Adaptive Focused Crawling of Linked Data

Ran Yu(✉), Ujwal Gadiraju, Besnik Fetahu,
and Stefan Dietze

L3S Research Center, Leibniz Universität Hannover, Hannover, Germany
{yu,gadiraju,fetahu,dietze}@L3S.de

Abstract. Given the evolution of publicly available Linked Data, crawling and preservation have become increasingly important challenges. Due to the scale of available data on the Web, efficient focused crawling approaches which are able to capture the relevant semantic neighborhood of seed entities are required. Here, determining relevant entities for a given set of seed entities is a crucial problem. While the weight of seeds within a seed list vary significantly with respect to the crawl intent, we argue that an adaptive crawler is required, which considers such characteristics when configuring the crawling and relevance detection approach. To address this problem, we introduce a crawling configuration, which considers seed list-specific features as part of its crawling and ranking algorithm. We evaluate it through extensive experiments in comparison to a number of baseline methods and crawling parameters. We demonstrate that, configurations which consider seed list features outperform the baselines and present further insights gained from our experiments.

Keywords: Focused crawling · Linked data · Relevance assessment

1 Introduction

With the emergence and continuous evolution of Linked Data, crawling and preservation of entities has become an important challenge. Usually entities as part of Linked Datasets are interlinked with other entities into a semantic neighborhood, with links having specific semantics. However, entities and their semantic neighborhood evolve, what leads to a number of practical implications.

For instance, we consider the task of document annotation, namely *entity linking* or *named entity disambiguation*. In this task, specific *surface forms* are linked to entities appearing in the Web data. Often entity linking refers to restricted knowledge graphs such as DBpedia [1], YAGO [17]. A linked entity can be referred to from different surface forms (e.g. `Barack Obama` can be referred as `'Obama', 'US President'` etc.). Considering the continuous evolution of Linked Data, capturing the temporal state and evolution of entities and their yet to be defined neighbourhood through dedicated crawling and preservation methods has become an increasingly important challenge.

Focused crawling, first introduced in [5] is a well studied field and has seen a wide application for web documents, specifically, especially in dataset creation

J. Wang et al. (Eds.): WISE 2015, Part I, LNCS 9418, pp. 554–569, 2015.
DOI: 10.1007/978-3-319-26190-4_37

and used by major search engines [3,4,6]. In the context of Linked Data, crawling is similar to that on the web and usually follow the links, i.e. object properties, of seed entities to find potential *candidate entities*.

Crawling for *candidate entities* in the Linked Data context is usually achieved through a breadth-first search (BFS) approach, as implemented by LDSpider [11], and can be tailored with respect to a prioritisation of crawling order and the considered distance, i.e. the maximum number of hops in the data graph.

Given the scale and dynamics of available Linked Data on the Web, more efficient *focused crawling* approaches are required which crawl and rank candidate entities for their relevance for a given *crawl intent*. The *crawl intent* usually is specified in the form of *seed entities*. Referring to the document annotation scenario, a typical seed list might be the set of entities extracted from a specific set of documents. The ranking of candidate entities can exploit two type of measures, *connectivity* or *similarity* between the candidate and seed entities. Such measures, exploit *lexical* as well as *structural*, respectively *graph-based* similarity metrics, which usually measure a notion of relatedness or similarity. As such, relevance assessment of entities is a critical challenge, of significant importance also with respect to other entity-centric tasks such as entity retrieval, semantic search or entity recommendation.

In our work, we show that the performance of entity ranking measures depends heavily on the nature of the seed list at hand, i.e., the coherence or crawl intent (see Sect. 6). We investigate these observations by comparing different crawl configurations which differ with respect to their maximum hop size and the chosen relevance detection metric. As one contribution, we analyse and compute the *crawl intent* of seed entities and reflect it in our premium crawling configuration. Here, we exploit an intuition documented in the work by Pound et al. [16], where the importance of individual entities of a composite query or seed list differs across seeds, with more specific entities usually reflecting the search intent to a higher degree. Exploiting this observation, we specifically boost entities closer to the crawl intent through a dedicated *attrition factor* as part of our candidate entity ranking.

To this end, our approach consists of the following steps: (i) seed list analysis, (ii) candidate crawling, (iii) seed list-specific candidate entity ranking, which are embedded into a crawling cycle. Our experiments show better performance and efficiency of our adaptive approach for a wide range of seed lists, compared with baseline ranking methods. In addition, insights gained from the above experiments are deemed applicable also in other scenarios, such as entity retrieval or entity recommendation, where candidate entity rankings need to be computed for a given query.

The paper is structured as follows. Section 2 introduces related works, followed by a problem definition and overview of the approach. Section 4 describes the seed list analysis, which provides a foundation for the adaptive crawling described later in the same Section. The experimental setup and results are described in Sects. 5 and 6 respectively. We discuss and conclude our work in Sect. 7.

