FactRank: Random Walks on a Web of Facts

Alpa Jain
Yahoo! Labs
alpa@yahoo-inc.com

Patrick Pantel
Microsoft Research
ppantel@microsoft.com

Abstract

Fact collections are mostly built using semi-supervised relation extraction techniques and wisdom of the crowds methods, rendering them inherently noisy. In this paper, we propose to validate the resulting facts by leveraging global constraints inherent in large fact collections, observing that correct facts will tend to match their arguments with other facts more often than with incorrect ones. We model this intuition as a graph-ranking problem over a fact graph and explore novel random walk algorithms. We present an empirical study, over a large set of facts extracted from a 500 million document webcrawl, validating the model and showing that it improves fact quality over state-of-the-art methods.

1 Introduction

Fact bases, such as those contained in *Freebase*, *DBpedia*, *KnowItAll*, and *TextRunner*, are increasingly burgeoning on the Internet, in government, in high tech companies and in academic laboratories. Bar the accurate manual curation typified by Cyc (Lenat, 1995), most fact bases are built using either semi-supervised techniques or wisdom of the crowds techniques, rendering them inherently noisy. This paper describes algorithms to validate and re-rank fact bases leveraging global constraints imposed by the semantic arguments predicated by the relations.

Facts are defined as instances of *n*-ary typed relations such as *acted-in*⟨*movie*, *actor*⟩, *director-of*⟨*movie*, *director*⟩, *born-in*⟨*person*, *date*⟩, and *buy*⟨*person*, *product*, *person*⟩. In all but very small fact bases, relations share an argument type, such as *movie* for the relations *acted-in* and *director-of* in the above example. The hypothesis explored in this paper is that when two fact instances from two relations share the same value for a shared argument type, then the validity of both facts should be increased. Conversely, we also hypothesize that an incorrect fact instance will tend to match a shared argument with other facts far less frequently. For example, consider the following four facts from the relations *acted-in*, *director-of*, and *is-actor*:

t_1: *acted-in*⟨Psycho, Anthony Perkins⟩
t_2: **acted-in*⟨Walt Disney Pictures, Johnny Depp⟩
t_3: *director-of*⟨Psycho, Alfred Hitchcock⟩
t_4: *is-actor*⟨Anthony Perkins⟩

Our confidence in the validity of t_1 increases with the knowledge of t_3 and t_4 since the argument *movie* is shared with t_3 and *actor* with t_4. Similarly, t_1 increases our confidence in the validity of t_3 and t_4. For t_2, we expect to find few facts that will match a *movie* argument with *Walt Disney Pictures*. Facts that share the *actor* argument *Johnny Depp* with t_2 will increase its validity, but the lack of matches on its *movie* argument will decrease its validity.

In this paper, we present *FactRank*, which formalizes the above intuitions by constructing a fact graph and running various random walk graph-ranking algorithms over it to re-rank and validate the facts. A collection of facts is modeled in the form of a graph where nodes are fact instances and edges connect nodes that have the same value for a shared argument type (e.g., t_1 would be linked by an edge to both t_3 and t_4.) Given a graph representation of facts, we explore various random walk algorithms to propagate our confidence in individual facts through the web of facts. We explore algorithms such as PageRank (Page et al., 1999) as well as propose novel algorithms that leverage several unique characteristics of fact graphs. Finally, we present an empirical analysis, over a large collection of facts extracted from a 500 mil-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 501–509,
Beijing, August 2010

lion document webcrawl, supporting our model and confirming that global constraints in a fact base can be leveraged to improve the quality of the facts. Our proposed algorithms are agnostic to the sources of a fact base, however our reported experiments were carried over a state-of-the-art semi-supervised extraction system. In summary, the main contributions of this paper are:

- We formalize the notion of ranking facts in a holistic manner by applying graph-based ranking algorithms (Section 2).
- We propose novel ranking algorithms using random walk models on facts (Section 3).
- We establish the effectiveness of our approach through an extensive experimental evaluation over a real-life dataset and show improvements over state-of-the-art ranking methods (Section 4).

2 Fact Validation Revisited

We denote an n-ary ***relation*** r with typed arguments $t_1, t_2, \cdots, t_n$ as $r\langle t_1, t_2, \cdots t_n\rangle$. In this paper, we limit our focus to unary and binary relations. A ***fact*** is an instance of a relation. For example, *acted-in*$\langle$*Psycho, Anthony Perkins*$\rangle$ is a fact from the *acted-in*$\langle movie, actor\rangle$ relation.

