## Table of Contents


1 Abstract....3

2 Introduction....3

3 Problem Definition....4

3.1 Ontology Definition....4

3.2 Generic Problem Definition....4

4 PIDGIN....5

4.1 Modified Problem Statement....5

4.2 Graph Construction....6

4.2.1 Relation-Entity Pair Edge....6

4.2.2 Entity-NP Pair Edge....6

4.2.3 NP Pair-Verb Edge....7

4.3 Alignment as classification over Graph....7

4.3.1 Equivalence Alignment....7

4.3.2 Subsption Alignment....7

4.3.3 Category Alignment....8

5 Datasets....8

5.1 Yago....8

5.2 NELL....8

5.3 Freebase....9

5.4 ClueWeb SVO Triple....9

6 Implementation....9

6.1 Components....9

6.2 Architecture....10

7 Results....11

7.1 Relation Alignment....11

7.2 Relation Subsumption....12

8 Conclusions and Comments....12

8.1 Conclusion....12

8.2 Future Works....12

# 1 Abstract

One of the biggest challenge to the goal of Smentic Web is the integration of growing number of independently designed ontologies. [1] The goal of this independent study was to do survey and implement current state of the are approach of ontology alignment of Knowledge Bases PIDGIN: Ontology Alignment using Web Text as Interlingua[2]. I have implemented alignment and subsumption score for NELL[3] and Freebase[4] and NELL and Yago[5].

In this report I summarize the technique employed by the both the papers, discuss it details the PIDGIN approach for alignment, implementation details, architecture employed. I will also breifly cover the code organization at top level. At the end I will summerize the results I got and the scope for futurework.

# 2 Introduction

Over the last few years several large publicly available Knowledge Bases have come up. Example of such Knowledge Bases are DBPedia, Freebase, Yago. In addition to these general purpose KBs, we also have a series of domain specific knowledge bases like MusicBrainz, IMDB, GeoNames. Many of theses knowledge Base, both large and small contain complementing data. For example a gneral ontology may know who discovered a certain enzyme, whereas a biological knowledge base would know its functions and properties. However as these KBs are often independently developed, using different terminologies and ontological structure for categories and relations. In this respect, they can be seem as isolated island of knowledge. There is a need to align these hetrogeneous KBs to create a universal Knowledge Base.

There has been a lot of emphasis and research on ontology alignment of Knowledge Base lately. Historically the approches towards ontology alignment have been focused on either instance machine(A-Box) arising naturally in record-linkage, duplicate detection and co refrence resolution, or claasees alignment(T-Box). PARIS and PIDGIN discussed here meet the need of mordern day complex Kbs and align align relations, categories,subrelations and instances. PARIS is a probabilistic matcher which uses overlap of instances between two relations from a pair of KBs as one of the primary cues to determine whether an equivalence or subsumption relationship exists between two relations(or categories). For the system that share instances like DPPedia, Yago and IMDB. This technique fails when the KBs shate no or very few instances. PIDGIN's approach overcome this problem by using a side information in the form of a very large text corpus. PIDGIN maps literal of instances to literals in text as use the verb connecting these literals in text as relation. PIDGIN alies this extracted information along with the shared instances using graph based self-supervised learning technique to determine their final alignment.

In section 3 we mathematically define the generic problem statement of ontology alignment in Knowledge Bases. In section 4 we will discuss the PIDGIN approach in detail. In section 5 we discuss the datasets used. Section 6 covers the implementation and design of the system. In section 7 we will discuss the results and section 8 will talk about the future work and components yet to be implemented and approach toward the problem.

# 3 Problem Definition

In this section we will lay out the terminalogy used and present the mathetical model of the problem statement at hand.