2 Related Work

Focused Web Crawling. The purpose of web crawling is done for various reasons. Two of the widely spread use cases are the construction of datasets and that of vertical search engines like Google, Yahoo! etc. We focus our literature review mainly on the first case. Crawling data of a specific topic of information need, is also known as focused crawling. The term focused crawling was first introduced by de Bra et al. [5]. The focus has been widely shifted towards topic focused crawling.

Diligenti et al. [6] proposed a focused crawling approach that uses context graphs. A context graph in their case represents the link hierarchy between HTML pages and the co-occurring pages that are known to have as a target a relevant page. The main advantage of such an approach against traditional breadth-first-search crawls is that the model is optimized for the global relevance of pages rather than the immediate gains that are achieved from directly connected pages given a seed list. Chakrabarti et al. [4] proposed a topic focused crawling approach by assessing the relevance of a HTML page to a topic and further identifying sources (or web domains) that contain relevant documents closely connected to the seed pages. Later on, Chakrabarti et al. [3] proposed an improved approach with two main supervised modules. The modules are trained on the topic taxonomy Dmoz and the corresponding documents that are assigned to the topics. The first module serves as a source of generating new crawl targets that are passed as a priority list to the second module, which later on uses the DOM features from every proposed target crawl page to assess its final relevance for a given topic. McCallum et al. [13] presented a task oriented crawler for building domain-specific search engines. The approach crawls pages that are relevant to a specific topic and further extract information from the documents, i.e. title, date etc., which are later on used as query fields. Tang et al. [18] use external knowledge bases, specifically that of the medical domain, to improve both the relevance and quality of focused crawling.

Crawling of Structured Web Data and Linked Data. Meusel et al. [14] proposed a novel focused crawling approach for structured data. Their approach crawls microdata embedded in HTML pages. Further, they proposed a two-step approach. In the first step they use an online classifier that is trained on features extracted from the URLs of the pages, and in the second step assess the relevance of a group of pages (usually from the same source). In a closely related work, Isele et al. introduced an open-source crawling framework for Web data [11]. LDSpider traverses the Web of Linked Data by following RDF links between entities. LDSpider is less effective when a focused crawl is required, as depicted in later sections of this paper. In our previous work, we proposed a system that iteratively crawls and captures the evolution of Linked Data sets based on flexible crawl definitions [7].

Entity Relevance. Graph-based and lexical features of entities are used to assess the relevance of an entity in a range of tasks, such as *entity linking* or *entity*

retrieval. Graph-based relevance has been used in entity linking [15]. Lexical features represent common features used in previous focused web crawling works [3,4,13]. In [8], the authors introduce an entity retrieval approach, which exploits specifically lexical features to cluster and retrieve entities from the Web of data.

The aforementioned works differ on several aspects w.r.t our work. The scope of the crawl techniques relies on HTML web pages, where the nature of the data and hence the features that can be extracted is different. For instance, DOM features are used in [3] to perform the focused crawling. Other works like [13,18] proposed specific task oriented crawls, that of domain-specific search engines and medical domain crawls based on respective knowledge bases. In contrast, our work focuses solely on focused entity crawling from the Web of Linked Data. While previous work on LD crawling, such as LDSpider, is focused in the sense that they use a set of seed entities as starting point, they do not compute the relevance of retrieved entities or provide a respective ranking. Another distinguishing feature is that, as opposed to our work, features of the seed list are not considered for configuring the crawl method. Instead, mostly static crawling approaches are used, which apply the same strategy and method on each seed list.

3 Problem Definition and Approach Overview

In this section, we first describe the motivation and use case of this work, then provide the formal problem definition as well as a brief overview of our approach.

3.1 Motivation and Problem Definition

Our work aims at providing a method for focused crawling of Linked Data, namely to crawl entities of relevance to a given seed list consisting of entities which describe the crawl intent. While this is a task of relevance for efficiently preserving a particular partition of the Linked Data cloud in general, we would like to emphasise its use in document annotation or data enrichment scenarios. Here, traditionally named entity disambiguation or entity linking techniques are deployed to annotate documents with entities from a specific knowledge graph. While such entity annotations enable structured queries to retrieve documents of relevance to specific entities, more sophisticated semantic search needs to exploit additional knowledge from the used reference graphs, such as DBpedia or related graphs. While the overall Linked Data graph might contain highly relevant information about the semantics of a specific entity, identifying the paths and entities of relevance, or the *semantic neighbourhood* of a given entity, is a non-trivial task.

Specifically, the challenge is to identify and crawl the most relevant entities for a given set of seed entities. We define *focused crawling* of Linked Data as follows. Given a specific seed list of entities $S = \{e_1, \ldots, e_n\}$, the aim is to crawl and rank relevant candidate entities $C = \{e'_1, \ldots, e'_n\}$, available from the Web. Though seeds could commonly be represented through terms which require a disambiguation step, for simplification purposes we assume that seed entities

are represented by entity URIs, for instance, referring to instances within the DBpedia graph.