Definition 2.1 [Fact base]: *A* fact base *is a collection of facts from several relations. Textrunner and Freebase are example fact bases (note that they also contain knowledge beyond facts such as entity lists and ontologies.)* □

Definition 2.2 [Fact farm]: *A* fact farm *is a subset of interconnected relations in a fact base that share arguments among them.* □

For example, consider a fact base consisting of facts for relations involving movies, organizations, products, etc., of which the relations *acted-in* and *director-of* could form a MOVIES fact farm.

Real-world fact bases are built in many ways. Semi-supervised relation extraction methods include *KnowItAll* (Etzioni et al., 2005), *TextRunner* (Banko and Etzioni, 2008), and many others such as (Riloff and Jones, 1999; Pantel and Pennacchiotti, 2006; Paşca et al., 2006; Mintz et al., 2009). Wisdom of the crowds methods include DBpedia (Auer et al., 2008) and Freebase which extracts facts from various open knowledge bases and allow users to add or edit its content.

Most semi-supervised relation extraction methods follow (Hearst, 1992). Starting with a relatively small set of seed facts, these extractors iteratively learn patterns that can be instantiated to identify new facts. To reflect their confidence in an extracted fact, extractors assign an *extraction score* with each fact. Methods differ widely in how they define the *extraction score*. Similarly, many extractors assign a *pattern score* to each discovered pattern. In each iteration, the highest scoring patterns and facts are saved, which are used to seed the next iteration. After a fixed number of iterations or when a termination condition is met, the instantiated facts are ranked by their *extraction score*.

Several methods have been proposed to generate such ranked lists (e.g., (Riloff and Jones, 1999; Banko and Etzioni, 2008; Matuszek et al., 2005; Pantel and Pennacchiotti, 2006; Paşca et al., 2006). In this paper, we re-implement the large-scale state-of-the-art method proposed by Paşca et al. (2006). This pattern learning method generates binary facts and computes the extraction scores of a fact based on (a) the scores of the patterns that generated it, and (b) the distributional similarity score between the fact and the seed facts. We computed the distributional similarity between arguments using (Pantel et al., 2009) over a large crawl of the Web (described in Section 4.1). Other implementation details follow (Paşca et al., 2006).

In our experiments, we observed some interesting ranking problems as illustrated by the following example facts for the *acted-in* relation:

id:	Facts (#Rank)
t_1:	*acted-in*⟨Psycho, Anthony Perkins⟩ (#26)
t_2:	**acted-in*⟨Walt Disney Pictures, Johnny Depp⟩ (#9)

Both t_1 and t_2 share similar contexts in documents (e.g., ⟨*movie*⟩ *film starring* ⟨*actor*⟩ *and* ⟨*movie*⟩ *starring* ⟨*actor*⟩), and this, in turn, boosts the pattern-based component of the extraction scores for t_1. Furthermore, due to the ambiguity of the term *psycho*, the distributional similarity-based component of the scores for fact t_2 is also lower than that for t_1.

Relations	id : Facts
acted-in	t_1 : ⟨Psycho, Anthony Perkins⟩
	t_2 : *⟨Walt Disney Pictures, Johnny Depp⟩
director-of	t_3 : ⟨Psycho, Alfred Hitchcock⟩
producer-of	t_4 : ⟨Psycho, Hilton Green⟩
is-actor	t_5 : ⟨Anthony Perkins⟩
	t_6 : ⟨Johnny Depp⟩
is-director	t_7 : ⟨Alfred Hitchcock⟩
is-movie	t_8 : ⟨Psycho⟩

Table 1: Facts share arguments across relations which can be exploited for validation.

Our work in this paper is motivated by the following observation: the ranked list generated by an individual extractor does not leverage any global information that may be available when considering a fact farm in concert. To understand the information available in a fact farm, consider a MOVIES fact farm consisting of relations, such as, *acted-in*, *director-of*, *producer-of*, *is-actor*, *is-movie*, and *is-director*. Table 1 lists sample facts that were generated in our experiments for these relations[1]. In this example, we observe that for t_1 there exist facts in foreign relations, namely, *director-of* and *producer-of* that share the same value for the *Movie* argument, and intuitively, facts t_3 and t_4 add to the validity of t_1. Furthermore, t_1 shares the same value for the *Actor* argument with t_5. Also, t_3, which is expected to boost the validity of t_1, itself shares values for its arguments with facts t_4 and t_7, which again intuitively adds to the validity of t_1. In contrast to this *web of facts* generated for t_1, the fact t_2 shares only one of its argument value with one other fact, i.e., t_6.

The above example underscores an important observation: *How does the web of facts generated by a fact farm impact the overall validity of a fact?* To address this question, we hypothesize that facts that share arguments with many facts are more reliable than those that share arguments with few facts. To capture this hypothesis, we model a web of facts for a farm using a graph-based representation. Then, using graph analysis algorithms, we propagate reliability to a fact using the scores of other facts that recursively connect to it.