## 3.1 Ontology Definition

PIDGIN paper defines a Knowledge Base as a 6 tuple.

$$(C, O_C, I_C, R, O_R, I_R)$$

where

$C$ is the set of categories e.g. (*athelete, sports)*

$O_C$ is the category ontology(heirarichal relation among categories e.g. (*athelete* is a subset of a *person*)

$I_C$ is the set of entity category pairs e.g. (*Tiger Wood, athelete)*

$R$ is a set of relations e.g. (*atheletePlaysSport(athelete, sports)*

$O_R$ is relation ontology (heirarichal relation among relations)
e.g. (*ceoOf(person, Company)* is a special case of
*worksFor(person, Company))*

$I_R$ is a set of entity-relation-entity triple for relation in $R$.
e.g. (*Tiger Woods, atheletePlaysSport, Golf).*

$O_c$ and $O_R$ can be empty depending on the schema of the knowledge base. 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 $c_1, c_2 \in C$

## 3.2 Generic Problem Definition

Let $A(R_1, R_2) = (r_1, a, r_2, w) | r_1 \in R_1, a \in , \subseteq, r_2 \in R_2, w \in R$ where $\equiv$ signifies relation equivalence, $\subseteq$ less general, and $R$ is set of real numbers. We similarly define the category alignment as $A(C_1, C_2) = (c_1, a, c_2, w) | c_1 \in C_1, a \in , \subseteq, c_2 \in C_2, w \in R$

Given two knowledge bases (Kbs) $K(C_1, O_{C_1}, I_{C_1}, R_1, O_{R_1}, I_{R_1})$ and $K(C_2, O_{C_2}, I_{C_2}, R_2, O_{R_2}, I_{R_2})$ discover the category and relation alignment $A(C_1, C_2)$

and $A(R_1, R_2)$ respectively.

# 4 PIDGIN

The PIDGIN also tries to address the issue of alignment of among the Knowledge Bases that share no or very few instances. It tries to do this using web text as a side information. This side information is used to map literal attached to instances across the KB using the verb phrases in web text.

PIDGIN formulates this problem as classification problem over an appropriately constructed graph. There are two stages in this formuation.

1. Graph Construction:

   In first stage we construct a graph using $KB_1$ and $KB_2$ and $D$, we first construct a graph. This graph has three kind of edges namely

   - Relation-Entity Pair Edge
   - Entity Pair-NP Pair Edge
   - NP-Pair Verb Edge

2. Alginment as Classification over Graph

   For graph generated in stage 1, the PIDGIN sets this up as a node classification problem using Modified Adsorbtion(MAD) algorithm[6]. It first associates label corresponding to alignment and subsumption matching to each and every node in $KB_1$ . Using this as seed data, then PIDGIN propogates this label using MAD. Based on the relation scores assigned to relation in PIDGIN calculates the alignment of two relations. Catogery alignment is calculated as a by product.

## 4.1 Modified Problem Statement

Let $D$ be the subject-verb-object(SVO) based interlingua consisting of tuples of the norm $(np_1, v, np_2, w)$ where $np_1$ and $np_2$ are noun phrases corresponding to subject and object respectively, $v$ is a verb, and $w \in R_+$ is the normalized count of this tuple in a large text corpus.

Given two knowledge bases (Kbs) $K(C_1, O_{C_1}, I_{C_1}, R_1, O_{R_1}, I_{R_1})$ and $K(C_2, O_{C_2}, I_{C_2}, R_2, O_{R_2}, I_{R_2})$ 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.