To illustrate the need for scalable and focused crawling approaches, note that our earlier experiments have shown that a 2-hop crawl of a given seed list results in 38,295 entities on average. Hence, we aim for a focused approach, where relevance of entities is computed as part of the crawling process and seeds for the following hop are determined based on their relevance to the seed list.

3.2 Approach Overview

In order to tackle the challenges of focused crawling, we adopt the following steps: (i) seed lists analysis, (ii) breadth-first search crawl (BFS) for candidates C, (iii) seed list-specific *candidate entity ranking*. The last two steps are embedded into a crawling cycle by using the relevant entities selected from step (iii) as next hop crawling queue for step (ii). Explained in more detail in the following section, we consider an *attrition factor* of each seed entity $e_i \in S$ to reflect the *crawl intent* during the focused crawling process. This is based on the assumption that entities within a seed list have varying importance for the *crawl intent*, and hence, their individual impact on the ideal result set differs.

The overall focused crawling framework is summarized in Algorithm 1.

Algorithm 1. Focused Crawling of Linked Data

Require: Seed list
Ensure: Related entities

```
1: function FocusedCrawling(seedlist, depth)
2:     coherence ← CoherenceComputation(seedlist)
3:     attritionFactors ← AttritionFactorComputation(seedlist)
4:     queue ← seedlist
5:
6:     for hop = 0 → depth do
7:         candidates ← BreadthFirstCrawl(queue)
8:         relatedEntities ←RelevanceAssessment(candidates, attritionFactors)
9:         queue ← relatedEntities
10:    end for
11:    return relatedEntities
12: end function
```

A focused crawling *configuration* represents an implementation of Algorithm 1, dependent on specific attributes of the candidate crawling process and seed list analysis, such as *depth* for candidate crawling and *attrition factor* of seed entities in a seed list. The corresponding configurations are described in detail in Sect. 5.

4 Adaptive Focused Crawling

In this section we describe the seed list analysis and the adaptive crawling approach.

4.1 Seed List Analysis

Here we define and describe in detail the seed list analysis, specifically on how we compute the *seed list coherence* and *attrition factor* in Algorithm 1.

Seed List Coherence. We assume that crawl configurations require tailoring to the *seed list coherence*. The underlying assumption is that very specific and targeted seed lists will require different crawling and relevance computation methods than very broad and unspecific seed lists. An example of the former is, for instance, a seed list containing entities labeled with '`Barack Obama`', '`President of the United States`', '`George W. Bush`', while '`John Lennon`', '`Africa`', '`Mahatma Gandhi`' would constitute a very unspecific seed list.

We introduce a score to measure *seed list coherence*, Γ, computed as the reciprocal average shortest path between any seed entity pairs (see Eq. 1). The shortest path between any two seed entities is denoted by $\varphi(e_i, e_j)$ on a given knowledge graph.

$$\Gamma = \frac{n(n-1)}{2 \sum\limits_{\substack{i,j \\ i<j}}^{n} \varphi(e_i, e_j)} \Leftarrow \begin{array}{c} \\ e_1 \\ e_2 \\ \vdots \\ e_n \end{array} \begin{array}{c} \begin{array}{cccc} e_1 & e_2 & \dots & e_n \end{array} \\ \begin{pmatrix} 0 & \varphi(e_1, e_2) & \dots & \varphi(e_1, e_n) \\ & 0 & \dots & \varphi(e_2, e_n) \\ & & & \vdots \\ & & & 0 \end{pmatrix} \end{array} \tag{1}$$

The main intuition behind the computation of *seed list coherence* in Eq. 1 is that entities with shorter paths in a given knowledge base tend to be more closely related. Hence, the average shortest path between seed entities determines the *coherence* of a seed list. In our experiments, we identify two crawl scopes based on Γ: (i) *low coherence*, and (ii) *high coherence*. We consider seed lists that follow $0 \leq \Gamma \leq \gamma$ to exhibit low coherence, while $\Gamma > \gamma$ indicates high coherence. The value of threshold $\gamma = 0.5$ was identified during our experiments, the detailed setup is described in Sect. 5.1.

Seed Entity Attrition Factor. As shown in our experimental evaluation, specific entities within a seed list strongly reflect the crawl intent. For example, consider the seed list {`Pulp Fiction, Film, Entertainment`}, the most specific entity '`Pulp Fiction`', reflects the most specific crawl intent, whereas the entities '`Film`' and '`Entertainment`' provide contextual information, namely that '`Pulp Fiction`' is a movie. Motivated by this, we assume that the relevance of specific candidate entities is dependent on the seed entity they are

related to. For example, candidate entities similar to entity `'Pulp Fiction'` will be ranked higher than entities that are similar to other seed entities.