Starting with a fact farm, to validate the facts in each consisting relation, we:

(1) Identify arguments common to relations in the farm.
(2) Run extraction methods to generate each relation.
(3) Construct a graph-based representation of the extracted facts using common arguments identified in Step (1) (see Section 3.1 for details on constructing this graph.)
(4) Perform link analysis using random walk algorithms over the generated graph, propagating scores to each fact through the interconnections (see Section 3.2 for details on various proposed random walk algorithms).
(5) Rank facts in each relation using the scores generated in Step (4) or by combining them with the original extraction scores.

For the rest of the paper, we focus on generating better ranked lists than the original rankings proposed by a state-of-the-art extractor.

3 FactRank: Random Walk on Facts

Our approach considers a fact farm holistically, leveraging the global constraints imposed by the semantic arguments of the facts in the farm. We model this idea by constructing a graph representation of the facts in the farm (Section 3.1) over which we run graph-based ranking algorithms. We give a brief overview of one such ranking algorithm (Section 3.2) and present variations of it for fact re-ranking (Section 3.3). Finally, we incorporate the original ranking from the extractor into the ranking produced by our random walk models (Section 3.4).

3.1 Graph Representation of Facts

Definition 3.1 *We define a* fact graph *FG(V, E), with V nodes and E edges, for a fact farm, as a graph containing facts as nodes and a set of edges between these nodes. An edge between nodes v_i and v_j indicates that the facts share the same value for an argument that is common to the relations that v_i and v_j belong to.* □

Figure 1 shows the fact graph for the example in Table 1 centered around the fact t_1.

Note on the representation: The above graph representation is just one of many possible options. For instance, instead of representing facts by nodes, nodes could represent the arguments of facts (e.g., *Psycho*) and nodes could be connected by edges if they occur together in a fact. The task of studying a "best" representation remains a future work direction. However, we believe that our proposed methods can be easily adapted to other such graph representations.

[1] The *is-actor*⟨actor⟩, *is-director*⟨director⟩, and *is-movie*⟨movie⟩ relations are equivalent to the relation *is-a*⟨c-instance, class⟩ where $class \in \{actor, director, movie\}$.

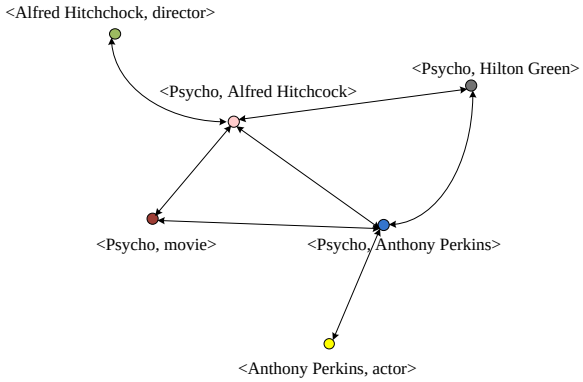

Figure 1: Fact graph centered around t_1 in Table 1.

3.2 The FactRank Hypothesis

We hypothesize that connected facts increase our confidence in those facts. We model this idea by propagating *extraction scores* through the fact graph similarly to how authority is propagated through a hyperlink graph of the Web (used to estimate the importance of a webpage). Several link structure analysis algorithms have been proposed for this goal, of which we explore a particular example, namely, PageRank (Page et al., 1999). The premise behind PageRank is that given the hyperlink structure of the Web, when a page v generates a link to page u, it confers some of its importance to u. Therefore, the importance of a webpage u depends on the number of pages that link to u and furthermore, on the importance of the pages that link to u. More formally, given a directed graph $G = (V, E)$ with V vertices and E edges, let $I(u)$ be the set of nodes that link to a node u and $O(v)$ be the set of nodes linked by v. Then, the importance of a node u is defined as:

$$p(u) = \sum_{v \in I(u)} \frac{p(v)}{|O(v)|} \quad (1)$$

The PageRank algorithm iteratively updates the scores for each node in G and terminates when a convergence threshold is met. To guarantee the algorithm's convergence, G must be irreducible and aperiodic (i.e., a connected graph). The first constraint can be easily met by converting the adjacency matrix for G into a stochastic matrix (i.e., all rows sum up to 1.) To address the issue of periodicity, Page et al. (1999) suggested the following modification to Equation 1:

$$p(u) = \frac{1-d}{|V|} + d \cdot \sum_{v \in I(u)} \frac{p(v)}{|O(v)|} \quad (2)$$

where d is a damping factor between 0 and 1, which is commonly set to 0.85. Intuitively, PageRank can be viewed as modeling a "random walker" on the nodes in G and the score of a node, i.e., PageRank, determines the probability of the walker arriving at this node.