## 4.2 Graph Construction

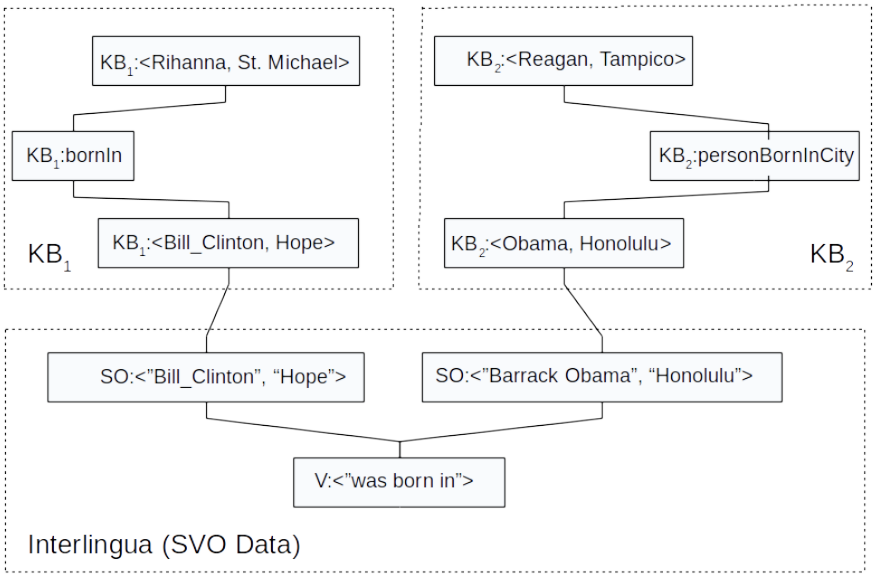


Figure1: Example Graph[1]

As mentioned above the the three relations are defined as follows:

### *4.2.1 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 $b = KB_k :< e_1, e_2 >$ to V , add the edge (a, b) to E, and set $W_{a,b} = W_{b,a} = 1.0$. In Figure 1, $(KB_1 : bornIn, KB_1 :< BillClinton, Hope >)$ is an example of such an edge.

### *4.2.2 Entity-NP Pair Edge*

For each k ∈ {1, 2}, PIDGIN defines,

$Q_k = \{SO :< np_1, np_2 > | np_1 \in N(e_1), np_2 N(e_2), KB_k :< e_1, e_2 > \in V\}$
where $N(e)$ returns the set of NPs corresponding to entity $e$.

Now, for each $b = KB_k :< e_1, e_2 > \in V \forall k \{1, 2\}$ they define,

$$Q_b = \{SO :< np_1, np_2 > | np_1 \in N(e_1), np_2 \in N(e_2)$$
$$(SO :< np_1, np_2 > \in Q_1 \cap Q_2 \exists v s.t. (np_1, v, np_2) \in D)\}$$

That is add an entity np pair edge if either noun phrase pair corresponding to entity pair is found in web text or if noun phrase pair matches among the KBs. The edge weight is set to 1.0 . In Figure 1, $(KB_1 :< BillClinton, Hope >, SO :< BillClinton, Hope >)$ is an example of such an edge.

#### 4.2.3 NP Pair-Verb Edge

Let $Q$ be the union of all $Q_b$ sets defined above. They define

$T = \{v | SO :< np_1, np_2 > \in Q, (np1, v, np2, w) \in D\}$. and set $V = V \cup T$ ,

and $E = E \cup \{(q, v) | q \in Q, v \in T\}$ with the edge weight set to w.

In Figure 1(b), $(SO :< BillClinton, Hope >, wasbornin)$ is an example of such an edge.

In other words, this is the edge between all the verbs for which noun phrase are present in either $KB_1$ or $KB_2$

### 4.3 Alignment as classification over Graph

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 $KB_1 : r_1$ 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 r 1 : This label is injected as seed to nodes $KB_1 : s \in V | s \in childrenOf(O_1, r_1)$. In other words, $l_{r_1}$ 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, $r_1$ .

For each node in Graph $G$, MAD algorithm will generate $Y_1(v, l)$ which is a $R_{nX|L_1|}$ matrix. So each nodes get score corresponding to each label injected to seed nodes. Now using these labels we can define all scores.

#### 4.3.1 Equivalence Alignment

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

$$A \equiv (R_1, R_2, Y_1, Y_2) = \{(r_1, \equiv, r_2, Y_1(r_2, l_{r_1}) X Y_2(r_1, l_{r_2})) | \\ r_1 \in R_1, r_2 \in R_2, l_{r_1} \in L_1, l_{r_2} \in L_2\}$$

where $Y_1$ and $Y_2$ are the label score matrices estimated by MAD. We define final equivalence alignment score as $Y_{\equiv}(r_1, r_2) = Y_1(r_2, l_{r_1}) X Y_2(r_1, l_{r_2})$.