In order to improve the candidate entity ranking we define the *attrition factor* $\lambda(e_i)$ for each seed entity in S. The *attrition factor* $\lambda(e_i)$ is measured as the fraction of the Katz centrality score [12] of e_i over the sum of all scores from all other entities in S terms of the proportion of Katz centrality. The Katz score is computed w.r.t a reference dataset (in our case DBpedia), and entities with lower centrality score will have higher *attrition factor*, hence reflecting more strongly the *crawl intent*. The attrition factor $\lambda(e_i)$ is computed as in Eq. 2.

$$\lambda(e_i) = \log \frac{C_{Katz}(e_i)}{\sum_{j=1}^{n} C_{Katz}(e_j)} \tag{2}$$

where, e_i is the seed entity, n is the number of seed entities in S, and $C_{Katz}(e_i)$ is the Katz centrality of e_i, and $e_i \in S$.

$$C_{Katz}(e_i) = \sum_{k=1}^{5} \sum_{j=1}^{n} \alpha^k (A^k)_{ji} \tag{3}$$

where, A is the adjacency matrix of entity e_i, namely the connections of e_i in the reference dataset, whereas k reflects the number of hops in the reference graph that are used to compute the adjacency matrix. The second sum in Eq. 3 measures the connectivity degree between e_i and entities in the knowledge graph that are reachable within the hop k. Finally, α is an exponential penalization factor, which prefers higher connectivity degree on the earlier hops, and it takes values in the range of $\alpha \in [0, 1]$.

4.2 Candidate Crawling and Ranking

In this section, we describe the breadth-first search (BFS) crawl for candidates entities and the seed list-specific candidate entity ranking.

Crawl for Candidate Entities. The BFS crawl starts with a queue Q_i, $i = 0...depth$. This queue is populated by the seed list S in the first hop, and is in later hops replaced by the relevant candidate entities C (see Algorithm 1). With the increase of the number of hops, the number of candidate entities increases, hence we perceive a drop in crawling efficiency. The crawling *depth* defines the maximum number of hops, where we aim for a depth that provides high efficiency by means of sufficient candidates as well as short runtime.

Candidate Entity Ranking. Candidate entities are ranked according to their relevance to the seed entities in S. The relevance of a candidate entity to a seed list can be determined by distinguishing the following aspects.

– Pairwise relevance of a candidate entity with respect to each seed entity, i.e., $rel\{c_i, e_j\}$ where $c_i \in C$ and $e_j \in S$.

– Overall relevance of a candidate entity to the entire seed list, i.e., $rel\{c_i, S\}$ where $c_i \in C$.

Relevance of candidate entities c_i with respect to seed entities e_j is computed by using a distance measure, abridged with our weighting scheme consisting of the attrition factor λ, as introduced in Sect. 4.1.

This method involves using text-based similarity metrics such as *Jaccard Similarity*, to assess the similarity between the candidate entities and seed entities. Using labels and descriptions of entities, we construct term-reference vectors for each entity in the seed and candidate set, respectively S and C. This step is semi-automated as a user can configure the relevant datatype properties used to construct the term vector w_i for each seed entity e_i. The term-reference vector in our case represents a set of unigrams (single terms) extracted from the textual literal datatypes from the corresponding entities.

The following formulas show the adapted pairwise Jaccard Similarity ($PairwiseJaccardSim^+$), where the traditional Jaccard similarity has been adopted to consider the crawl intent, as well as the overall candidate relevance computed through the *Entity-based Jaccard Similarity ($EJSim^+$)*.

$$PairwiseEJSim^+(e_i, c_j) = \frac{1}{\lambda(e_i)} \cdot \frac{|w_i \cap w'_j|}{|w_i \cup w'_j|} \tag{4}$$

$$EJSim^+(c_j) = \frac{\sum_{i=1}^{n} PairwiseEJSim^+(e_i, c_j)}{n} \tag{5}$$

where, $\lambda(e_i)$ is the attrition factor of the seed entity e_i, and w_i, w_j are the entity reference-vectors corresponding to e_i and c_j.

5 Experimental Setup

In this section, we introduce the datasets we consider for the crawling process and the approaches on generating the seed lists. Next we assess the performance of the different crawling configurations, constrained for the varying parameters: (i) *seed list coherence*, (ii) *hop-size* and (iii) *crawl-candidate ranking approaches*. Consequently, we draw conclusions about the optimal *crawl parameters*. Finally, we compare our approach against established baseline methods.

5.1 Seed Lists

Our experiments are following our original motivation, that is the document annotation scenario, where seed entities are directly linked to and extracted from documents in a corpus. As we aim to evaluate crawling performance for a variety of seed lists, we generate a range of seed lists from real-world documents. In order to eliminate noise, potentially introduced by extracting entities from an arbitrary corpus, we directly utilise Wikipedia, where DBpedia entities can be derived directly from the hyperlinks of a source page to other Wikipedia pages.