While our method makes use of the PageRank algorithm, we can also use other graph analysis algorithms (e.g., HITS (Kleinberg, 1999)). A particularly important property of the PageRank algorithm is that the stationary scores can be computed for *undirected* graphs in the same manner described above, after replacing each undirected edge by a bi-directed edge. Recall that the edges in a fact graph are bi-directional (see Figure 1).

3.3 Random Walk Models

Below, we explore various random walk models to assign scores to each node in a fact graph *FG*.

3.3.1 Model Implementations

Pln: Our first method applies the traditional PageRank model to *FG* and computes the score of a node u using Equation 2.

Traditional PageRank, as is, does not make use of the *strength* of the links or the nodes connected by an edge. Based on this observation, researchers have proposed several variations of the PageRank algorithm in order to solve their problems. For instance, variations of random walk algorithms have been applied to the task of extracting important words from a document (Hassan et al., 2007), for summarizing documents (Erkan and Radev, 2004), and for ordering user preferences (Liu and Yang, 2008). Following the same idea, we build upon the discussion in Section 3.2 and present random walk models that incorporate the strength of an edge.

Dst: One improvement over ***Pln*** is to distinguish between nodes in *FG* using the extraction scores of the facts associated with them: extraction methods such as our reimplementation of (Paşca et al., 2006) assign scores to each output fact to reflect its confidence in it (see Section 3.2). Intuitively, a higher scoring node that connects to u should increase the importance of u more than a connection from a lower scoring node. Let $I(u)$ be the set of nodes that link to u and $O(v)$ be the set of nodes

linked by v. Then, if $w(u)$ is the extraction score for the fact represented by node u, the score for node u is defined:

$$p(u) = \frac{1-d}{|V|} + d \cdot \sum_{v \in I(u)} \frac{w(v) \times p(v)}{|O(v)|} \quad (3)$$

where $w(v)$ is the confidence score for the fact represented by v. Naturally, other (externally derived) extraction scores can also be substituted for $w(v)$.

Avg: We can further extend the idea of determining the strength of an edge by combining the extraction scores of *both* nodes connected by an edge. Specifically,

$$p(u) = \frac{1-d}{|V|} + d \cdot \sum_{v \in I(u)} \frac{avg(u,v) \times p(v)}{|O(v)|} \quad (4)$$

where $avg(u, v)$ is the average of the extraction scores assigned to the facts associated with nodes u and v.

Nde: In addition to using extraction scores, we can also derive the strength of a node depending on the number of *distinct* relations it connects to. For instance, in Figure 1, t_1 is linked to four distinct relations, namely, *director-of*, *producer-of*, *is-actor*, *is-movie*, whereas, t_2 is linked to one relation, namely, *is-actor*. We compute $p(u)$ as:

$$p(u) = \frac{1-d}{|V|} + d \cdot \sum_{v \in I(u)} \frac{(\alpha \cdot w(v) + (1-\alpha) \cdot r(v)) \times p(v)}{|O(v)|} \quad (5)$$

where $w(v)$ is the confidence score for node v and $r(v)$ is the fraction of total number of relations in the farm that contain facts with edges to v.

3.3.2 Dangling nodes

In traditional hyperlink graphs for the Web, dangling nodes (i.e., nodes with no associated edges) are considered to be of low importance which is appropriately represented by the scores computed by the PageRank algorithm. However, an important distinction from this setting is that fact graphs are sparse causing them to have valid facts with no counterpart matching arguments in other relation, thus rendering them dangling. This may be due to several reasons, e.g., extractors often suffer from less than perfect recall and they may miss valid facts. In our experiments, about 10% and 40% of nodes from *acted-in* and *director-of*, respectively, were dangling nodes.

Handling dangling nodes in our extraction-based scenario is a particularly challenging issue: while demoting the validity of dangling nodes could critically hurt the quality of the facts, lack of global information prevents us from systematically introducing them into the re-ranked lists. We address this issue by maintaining the original rank positions when re-ranking dangling nodes.

3.4 Incorporating Extractor Ranks

Our proposed random walk ranking methods ignore the ranking information made available by the original relation extractor (e.g., (Paşca et al., 2006) in our implementation). Below, we propose two ways of combining the ranks suggested by the original ranked list O and the re-ranked list G, generated using the algorithms in Section 3.3.

