new relation instances from text based on assignment of relation labels on NP pair nodes. We can estimate the probability that an NP-pair $\langle np_1, np_2\rangle$ belongs to a relation r, by estimating $P(\langle np_1, np_2\rangle|r) \approx \frac{\hat{Y}(\langle np_1, np_2\rangle, r)}{\sum_{\langle np'_1, np'_2\rangle} \hat{Y}(\langle np'_1, np'_2\rangle, r)}$. For each relation we can then return a ranked list of NP-pairs that can belong to the relation, some of which may not already be in the KB and can be proposed as new instances for the relation. We return the top 100 NP-pairs for each relation, pick those that are not already in the KB and evaluate the average precision. Consistent with our re-

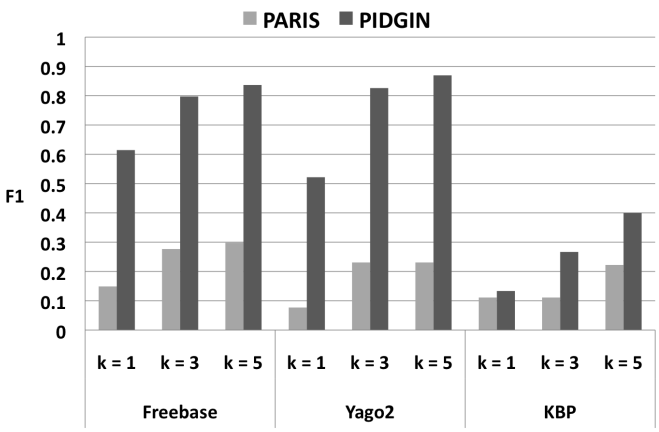


Figure 6: F1 scores @k = 1, 3 and 5 when typing relations in Freebase, Yago2, and KBP with NELL categories, comparing performance of PARIS (light grey) and PIDGIN (grey). (See Section 5.7.1 for details)

Knowledge Base	Relation	Learned Verbs
Freebase	/sports/sports_team/ arena_stadium	played at, played in, defeated at, will host at, beaten at
	/medicine/medical_ treatment/side_effects	may promote, can cause, may produce, is worsen, exacerbate
NELL	drugPossiblyTreats PhysiologicalCondition	treat, relieve, reduce, help with, can help alleviate
	politicianHoldsOffice	serves as, run for, became, was elected
Yago2	actedIn	played in, starred in, starred, played, portrayed in
	isMarriedTo	married, met, date, wed, divorce

Table 5: Examples of relation-verb pairs automatically learned by PIDGIN. Although we use verb stems in experiments, for better readability, we present the original non-stemmed forms of the same verbs above. (See Section 5.7.2 for details)

sults so far, precision of proposed new instances improves with the size of the KB. New instances proposed for Freebase have the highest average precision of 73.9% while new instances proposed for NELL, a smaller KB, have an average precision of 65.5%. Correct new instances discovered include ⟨*ratatouille*, *brad_bird*⟩ for Freebase */film/directed_by*, and ⟨*wwl*, *cbs*⟩ for NELL *televisionStationAffiliatedWith*. In the future, we want to use verb patterns that PIDGIN learns for relations in Section 5.7.2 to extract new NP pairs belonging to the relations from any free text.

6. RELATED WORK

Ontology Alignment has received considerable attention in prior research, please see [16] for a recent survey. PARIS [17] is a recently proposed state-of-the-art ontology alignment system that is most related to PIDGIN. PARIS primarily relies on instance overlap-based cues to align instances, categories, and relations from two KBs. This can be problematic in many cases of practical interest where there is sparsity in instance overlap, and noise in the data. PIDGIN overcomes these limitations using of natural language text as interlingua and graph-based self-supervised learning. In extensive comparisons, we find that PIDGIN significantly outperforms PARIS on the ontology alignment task.