Since popular Wikipedia pages tend to be linked more consistently, we particularly consider Wikipedia pages with high page views during the first week of 2012 accumulating 1.6 billion records. We have specifically chosen a period in the past to ensure well-populated and popular pages, rather than new and insufficiently annotated pages. To generate seed lists of varying *coherence*, we follow two different strategies for creating seed lists.

- (1) top-k entities in single documents
- (2) k entities extracted from sets of related documents

The intuition is that (1) will produce more coherent and (2) more diverse seed lists. In (1) we start with the page of an original entity u, we extract the entities $u_1, u_2, ..., u_t$ appearing in the page and rank these according to their frequency. Assuming that the entities with the highest tf score are closer related to the original entity, we use the top k entities as seeds.

For (2) we start again with a set one most viewed wikipedia pages, i.e. entities. For each entity u, we randomly obtain a category from the set of categories associated with u and retrieve all entities $u_1, u_2, ..., u_k$ in the same category. We then extract the entities $u'_1, u'_2, ..., u'_k$ appearing in the pages of $u_1, u_2, ..., u_k$ as candidates. Then we randomly select k entities from the accumulated candidate set as low coherence seed list.

Applying this method on 3 wikipedia pages and 3 categories, we obtained 6 seed lists of size $k = 5$ with varying coherence Γ. The seed lists and the groundtruth are available at http://l3s.de/~fetahu/crawler_wise2015/.

5.2 Crowdsourced Ground Truth

We leverage crowdsourcing as a means to establish the ground truth in the form of a ranking of the most relevant entities within a specified n-hop neighborhood for a given seed list S. We adopt the following steps in order to establish the ground truth for evaluation.

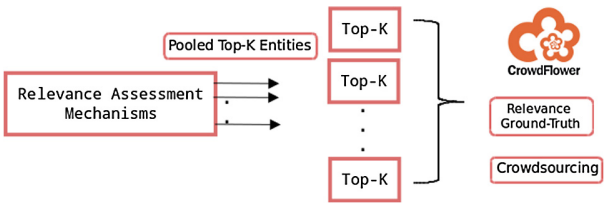


Fig. 1. Generation of ground truth for evaluation of relevance assessment measures.

First, we pool the top-k entities[1], as obtained by using the different relevance assessment measures. We model the activity of assessing the relevance

[1] In this case we pooled the $Top - 500$ entities resulting from all the different configurations for each seed list.

of an entity to the seed list, as an atomic microtask that can be deployed on a crowdsourcing platform such as CrowdFlower. In order to assist the crowd workers in assessing the relevance of an entity to a given seed list, we represent the seed list with a corresponding *topic* that describes the collective meaning projected by the seed list. In order to determine an apt *topic* that can represent a seed list and thereby the crawl intent, we follow a gamified approach such as the popular ESP game proposed by Von Ahn and Dabbish [19]. Each seed list is presented to 5 different experts, who are then asked to independently tag the seed lists with representative topics. When all 5 experts reach a concensus with respect to a particular description, that tag is chosen to be the representative *topic* of the seed list. For instance, for the seed list generated from `How I Met Your Mother (TV Series)` Wikipedia page, experts assigned the topic `How I Met Your Mother (TV Series) - Characters and Actors`.

Next, we present each entity gathered from the neighborhood of the seed list alongside the corresponding *topic* to crowd workers and request judgments of relevance according to a 5-point Likert-scale (ranging from *Not Relevant* to *Highly Relevant*). By aggregating the judgments from the workers into relevance scores, and ranking the pooled top-k entities based on these relevance scores, we thereby generate our ground truth of relevant entities corresponding to the seed list S. Finally, we use the ground truth thus established to evaluate each of the relevance assessment methods across the top-k entities.

In order to ensure that the ground truth is reliable, we take several precautions as recommended by our previous works [9,10] to curtail malicious activity and to avoid misinterpretations in the relevance scoring.

5.3 Crawl Configurations and Baselines

We distinguish *crawling configurations* by (a) the used candidate ranking approach, (b) the hop size (depth) for the crawling process, and (c) the consideration of the seed list coherence (or lack thereof). The parameters are listed below. Here the candidate ranking considers two cases: (i) candidate relevance scoring taking into account the attrition factor λ which is denote as $EJSim^{+}$, and (ii) candidate relevance scoring without the attrition factor, i.e. the baselines below.

The maximum hop size *depth* considered during the crawling process is one of the most significant impact factors for runtime and NDCG score. We run experiments with *depth* in $\{2, 3\}$, the corresponding methods are denoted with a subscript indicating the depth, where, for instance, $EJSim_2$ indicates a $depth = 2$.

We consider the following baseline and state-of-the-art approaches for the focused crawling task.

EJSim. Entity Jaccard Similarity, *EJSim* computed as *EJSim+* but without considering the *attrition factor*.