R-Avg: The first combination method computes the average of the ranks obtained from the two lists. Formally, if $O(i)$ is the original rank for fact i and $G(i)$ is the rank for i in the re-ranked list, the combined rank $M(i)$ is computed as:

$$M(i) = \frac{O(i) + G(i)}{2} \quad (6)$$

R-Wgt: The second method uses a weighted average of the ranks from the individual lists:

$$M(i) = \frac{w_o \cdot O(i) + (1 - w_o) \cdot G(i)}{2} \quad (7)$$

In practice, this linear combination can be learned; in our experiments, we set them to $w_o = 0.4$ based on our observations over an independent training set. Several other combination functions could also be applied to this task. For instance, we explored the *min and* max functions but observed little improvements.

4 Experimental Evaluation

4.1 Experimental Setup

Extraction method: For our extraction method, we reimplemented the method described in (Paşca et al., 2006) and further added a validation layer on top of it based on Wikipedia (we boosted the scores of a fact if there exists a Wikipedia page for either of the fact's arguments, which mentions the other argument.) This state-of-the-art method forms a *strong* baseline in our experiments.

Corpus and farms: We ran our extractor over a large Web crawl consisting of 500 million English

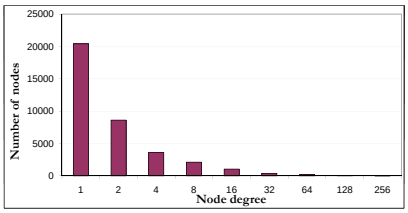

Figure 2: Degree distribution for MOVIES.

webpages crawled by the Yahoo! search engine. We removed paragraphs containing fewer than 50 tokens and then removed all duplicate sentences. The resulting corpus consists of over 5 million sentences. We defined a farm, MOVIES, with relations, *acted-in*, *director-of*, *is-movie*, *is-actor*, and *is-director*.

Evaluation methodology: Using our extraction method over the Web corpus, we generate over 100,000 facts for the above relations. However, to keep our evaluation manageable, we draw a random sample from these facts. Specifically, we first generate a ranked list using the extraction scores output by our extractor. We will refer to this method as ***Org*** (original). We then generate a fact graph over which we will run our methods from Section 3.3 (each of which will re-rank the facts). Figure 2 shows the degree, i.e., number of edges, distribution of the fact graph generated for MOVIES. We ran ***Avg***, ***Dst***, ***Nde***, ***R-Avg***, and ***R-Wgt*** on this fact graph and using the scores we re-rank the facts for each of the relations. In Section 4.2, we will discuss our results for the *acted-in* and *director-of* relations.

Fact Verification: To verify whether a fact is valid or not, we recruit human annotators using the paid service Mechanical Turk. For each fact, two annotations were requested (keeping the total cost under $100). The annotators were instructed to mark incorrect facts as well as disallow any values that were not "well-behaved." For instance, *acted-in*⟨*Godfather, Pacino*⟩ is correct, but *acted-in*⟨*The, Al Pacino*⟩ is incorrect. We manually adjudicated 32% of the facts where the judges disagreed.

Evaluation metrics: Using the annotated facts, we construct a goldset S of facts and compute the precision of a list L as: $\frac{|L \cap S|}{|S|}$. To compare the effectiveness of the ranked lists, we use average precision, a standard measure in information retrieval for evaluating ranking algorithms, defined as: $A_p(L) = \frac{\sum_{i=1}^{|L|} P(i) \cdot isrel(i)}{\sum_{i=1}^{|L|} isrel(i)}$, where $P(i)$ is the precision of L at rank i, and *isrel(i)* is 1 if the fact at rank i is in S, and 0 otherwise. We also study the precision values at varying ranks in the list. For robustness, we report the results using 10-fold cross validation.

Method	Average precision		
	30%	50%	100%
Org	0.51	0.39	0.38
Pln	0.44	0.35	0.32
Avg	0.55	0.44	0.42
Dst	0.54	0.44	0.41
Nde	0.53	0.40	0.41
R-Avg	0.58	0.46	**0.45**
R-Wgt	**0.60**	**0.56**	0.44

Table 2: Average precision for *acted-in* for varying proportion of fact graph of MOVIES.

Method	Average precision		
	30%	50%	100%
Org	0.64	0.69	0.66
Pln	0.69	0.67	0.59
Avg	0.69	0.70	0.64
Dst	0.67	0.69	0.64
Nde	0.69	0.69	0.64
R-Avg	0.70	0.70	0.64
R-Wgt	**0.71**	**0.71**	**0.69**

Table 3: Average precision for *director-of* for varying proportion of fact graph of MOVIES.