Other prior work on ontology alignment have primarily relied on lexical and structural matching of the elements (i.e., categories or relations) in the ontology only [16], with the exception of a few that also consider instance similarities [17, 12]. And in most cases, category alignment has been the sole focus. PIDGIN exploits instance similarities, and aligns both categories and relations.

Low alignment recall (≈ 30%) in real-world datasets has been a common problem in most previous ontology alignment systems [10]. One way to improve recall is through iterative matching to increase completeness of alignments [2, 10, 12], which is similar in spirit to PIDGIN's use of label propagation for alignment inference. Background knowledge has been found to be useful in improving recall [16]. Most systems that use background knowledge share a common theme: to expand elements' features before lexical similarity computation. WordNet or WordNet-like data is a popular choice for background knowledge, with exceptions such as [9] that use documents related to instances to create lexical resource for the elements and use cosine similarity measures on the lexicons. Other works use background knowledge to construct additional paths to align elements. These works however, require background knowledge to be provided in the form of formal ontology [2, 15], which is not always available. An exception is [13] which uses unstructured documents and co-occurrence of category instances in documents to align categories. However, they only align categories. Co-occurrences of relation instances (i.e., a pairs of concept-pairs) in documents is significantly sparser compared to co-occurrence of concept pairs (as used in previous work). This sparsity can lead to low recall of their relation alignment. PIDGIN overcomes these limitations by exploiting a large coverage, schema-free natural language text corpus as interlingua. However, PIDGIN is flexible enough to incorporate other (and multiple) types of background knowledge (e.g., formal ontology) whenever available.

Although increasing the amount of background knowledge has been shown to improve alignment further [1], combining these pieces of information in prior work requires careful consideration due to the different structures of the background knowledge [2]. PIDGIN's graph construction provides a flexible way of integrating several background knowledge easily without restrictive constraints. Furthermore, integration of alignments obtained from multiple background knowledge sources has been rather ad-hoc in prior work [16]. In contrast, PIDGIN integrates heterogeneous evidence in the graph automatically and in a principled manner via its joint inference and optimization process.

Most previous systems employing iterative matching have relied on hand-written rules, with the exception of S-match [10]. However, S-match uses ontology-level information only, without considering instances. In PIDGIN, iterative matching is automatically obtained via the transitivity of inference in label propagation (MAD). Another system that uses label

propagation for ontology alignment is [20]. However, they do not incorporate any interlingua in their approach.

One approach that may strike resemblance to PIDGIN is Similarity Flooding [14], but there are critical differences. Firstly, it is not clear how natural language-based interlingua which do not have any well-defined schema can be incorporated into Similarity Flooding. PIDGIN is flexible enough to incorporate such side information, and improve resulting ontology alignment. Secondly, Similarity Flooding requires both ontologies to have similar edge types, while PIDGIN doesn't impose any such strong requirement. Thirdly, Similarity Flooding is also sensitive to noise in the ontology since the cross join of elements amplifies a noisy element in one ontology to elements in other ontologies. PIDGIN is more robust to noise as it doesn't multiply the effect of a noisy alignment, and instead integrates it with other competing alignments to produce a final alignment.

7. CONCLUSION

In this paper we introduce PIDGIN, a novel, flexible, and scalable approach to automatic alignment of real-world KB ontologies, demonstrating its superior performance at aligning large real-world KB ontologies including those of NELL, Yago and Freebase. The key idea in PIDGIN is to align KB ontologies by integrating two types of information: relation instances that are shared by the two KBs, and mentions of the KB relation instances across a large text corpus.

PIDGIN uses a natural language web text corpus of 500 million dependency-parsed documents as interlingua and a graph-based self-supervised learning to infer alignments. To the best of our knowledge, this is the first successful demonstration of using such a large text resource for ontology alignment. PIDGIN is *self-supervised*, and does not require human labeled data. Moreover, PIDGIN can be implemented in MapReduce, making it suitable for aligning ontologies from large KBs.