#### 4.3.2 Subsption Alignment

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

$$A \subseteq (R_1, R_2, Y_1, Y_2) = \{(r_2, \subseteq, r_1, Y_1(r_2, l_{r_1}) | \\ r_1 \in R_1, r_2 \in R_2, l_{r_1} \in L_1\}$$

PIDGIN referes to this score as $Y_{\subseteq}(r_1, r_2) = Y_1(r_2, l_{r_1})$

#### 4.3.3 Category Alignment

we have $Y_{\subseteq}(r_1, r_2)$ as the relation equivalence score estimated by PIDGIN for relations $r_1 \in R_1$ and $r_1 \in R_1$ . PIDGIN uses this estimate to establish category equivalence alignments as follows. Given a relation r, let $Dom(r)$ and $Ran(r)$ be its domain and range categories, i.e., categories of the two entities connected by this relation.

Pidgin define,

$$H_{\equiv}^{Dom}(c_1, c_2) = \sum_{\substack{r_1 \in R_1, c_1 = Dom(r_1), \\ r_2 \in R_2, c_2 = Dom(r_2)}} = Y_{\subseteq}(r_1, r_2)$$

$$H_{\equiv}^{Ran}(c_1, c_2) = \sum_{\substack{r_1 \in R_1, c_1 = Ran(r_1), \\ r_2 \in R_2, c_2 = Ran(r_2)}} = Y_{\subseteq}(r_1, r_2)$$

Final category equivalence alignments is defined as,

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

# 5 Datasets

| Knowledge Base | Relations | Instances |
|---|---|---|
| Freebase | 79 | 7,450,452 |
| NELL | 254 | 2,882,193 |
| Yago | 23 | 1,770,163 |

Table 1: Relation and Instances per KB

## 5.1 Yago

Quoting Yago, YAGO2s is a huge semantic knowledge base, derived from Wikipedia WordNet and GeoNames. Currently, YAGO2s has knowledge of more than **10 million entities** (like persons, organizations, cities, etc.) and contains more than **120 million facts** about these entities. Pidgin uses 23 relations from Yago which have known mapping in NELL.

## 5.2 NELL

It is a project by CMU in which the objective is to build a system to extract structrured information from unstructured data. This results in a knowledge base that mirros the content of the web. Currently NELL has 2,173,184 instances.

## 5.3 Freebase

Freebase is a large collaborative knowledge base consisting of metadata composed mainly by its community members. It is an online collection of structured data harvested from many sources, including individual 'wiki' contributions.[2] Freebase aims to create a global resource which allows people (and machines) to access common information more effectively. Freebase contains **43,647,507 topic(instances)** and **2,483,419,016 facts**.

## 5.4 ClueWeb SVO Triple

This dataset is again provided by CMU Read the Web Project. It currently contains 604 million triples extracted from the entire dependency parsed ClueWeb09 dataset.

# 6 Implementation

## 6.1 Components

Following packages/tools were used in building PIDGIN.

- Mongo DB server (Nell Data doesn't follow rdf format and hence is kept in a regular DB)
- Jena Fuseki and TDB (Yago and Freebase being rdf complaint are hosted as SPARQL servers. The storage format for indexed data is TDB. For Yago, the jena complaint data is available at Yago website. For freebase the data set was coverted to TDB format and then hosted as a Jena based Sparql endpoint
- Solr 4.6: Solr server was used to host SVO data. This helps in Noun Phrase Pair query.
- Junto Toolkit: The MAD implementation by Partha etal. It is available on github. It contains for a scala implementation and for bigger datasets, it also provides MapReduce implementation. For Yago and NELL alignment, scala implementation did nell, for Freebase Hadoop based method were used.