NJSim. We also consider the graph-based relatedness baseline, the *Neighborhood Jaccard Similarity* which is computed as follows.

$$PairwiseNJSim(e_i, c_j) = \frac{1}{\lambda(e_i)} \cdot \sum_{k=1}^{\tau} \alpha^k \frac{|N_k(e_i) \cap N_k(c_j)|}{|N_k(e_i) \cup N_k(c_j)|} \tag{6}$$

$$NJSim(c_j) = \frac{\sum_{i=1}^{n} PairwiseNJSim(e_i, c_j)}{n} \tag{7}$$

where, $N_k(e_i)$, $N_k(c_i)$ are the sets of neighboring vertices at the k^{th} step of seed entity (e_i) and candidate entity (c_i) respectively, $\alpha \in [0, 1]$ is a real number to ensure that closer neighbors have lower weight, and τ is the maximum length of the paths considered.

PageRank. Is a widely adopted approach for ranking crawling results (documents or entities). The computation of PageRank is formalized in Eq. 8.

$$PageRank(c_i) = \frac{1-d}{N} + d \sum_{c_j \in M(c_i)} \frac{PR(c_j)}{L(c_j)} \tag{8}$$

where $M(c_i)$ is the set of entities linked to candidate c_i in the entity graph, $L(c_j)$ is the number of outbound links to the entity c_j, N is the total number of entities, and d is the damping factor [2].

NB. We consider [3] as state-of-the-art approach for focused web crawling. However, while this approach originally implements focused crawling of Web documents as opposed to Linked Data, we introduce some adaptations. The DOM tree features cannot be considered in case of Linked Data, hence, we adopt lexical features for the NB classification. The state-of-the-art approach uses a predefined topic taxonomy with example URLs as training set. In our senario, the topic, reflected by the crawl intent in our work, is dependent on the seed list without predefined limitations. However, our groundtruth can be used as training set for a classifier. The classification of candidate entity as relevant is performed by the function formalized in Eq. 9.

$$NB(c_i) = P(y = 1|c_i) = \frac{P(y)P(c_i|y)}{P(c_i)} \tag{9}$$

where $P(y = 1|c_i)$ is the probability of candidate c_i belonging to class $y = 1$, which in our case translates as a *'relevant'* entity for a given seed list.

5.4 Evaluation Metrics

To evaluate the crawling performance of the different configurations, we consider the *Normalized Discounted Cumulative Gain (NDCG)* of the ranked set of candidate entities with respect to an established ground truth. The NDCG score is computed as follows:

$$nDCG@k = \frac{DCG@k}{iDCG@k} \quad DCG@k = rel_1 + \sum_{i=2}^{k} \frac{rel_i}{log_2 i}$$

where, $DCG@k$ represents the discounted cumulative gain at rank 'k', and $iDCG@k$ is the ideal $DCG@k$ computed with respect to the *ground truth* (described in the following section).

Another important factor to evaluate is the runtime performance of the different configurations. Due to the likelihood of large set of candidate entities existing for a particular seed list, runtime is a crucial factor. The runtime is simply measured in terms of the amount of time taken to complete a full-cycle of the crawling process.

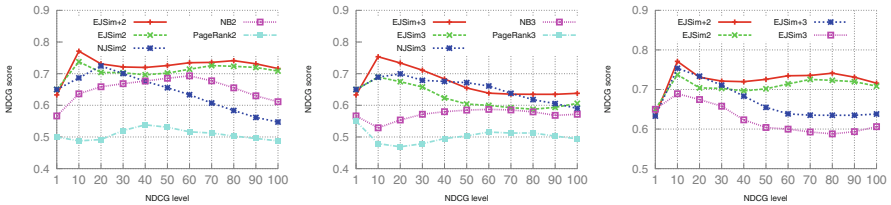
6 Evaluation

6.1 Performance of Focused Crawling Configurations

In this section we report and discuss the performance of the different focused crawling configurations. On average, for the different seed lists we crawl approximately 491,425 triples. Hence, an important aspect is how well the different crawl configurations rank the candidate entities for their relevance w.r.t the seed lists. In Table 1 we report the performance of the different crawling configurations. The values are reported for the average NDCG scores at rank 100.

Table 1. Average NDCG@100 for different focused crawling configurations across seed lists with varying coherence.

Coherence	$EJSim_2^+$	$EJSim_2$	$NJSim_2$	NB_2	$PageRank_2$
$\Gamma < 0.5$	**0.7446**	0.7364	0.5752	0.6955	0.5346
$\Gamma \geq 0.5$	**0.6876**	0.6802	0.5193	0.5273	0.4408
	$EJSim_3^+$	$EJSim_3$	$NJSim_3$	NB_3	$PageRank_3$
$\Gamma < 0.5$	**0.6928**	0.6382	0.6608	0.6612	0.5722
$\Gamma \geq 0.5$	**0.5831**	0.5740	0.5197	0.4821	0.4166



(a) (b) (c)

Fig. 2. (a) Performance of different configurations with depth 2 measured as average NDCG across all seed lists. (b) Performance of different configurations with depth 3 measured as average NDCG across all seed lists. (c) Performance of attrition factor (on depth 2 and 3) measured as average NDCG across all seed lists.