4.2 Experimental Results

Effectiveness of graph-based ranking: Our first experiment studies the overall quality of the ranked lists generated by each method. Table 2 compares the average precision for *acted-in*, with the maximum scores highlighted for each column. We list results for varying proportions of the original fact graph (30%, 50%, and 100%). Due to our small goldset sizes, these results are not statistically significant over ***Org***, however we consistently observed a positive trend similar to those reported in Table 2 over a variety of evaluation sets generated by randomly building 10-folds of all the facts.

Overall, the ***Avg*** method offers a competitive alternative to the original ranked list generated by the extractor ***Org***: not only are the average precision values for ***Avg*** higher than ***Org***, but as we will see later, the rankings generated by our graph-based methods exhibits some positive unique characteristics. These experiments also

R	*Org*	*Pln*	*Avg*	*Dst*	*Nde*	*R-Avg*	*R-Wgt*
5	0.44	0.40	0.52	0.48	0.40	0.52	**0.56**
10	0.36	0.36	**0.42**	0.38	0.36	0.36	0.36
15	0.287	0.24	**0.30**	0.28	0.26	**0.30**	**0.30**
20	0.26	0.26	0.26	0.26	0.26	**0.27**	**0.27**
21	0.27	0.27	0.27	0.27	0.27	0.27	0.27

Table 4: Precision at varying ranks for the *acted-in* relation (R stands for Ranks).

R	*Org*	*Pln*	*Avg*	*Dst*	*Nde*	*R-Avg*	*R-Wgt*
5	0.58	0.68	**0.70**	0.68	0.64	0.66	**0.70**
10	0.60	0.57	0.59	0.58	0.59	0.6	**0.69**
15	0.57	0.53	**0.58**	0.56	0.56	0.56	**0.60**
20	0.57	0.57	0.58	0.58	0.58	0.58	**0.60**
25	**0.60**	0.54	0.56	0.57	0.56	0.57	0.57
30	0.57	0.57	0.57	0.57	0.57	0.58	**0.59**
33	0.56	0.56	0.56	0.56	0.56	0.56	0.56

Table 5: Precision at varying ranks for the *director-of* relation (R stands for Ranks).

confirm our initial observations: using traditional PageRank (***Pln***) is not desirable for the task of re-ranking facts (see Section 3.3). Our modifications to the PageRank algorithm (e.g., ***Avg***, ***Dst***, ***Nde***) consistently outperform the traditional PageRank algorithm (***Pln***). The results also underscore the benefit of combining the original extractor ranks with those generated by our graph-based ranking algorithms with ***R-Wgt*** consistently leading to highest or close to the highest average precision scores.

In Table 3, we show the average precision values for *director-of*. In this case, the summary statistic, average precision, does not show many differences between the methods. To take a finer look into the quality of these rankings, we investigated the precision scores at varying ranks across the methods. Table 4 and Table 5 show the precision at varying ranks for *acted-in* and *director-of* respectively. The maximum precision values for each rank are highlighted.

For *acted-in* again we see that ***Avg***, ***R-Avg***, ***R-Wgt*** outperform ***Org*** and ***Pln*** at *all* ranks, and ***Dst*** outperforms ***Org*** at two ranks. While the method ***Nde*** outperforms ***Org*** for a few cases, we expected it to perform better. Error analysis revealed that the sparsity of our fact graph was the problem. In our MOVIES fact graph, we observed very few nodes that are linked to *all* possible relation types, and the scores used by ***Nde*** rely on being able to identify nodes that link to numerous relation types. This problem can be alleviated by reducing the sparsity of the fact graphs (e.g., by allowing edges between nodes that are "similar enough"), which we plan to explore as future work. For *director-of*, Table 5 now shows that for small ranks (less than 15), a small (but consistent in our 10-folds) improvement is observed when comparing our random walk algorithms over ***Org***.

#Relation	*Avg*	*Dst*	*Nde*
2	0.35	0.34	0.33
3	0.35	0.35	0.34
4	0.37	0.36	0.35
5	0.38	0.38	0.37
6	0.42	0.41	0.41

Table 6: Average precision for *acted-in* for varying number of relations in the MOVIES fact farm.

While our proposed algorithms show a consistent improvement for *acted-in*, the case of *director-of* needs further discussion. For both average precision and precision vs. rank values, ***Avg***, ***R-Avg***, and ***R-Wgt*** are similar or slightly better than ***Org***. We observed that the graph-based algorithms tend to bring together "clusters" of noisy facts that may be spread out in the original ranked list of facts. To illustrate this point, we show the ten lowest scoring facts for the *director-of* relation. Table 7 shows these ten facts for ***Org*** as well as ***Avg***. These examples highlight the ability of our graph-based algorithms to demote noisy facts.