Folllowing were the individual components of the system:

- NP Server: This is a Solr based server used to query Noun Phrases from web text. The server contained more than 600 million SVO triples.
- Yago2 Server: Given that Yago conforms to rdf/rdfs schema, we used a minimal SPARQL server to query yago2 triples. We only queried 23 relations for which alignment in NELL was hand annotated.
- Freebase  Server: The freebase provides rdf dumps which can be used with an rdf server. The dump contains close to 22 billion facts. Dump was converted to TDB format and was used by running Jena Fuseki Sparql server.
- NELL Server: Considering that NELL output is not in rdf format, we needed a server in traditional sense to query results. I used mongoDB because of its simplicity and familiarlity

- Junto Toolkit: Learning over graph was handled by Junto library provided by Partha. It has two implementation of the algorithm, one scala based to run on a single machine another Hadoop based to bigger datasets. Yago and NELL alignment was done using scala implementation but Freebase and NELL needed Hadoop based approach due to the size of the graph.

## 6.2 Architecture

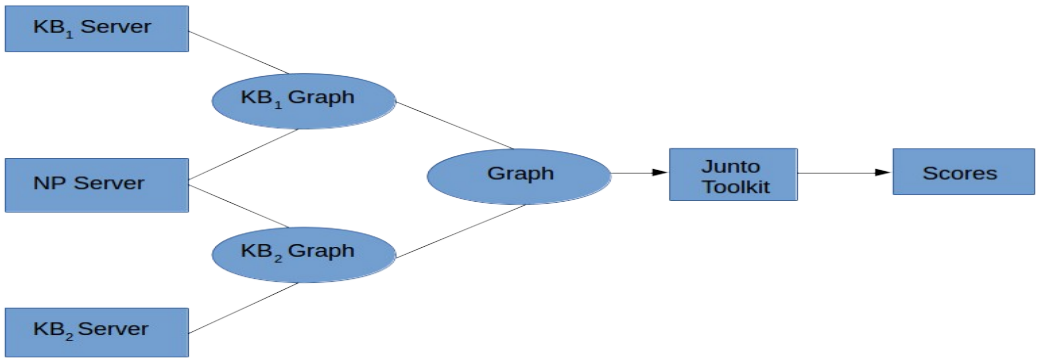


Fig 2: Simplistic view of system architecture

Following defines the process followed in aligning two Knowledge Bases:

1. **Graph Generation**

   First stage of PIDGIN is to generate the graph to do label propogation. We add three types of edges in two different steps.

   1. As a first step we query each KB for list of relations to be aligned. In this stage we dump as Entity Pair-Relation Edges. In addition to this, we also dump all Noun Phrase pairs associated with each entity pair to be used in stage two.

   2. In stage two, we check noun phrase pairs dumped by each knowledge base for match. In case there is a match, we add Entity Pair Noun Phrase Pair Edge. This is first type of Entity Pair- Noun Phrase Pair Edge, the other type, the one due to Noun Phrase corresponding to entity Pair existing in web text, will be added in stage three

   3. In stage three, we query SVO server for matching Noun Phrases. In case Noun Phrase queried exists, we add Entity Pair – Noun Phrase Pair edge corresponding to Noun Phrase Pair. At the same time we add Noun Phrase Pair Verb Edge.

2. **Alignment over Graph**

   Once we have the graph, next stage is to do controlled label propogation on Graph. For this we used Junto Toolkit which is an open source implementation of MAD algorithm available on Github. This toolkit can be used in two mode, standalone mode

which is implemented in scala or as a Hadoop job. We used it in standalone mode to compute scores for Yago and NELL and in hadoop more for calculating Freebase and NELL scores. The Hadoop job need the Graph to be in NodeFactored format instead of Edge Factored format., hence it needs an extra step to convert Graph in required format.

3. **Score Computation**

For relation Alignment we need to run the step 2 twice, once propogating $KB_1$ labels onto relations in $KB_2$ and again to propograte $KB_2's$ labels to $KB_1$'s relations.

Once we have $Y_1(r_1, l_2)$ and $Y_2(r_2, l_1)$, we compute Alignment score $Y(r_1, r_2) = Y_1(r_1, l_2)XY_2(r_2, l_1)$ .