Performance. Table 1 presents the average NDCG@100 score of different configurations described in Sect. 5.3 with seed lists of varying coherence. Figure 2a

and b present the NDCG score at different levels for the different configurations. Figure 2c presents a detail comparison between the performance of $EJSim$ and $EJSim^+$. It shows that the NDCG scores increased after introducing the attrition factor to EJSim. The average improvement across different NDCG levels is 1.6 % on *depth* 2 and 4.3 % on *depth* 3, suggesting a positive effect of the *attrition factor* for the cases of our seed lists. On the other hand, the *coherence* of the seed list appears to have no significant impact on the suitability of particular configuration.

From the results we can also observe that, the $EJSim^+$ outperforms baseline methods, specifically $NJSim$ and $PageRank$. The NDCG@100 of $EJSim^+$ is 16.9 % and 4.8 % higher than $NJSim$ for *depth* 2 and 3 respectively. The NDCG scores at different levels (Fig. 2a and b) also support this insight. $PageRank$ shows the weakest performance since it does not take the crawl intent into consideration during candidate ranking. Furthermore, $EJSim^+$ shows 7.3 % and 9.8 % improvement in average for *depth* 2 and 3 respectively compared to NB in our experiment. Based on the dataset, one of the reasons that NB and $EJsim^+$ have different performance while both use lexical features is that the classes within the training set are not independent from each other. The overlap between the feature set of different classes causes negative effects on the NB classification. The reason of the overlap is that the crawl intent is based on the seed lists instead of the topic from a predefined topic taxonomy. This also reveals that our method is more robust and flexible for a range of crawl intents.

Another useful and conclusive insight is the fact that *depth* 3 has not led to performance gains with respect to *crawl depth* of 2. In fact we can observe the opposite from Fig. 2c, where for $EJSim^+$, increasing the crawl depth beyond 2 seems to lead to weaker NDCG scores on average.

Efficiency. Since the time-intensive step is the candidate crawling, there is no considerable difference between the crawling configurations with different ranking methods. However, the average runtime across seed lists and ranking methods increased from 548 s for depth 2 to 1936.6 s for depth 3 in average while there is no significant improvement of the NDCG score. Given the significantly increased runtime when crawling beyond hop 2, a crawl depth of 2 seems to provide optimal efficiency, and it is not advisable to crawl to a higher distance.

6.2 Discussion and Limitations

Next to the aforementioned observations, we also carried out an analysis on more varied characteristics of seed lists, leading to some interesting insights. After the seed list size reaches a certain threshold (20 in our case), the graph-based methods seem to outperform lexical ones in most cases. This might be explainable with the fact that the increasing heterogeneity of larger seed lists renders lexical methods less applicable.

Meanwhile, PageRank performance seems to improve with increasing seed list size and decreasing coherence. This can be explained since Page Rank is not query-biased in any way and the performance gap to the superior, seed list-based configurations is decreasing the more generic the seed lists become.

We evaluate the result of experiments based on a ground truth as described in Sect. 5.2. During the preliminary analysis of the dataset, we observe that the overall NDCG score for results of low coherence seed lists seems significantly higher than those of high coherence seed lists. This is due to the fact that high coherence seed lists have a more specific crawl intent, leading to narrow and often small result sets, and hence also a limited ground truth, while the low coherence lists have a much broader crawl intent as well as relevant entity set. This is reflected in our ground truth: the average number of entities labeled as related (score ≥ 3 and beyond) is 208 for low coherence seed list, and 145 for high coherence seed lists. Meanwhile, the narrow search intent also causes more disagreement among crowdsourcing workers for generating the ground truth, which makes the results for high coherence seed lists less consensual.

Another difficulty faced when evaluating the crawling task is the highly heterogeneous and varied nature of the possible result sets, originating from a highly heterogeneous Linked Data graph. While certain seed lists and crawl intents are well reflected in the Web of Linked Data, others are only sparsely represented, leading to highly varying quality of the retrieved result sets. Here, to some extent the availability of matching data, or the lack thereof, can have a significant impact on the measured NDCG scores. This problem is elevated due to the fact that crawl results rely on links, where the uneven distribution of links across entity types and partitions of the Linked Data cloud leads to result sets of varying size and quality.

7 Conclusions and Future Work

In our work, we have presented an adaptive focused crawling method which takes into account characteristic features of seed lists to improve the crawling and relevance ranking method. Our crawl configurations consider experimentally derived heuristics and thresholds for the crawl *depth* and relevance assessment method. Our experimental results support the underlying assumption that the crawl intent can be reflected dynamically through a dedicated *attrition factor* to improve the ranking of retrieved entities. Our results demonstrate that the attrition factor has a positive impact on the performance, showing significant performance improvement across a range of seed lists, compared to all considered baselines.