Effect of number of relations: To understand the effect of the number of relations in a farm (and hence connectivity in a fact graph), we verified the re-ranking quality of our proposed methods on various subsets of the MOVIES fact farm. We generated five different subsets, one with 2 relations, another with 3 relations, and three more with four, five, and six relations (note that although we have 5 relations in the farm, *is-movie* can be used in combination with both *acted-in* and *director-of*, thus yielding six relations to ablate.) Table 6 shows the results for *acted-in*. Overall, performance improves as we introduce more relations (i.e., more connectivity). Once again, we observe that the performance deteriorates for sparse graphs: using very few relations results in degenerating the average precision of the original ranked list. The issue of identifying the "right" characteristics of the fact graph (e.g., number of relations, degree distribution, etc.) remains future work.

Org	*Avg*
⟨david mamet, bob rafelson⟩	⟨ drama, nicholas ray⟩
⟨cinderella, wayne sleep⟩	⟨ drama, mitch teplitsky official⟩
⟨mozartdie zauberflte, julie taymor⟩	⟨ hollywood, marta bautis⟩
⟨matthew gross, julie taymor⟩	⟨ hollywood, marek stacharski⟩
⟨steel magnolias, theater project⟩	⟨ drama, kirk shannon-butts⟩
⟨rosie o'donnell, john badham⟩	⟨ drama, john pietrowski⟩
⟨my brotherkeeper, john badham⟩	⟨ drama, john madden starring⟩
⟨goldie hawn, john badham⟩	⟨ drama, jan svankmajer⟩
⟨miramaxbad santa, terry zwigoff⟩	⟨ drama, frankie sooknanan⟩
⟨premonition, alan rudolph⟩	⟨ drama, dalia hager⟩

Table 7: Sample facts for *director-of* at the bottom of the ranked list generated by (a) ***Org*** and (b) ***Avg***.

Evaluation conclusion: We demonstrated the effectiveness of our graph-based algorithms for re-ranking facts. In general, ***Avg*** outperforms ***Org*** and ***Pln***, and we can further improve the performance by using a combination-based ranking algorithm such as ***R-Wgt***. We also studied the impact of the size of the fact graphs on the quality of the ranked lists and showed that increasing the density of the fact farms improves the ranking using our methods.

5 Related Work

Information extraction from text has received significant attention in the recent years (Cohen and McCallum, 2003). Earlier approaches relied on hand-crafted extraction rules such as (Hearst, 1992), but recent efforts have developed supervised and semi-supervised extraction techniques (Riloff and Jones, 1999; Agichtein and Gravano, 2000; Matuszek et al., 2005; Pantel and Pennacchiotti, 2006; Paşca et al., 2006; Yan et al., 2009) as well as unsupervised techniques (Davidov and Rappoport, 2008; Mintz et al., 2009). Most common methods today use semi-supervised pattern-based learning approaches that follow (Hearst, 1992), as discussed in Section 2. Recent work has also explored extraction-related issues such as, *scalability* (Paşca et al., 2006; Ravichandran and Hovy, 2002; Pantel et al., 2004; Etzioni et al., 2004), *learning extraction schemas* (Cafarella et al., 2007a; Banko et al., 2007), and *organizing extracted facts* (Cafarella et al., 2007b). There is also a lot of work on deriving extraction scores for facts (Agichtein and Gravano, 2000; Downey et al., 2005; Etzioni et al., 2004; Pantel and Pennacchiotti, 2006).

These extraction methods are complementary to our general task of fact re-ranking. Since our proposd re-ranking algorithms are agnostic to the methods of generating the initial facts and since they do not rely on having available corpus statistics, we can use any of the available extractors in combination with any of the scoring methods. In this paper, we used Paşca et al.'s (2006) state-of-the-art extractor to learn a large set of ranked facts.

Graph-based ranking algorithms have been explored for a variety of text-centric tasks. Random walk models have been built for document summarization (Erkan and Radev, 2004), keyword extraction (Hassan et al., 2007), and collaborative filtering (Liu and Yang, 2008). Closest to our work is that of Talukdar et al. (2008) who proposed random walk algorithms for learning instances of semantic classes from unstructured and structured text. The focus of our work is on random walk models over fact graphs in order to re-rank collections of facts.

6 Conclusion

In this paper, we show how information available in a farm of facts can be exploited for re-ranking facts. As a key contribution of the paper, we modeled fact ranking as a graph ranking problem. We proposed random walk models that determine the validity of a fact based on (a) the number of facts that "vote" for it, (b) the validity of the voting facts, and (c) the extractor's confidence in these voting facts. Our experimental results demonstrated the effectiveness of our algorithms, thus establishing a stepping stone towards exploring graph-based frameworks for fact validation. While this paper forms the basis of employing random walk models for fact re-ranking, it also suggests several interesting directions for future work. We use and build upon PageRank, however, several alternative algorithms from the link analysis literature could be adapted for ranking facts. Similarly, we employ a single (simple) graph-based representation that treats all edges the same and exploring richer graphs that distinguish between edges supporting different arguments of a fact remains future work.