For sumsumption relations, we just propograte subsumption labels from NELL to Freebase and Yago, and calcultate $Y(r_1, l'_2)$.

# 7 Results

## 7.1 Relation Alignment

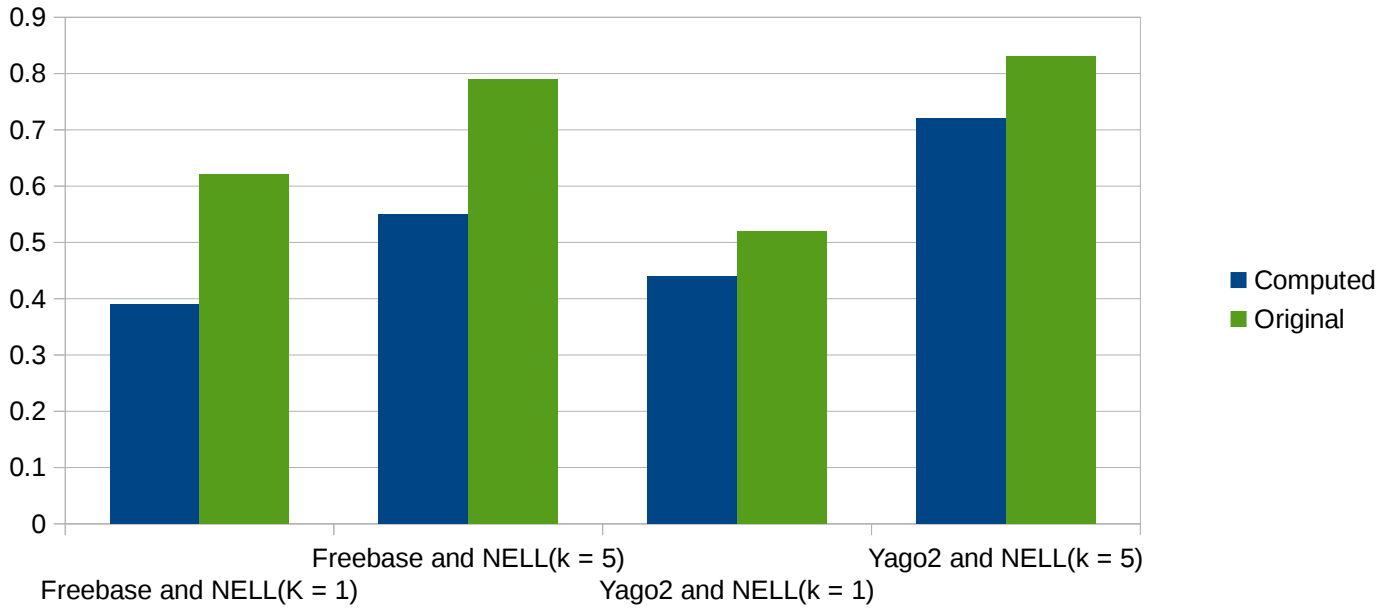


Figure 3: Relation Alignment for Freebase & NELL and Yago & NELL at K = 1 and K = 5

| KB Pair | Precision | Recall | F1 |
|---|---|---|---|
| Freebase & NELL | 0.3846 | 0.3797 | 0.3821 |
| Yago & NELL | 0.4545 | .0.4343 | 0.4444 |