We have provided extensive experimental results on multiple real world datasets, demonstrating that PIDGIN significantly outperforms PARIS, the current state-of-the-art approach to ontology alignment. We observe in particular that PIDGIN is typically able to improve recall over that of PARIS, without degradation in precision. This is presumably due to PIDGIN's ability to use text-based interlingua to establish alignments when there are few or no relation instances shared by the two KBs. Additionally, PIDGIN automatically learns which verbs are associated with which ontology relations. These verbs can be used in the future to extract new instances to populate the KB or identify relations between entities in documents. PIDGIN can also assign relations in one KB with argument types of another KB. This can help type relations that do not yet have argument types, like that of KBP. Argument typing can improve the accuracy of extraction of new relation instances by constraining the instances to have the correct types.

In the future, we plan to extend PIDGIN's capabilities to provide explanations for its inferred alignments. We also plan to experiment with aligning ontologies from more than two KBs simultaneously.

Acknowledgments

We thank the ClueWeb project (CMU) and the Hazy Research Group (http://hazy.cs.wisc.edu/hazy/) for their generous help with data sets; and to the anonymous reviewers for their constructive comments. We also thank Ndapandula Nakashole and other members of the NELL Research Group at CMU for their helpful comments. This research has been supported in part by DARPA (under contract number FA8750-13-2-0005), Google, and Fulbright. Any opinions, findings, conclusions and recommendations expressed in this paper are the authors' and do not necessarily reflect those of the sponsors.

8. REFERENCES

[1] Z. Aleksovski, M. Klein, W. Ten Kate, and F. Van Harmelen. Matching unstructured vocabularies using a background ontology. In *Managing Knowledge in a World of Networks*. 2006.

[2] Z. Aleksovski, W. ten Kate, and F. van Harmelen. Exploiting the structure of background knowledge used in ontology matching. 2006.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*. 2007.

[4] T. Berners-Lee, J. Hendler, O. Lassila, et al. The semantic web. *Scientific american*, 284(5), 2001.

[5] C. Bizer, T. Heath, and T. Berners-Lee. Linked data-the story so far. *IJSWIS*, 5(3), 2009.

[6] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.

[7] J. Callan, M. Hoy, C. Yoo, and L. Zhao. Clueweb09 data set. *boston.lti.cs.cmu.edu*, 2009.

[8] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.

[9] D. Fossati, G. Ghidoni, B. Di Eugenio, I. Cruz, H. Xiao, and R. Subba. The problem of ontology alignment on the web: a first report. In *Proceedings of the 2nd International Workshop on Web as Corpus*, pages 51–58. Association for Computational Linguistics, 2006.

[10] F. Giunchiglia, P. Shvaiko, and M. Yatskevich. Discovering missing background knowledge in ontology matching. *Frontiers in AI and Applicaitons*, 141, 2006.

[11] J. Hoffart, F. Suchanek, K. Berberich, E. Lewis-Kelham, G. De Melo, and G. Weikum. Yago2: exploring and querying world knowledge in time, space, context, and many languages. In *WWW*, 2011.

[12] Y. Jean-Mary and M. Kabuka. Asmov: Ontology alignment with semantic validation. In *Joint SWDB-ODBIS Workshop*, 2007.

[13] C. Kingkaew. Using unstructured documents as background knowledge for ontology matching. 2012.

[14] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE*, 2002.

[15] M. Sabou, M. d'Aquin, and E. Motta. Using the semantic web as background knowledge for ontology mapping. *Ontology Matching*, 2006.

[16] P. Shvaiko and J. Euzenat. Ontology matching: state of the art and future challenges. *TKDE*, 2012.

[17] F. M. Suchanek, S. Abiteboul, and P. Senellart. Paris: Probabilistic alignment of relations, instances, and schema. *PVLDB*, 5(3), 2011.

[18] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, 2007.

[19] P. P. Talukdar and K. Crammer. New regularized algorithms for transductive learning. *ECML*, 2009.

[20] P. P. Talukdar, Z. G. Ives, and F. Pereira. Automatically incorporating new sources in keyword search-based data integration. In *SIGMOD*, 2010.