References

[Agichtein and Gravano2000] Agichtein, Eugene and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *DL-00*.

[Auer et al.2008] Auer, S., C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. 2008. Dbpedia: A nucleus for a web of open data. In *ISWC+ASWC 2007*.

[Banko and Etzioni2008] Banko, Michele and Oren Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *ACL-08*.

[Banko et al.2007] Banko, Michele, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of IJCAI-07*.

[Cafarella et al.2007a] Cafarella, Michael, Dan Suciu, and Oren Etzioni. 2007a. Navigating extracted data with schema discovery. In *Proceedings of WWW-07*.

[Cafarella et al.2007b] Cafarella, Michael J., Christopher Re, Dan Suciu, Oren Etzioni, and Michele Banko. 2007b. Structured querying of web text: A technical challenge. In *Proceedings of CIDR-07*.

[Cohen and McCallum2003] Cohen, William and Andrew McCallum. 2003. Information extraction from the World Wide Web (tutorial). In *KDD*.

[Davidov and Rappoport2008] Davidov, Dmitry and Ari Rappoport. 2008. Unsupervised discovery of generic relationships using pattern clusters and its evaluation by automatically generated sat analogy questions. In *ACL-08*.

[Downey et al.2005] Downey, Doug, Oren Etzioni, and Stephen Soderland. 2005. A probabilistic model of redundancy in information extraction. In *Proceedings of IJCAI-05*.

[Erkan and Radev2004] Erkan, Güneş and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *JAIR*, 22:457–479.

[Etzioni et al.2004] Etzioni, Oren, Michael J. Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Web-scale information extraction in KnowItAll. In *Proceedings of WWW-04*.

[Etzioni et al.2005] Etzioni, Oren, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: an experimental study. *Artif. Intell.*, 165:91–134.

[Hassan et al.2007] Hassan, Samer, Rada Mihalcea, and Carmen Banea. 2007. Random-walk term weighting for improved text classification. *ICSC*.

[Hearst1992] Hearst, Marti A. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of COLING-92*.

[Kleinberg1999] Kleinberg, Jon Michael. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632.

[Lenat1995] Lenat, Douglas B. 1995. Cyc: a large-scale investment in knowledge infrastructure. *Commun. ACM*, 38(11).

[Liu and Yang2008] Liu, Nathan and Qiang Yang. 2008. Eigenrank: a ranking-oriented approach to collaborative filtering. In *SIGIR 2008*.

[Matuszek et al.2005] Matuszek, Cynthia, Michael Witbrock, Robert C. Kahlert, John Cabral, Dave Schneider, Purvesh Shah, and Doug Lenat. 2005. Searching for common sense: Populating cyc from the web. In *AAAI-05*.

[Mintz et al.2009] Mintz, Mike, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL-09*.

[Paşca et al.2006] Paşca, Marius, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. 2006. Organizing and searching the world wide web of facts - step one: The one-million fact extraction challenge. In *Proceedings of AAAI-06*.

[Page et al.1999] Page, Lawrence, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank citation ranking: Bringing order to the Web. Technical Report 1999/66, Stanford University, Computer Science Department.

[Pantel and Pennacchiotti2006] Pantel, Patrick and Marco Pennacchiotti. 2006. Espresso: leveraging generic patterns for automatically harvesting semantic relations. In *ACL/COLING-06*.

[Pantel et al.2004] Pantel, Patrick, Deepak Ravichandran, and Eduard Hovy. 2004. Towards terascale knowledge acquisition. In *COLING-04*.

[Pantel et al.2009] Pantel, Patrick, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *EMNLP-09*.

[Ravichandran and Hovy2002] Ravichandran, Deepak and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of ACL-08*, pages 41–47. Association for Computational Linguistics.

[Riloff and Jones1999] Riloff, Ellen and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of AAAI-99*.

[Talukdar et al.2008] Talukdar, Partha Pratim, Joseph Reisinger, Marius Pasca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. 2008. Weakly-supervised acquisition of labeled class instances using graph random walks. In *Proceedings of EMNLP-08*.

[Yan et al.2009] Yan, Yulan, Yutaka Matsuo, Zhenglu Yang, and Mitsuru Ishizuka. 2009. Unsupervised relation extraction by mining wikipedia texts with support from web corpus. In *ACL-09*.