Table2: Relation alignment score for K=1

## 7.2 Relation Subsumption

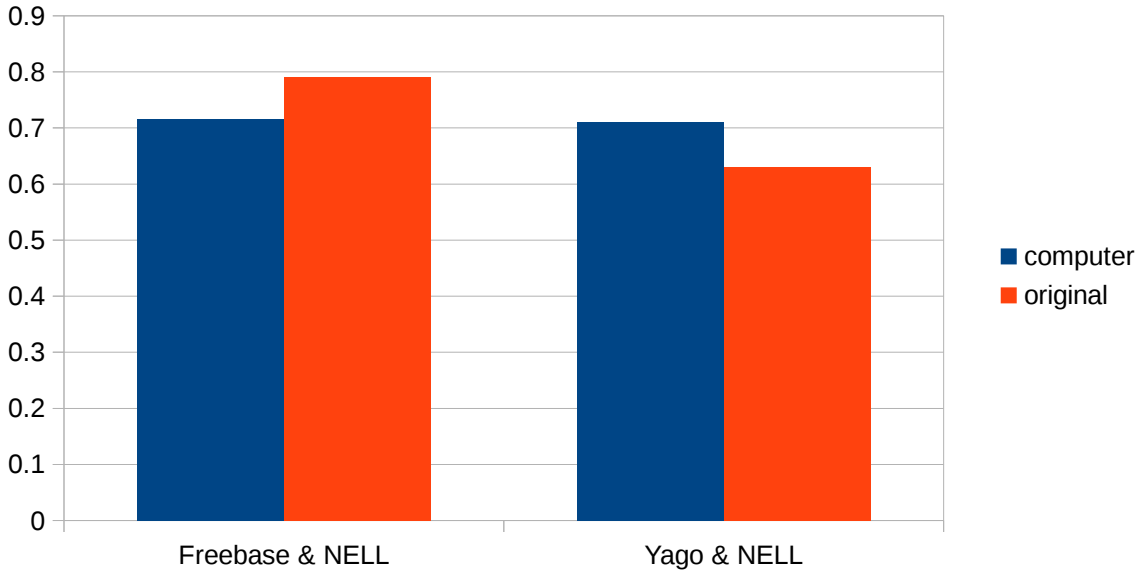


Figure 4: Relation Subsumption for Freebase & NELL and Yago & NELL

| KB Pair | Precision | Recall | F1 |
|---|---|---|---|
| Freebase & NELL | 0.7272 | 0.7088 | 0.7179 |
| Yago & NELL | 0.7619 | 0.6956 | 0.7272 |

Table 2: Relation subsumption score for Freebase & NELL and Yago & NELL

# 8 Conclusions and Comments

## 8.1 Conclusion

With this project, I have been able to implement the pipeline required to replicate PIDGIN results as well as to enhance the future work in this direction. The results for Yago and NELL alignment as almost in the same range. There is a need to improve the results for Freebase and NELL alignment. One of the area to look into to improve results is to get the graph from CMU group and look into NELL relation I might have missed or included extra. Category alignment couldn't be calculated for either NELL and Freebase or NELL and Yago because of lack of time.

## 8.2 Future Works

This pipeline can be used to generate the by products of PIDGIN, especially the by products of PIDGIN approach. Some of the by product like verbs for relation can be used to write a Natural Language Processing query layer over individual or unified KBs. At the same time examining verb entity pair relation can help is learning new relations too.

**References:**

1 Bizer, Christian, Tom Heath, and Tim Berners-Lee. "Linked data-the story so far." International journal on semantic web and information systems 5.3 (2009): 1-22.

2 Wijaya, Derry, Partha Pratim Talukdar, and Tom Mitchell. "PIDGIN: ontology alignment using web text as interlingua." Proceedings of the 22nd ACM international conference on Conference on information & knowledge management. ACM, 2013.

3 Carlson, Andrew, et al. "Toward an Architecture for Never-Ending Language Learning." AAAI. 2010.

4 Bollacker, Kurt, et al. "Freebase: a collaboratively created graph database for structuring human knowledge." Proceedings of the 2008 ACM SIGMOD international conference on Management of data. ACM, 2008.

5 Suchanek, Fabian M., Gjergji Kasneci, and Gerhard Weikum. "Yago: a core of semantic knowledge." Proceedings of the 16th international conference on World Wide Web. ACM, 2007.

6 Talukdar, Partha Pratim, and Koby Crammer. "New regularized algorithms for transductive learning." Machine Learning and Knowledge Discovery in Databases. Springer Berlin Heidelberg, 2009. 442-457.