Additional insights from our experiments might be of practical value for focused Linked Data crawling approaches in general. For instance, our results suggest that crawls beyond depth 2, i.e. a distance of 2 hops in the Linked Data graph, seem to not improve the crawl quality but instead, introduce noise and significantly increase the crawl runtime, hence decreasing the efficiency.

One central obstacle when evaluating methods for the focused crawling task is the lack of a sufficient ground truth and the heterogeneity of the Linked Data graph, which naturally leads to highly diverse ground truth result sets for different seed lists. This might have a negative influence on the resulting ground truth and might dilute the performance evaluation to a certain extent.

However, given the scale of our experiments, the shown results provide conclusive indicators about the performance of the investigated configurations.

While our work aimed at achieving a general performance gain, specifically measured through NDCG, our ongoing and future work foresees in particular the on-the-fly optimisation towards particular performance goals with regards to precision, recall, or runtime. While the actual crawl aims may vary strongly between different crawls - in some cases, high precision may be mandatory, in others a broad crawl, i.e. high recall will be of higher priority - through further experiments we aim at identifying more specific heuristics which can lead to a more tailored adaptation.

In this context, we are also investigating the specific characteristics of graph-based relevance metrics and lexical metrics. The high divergence of result sets produced by these two different approaches documents that, naturally, lexical methods seem better suited to retrieve *similar* entities, while graph-based ones seems better suited to retrieve otherwise *connected* entities. This suggests that each approach might be specifically suited to certain kind of seed lists, or crawl intent, what might require an even more adaptive approach, involving the selection or configuration of suitable relevance ranking methods at runtime.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007)
2. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Comput. Netw. ISDN Syst. **30**(1), 107–117 (1998)
3. Chakrabarti, S., Punera, K., Subramanyam, M.: Accelerated focused crawling through online relevance feedback. In: Proceedings of the 11th International Conference on World Wide Web, WWW, pp. 148–159. ACM, New York (2002)
4. Chakrabarti, S., Van den Berg, M., Dom, B.: Focused crawling: a new approach to topic-specific web resource discovery. Comput. Netw. **31**(11), 1623–1640 (1999)
5. De Bra, P., Houben, G.-J., Kornatzky, Y., Post, R.: Information retrieval in distributed hypertexts. In: RIAO, pp. 481–493 (1994)
6. Diligenti, M., Coetzee, F., Lawrence, S., Giles, C.L., Gori, M., et al.: Focused crawling using context graphs. In: VLDB, pp. 527–534 (2000)
7. Fetahu, B., Gadiraju, U., Dietze, S.: Crawl me maybe: iterative linked dataset preservation. In: Proceedings of the 13th International Semantic Web Conference (ISWC) Posters & Demonstrations Track, pp. 433–436 (2014)
8. Fetahu, B., Gadiraju, U., Dietze, S.: Improving entity retrieval on structured data. In: Proceedings of the 14th International Semantic Web Conference. Springer (2015)
9. Gadiraju, U., Demartini, G., Kawase, R., Dietze, S.: Human beyond the machine: challenges and opportunities of microtask crowdsourcing. IEEE Intell. Syst. **30**(4), 81–85 (2015)
10. Gadiraju, U., Kawase, R., Dietze, S., Demartini, G.: Understanding malicious behaviour in crowdsourcing platforms: the case of online surveys. In: Proceedings of CHI 2015 (2015)

11. Isele, R., Umbrich, J., Bizer, C., Harth, A.: Ldspider: an open-source crawling framework for the web of linked data. In 9th International Semantic Web Conference, ISWC. Citeseer (2010)
12. Katz, L.: A new status index derived from sociometric analysis. Psychometrika **18**(1), 39–43 (1953)
13. McCallumzy, A., Nigamy, K., Renniey, J., Seymorey, K.: Building domain-specific search engines with machine learning techniques (1999)
14. Meusel, R., Mika, P., Blanco, R.: Focused crawling for structured data. In: Proceedings of the 23rd ACM International Conference on Information and Knowledge Management, CIKM, pp. 1039–1048 (2014)
15. Pereira Nunes, B., Dietze, S., Casanova, M.A., Kawase, R., Fetahu, B., Nejdl, W.: Combining a co-occurrence-based and a semantic measure for entity linking. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 548–562. Springer, Heidelberg (2013)
16. Pound, J., Mika, P., Zaragoza, H.: Ad-hoc object retrieval in the web of data. In: Rappa, M., Jones, P., Freire, J., Chakrabarti, S. (eds.) WWW, pp. 771–780. ACM (2010)
17. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of the 16th International Conference on World Wide Web, pp. 697–706. ACM (2007)
18. Tang, T.T., Hawking, D., Craswell, N., Griffiths, K.: Focused crawling for both topical relevance and quality of medical information. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, pp. 147–154. ACM (2005)
19. Von Ahn, L., Dabbish, L.: Labeling images with a computer game. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 319–326. ACM (2004)