Long-Tail Entity Extraction With Low-Cost Supervision

Paper 101

ABSTRACT

Named Entity Recognition and Typing (NER/NET) is a challenging task, especially with long-tail entities such as the ones found in scientific publications. These entities – e.g. "datasets for evaluating recommender systems" – are rare, and often relevant only in specific knowledge domains. Yet, their automatic recognition and typing is essential to unlock and mine the knowledge contained in digital libraries.

State-of-the-art NER/NET approaches employ supervised machine learning models, trained on expensive type-labeled data produced by human annotators. A common workaround is the generation of labeled training data from knowledge bases; this approach is not suitable for long-tail entity types that are, by definition, scarcely represented in KBs.

This paper presents a novel approach for training NER and NET classifiers for long-tail entity types. The approach relies on minimal human input, namely a small seed set of instances for the targeted entity type. In an automatic fashion, the approach generates labeled data and trains a NER/NET classifier. The NER/NET is then applied to the targeted document collection, and new entities are identified. The approach expands and refines the list of entities in an automatic and iterative fashion.

We propose and discuss different strategies for training data extraction and named entity filtering. The approach is showcased in the context of scientific publication annotation, with long-tail entities types such as *Datasets* and *Methods*. Through quantitative and qualitative evaluation, we show that our approach can provide good quality results (up to 0.91 precision and 0.41 recall) with a seed set of 100 entities, and achieve comparable performance with a seed set as small as 5 entities and two iterations.

Keywords

Information Extraction. Document Metadata, Named Entity Recognition, Long-Tail Entity Types, Training Data Generation

1. INTRODUCTION

The growth of domain-specific knowledge available in digital form demands more effective methods for document query, access, and exploration. Scientific publications are a compelling example. Online digital libraries such as the ACM Digital Library (DL), IEEE Xplore, Google Scholar, etc, contain hundreds of thousands documents; yet, the available retrieval functionalities are often limited to keyword/-faceted search on *shallow* meta-data such as the paper's title, authors, and keywords. A query like *retrieve the publications that used social media datasets for food recipes recommendation* is bound to return unsatisfactory results[1].

Manually or automatically produced annotations of relevant *named entities* obtained through an analysis of a document's content are an effective way to achieve better retrieval performance. Alternatively, they could also enable complex entity-centric queries, e.g., to retrieve the names of social media datasets as in the aforementioned example. In this work, we focus on the use case example of data science publications, where relevant named entity types are dataset names (e.g. `Imagenet`), the names of data data processing methods (e.g. `LSTM`); or the names of the employed software (e.g `Pytorch`).

A considerable body of work investigated *Named Entity* Recognition and Typing (NER/NET) from textual content [2, 16, 43, 44]. NER/NET methods usually capitalise on the existence of textual patterns associated to named entities in the targeted collection to recognize them in new document. These patterns can be learned through handcrafted rules [36, 28, 6, 32], dictionary-based solutions [4, 3, 18], or machine learning techniques [17]. To properly perform, these techniques either require comprehensive domain knowledge, or rely on a large amount of human-labeled training data, both temporally and monetarily expensive solutions. A cheaper alternative is to generate labeled training data by obtaining existing instances of the targeted entity type from Knowledge Bases (KB) [2, 43, 44] - a technique which can only be applied if the desired entity type is well-covered in the knowledge base.

While achieving impressive performance with high-recall named entities (e.g. locations and age) [16], state-of-the art NER/NETs show their limits with domain-specific entities. Consider the following sentence: *"We evaluated the performance of SimFusion+ on the WEBKB dataset"* [42].

[1] `https://scholar.google.de/scholar?q=publications+using++social+media+datasets+for+food+recipes+recommendation`

Despite *WebKB*[2] being a popular dataset in the Web research community, state-of-the-art general-purpose NERs (e.g. Textrazor[3]) mistype it as an organization instead of the domain-specific entity type (Dataset). The entity *SimFusion+* of type Software is missed completely.

Problem Statement. Literature focusing on NER/NET from scientific publications [35, 32, 26, 38, 40] show that training of *domain-specific* NER/NETs is still an open challenge for two main reasons: 1) the *long-tail* nature of such entity types, both in existing knowledge bases *and* in the targeted document collections [30]; and 2) the high cost associated with the creation of hand-crafted rules or human-labeled training datasets for supervised machine learning techniques. Yet, automatic NER/NET is essential to unlock and mine the knowledge contained in digital libraries as most smaller domains lack the resources for manual annotation work. In this paper, we focus on the following research question:

RQ: How to train high-performance *long-tail* Entity Extraction and Typing with minimal human supervision?

Original Contribution. We contribute an iterative five-step low-cost approach for training NER/NET classifiers for long-tail entity types. Starting from a *seed set* of instances of the targeted entity type, (1) we obtain text snippets from a large corpus to be used for training in a first *training data extraction* step; (2) both the set of seed instances and training snippets are *semantically expanded* to include potential yet unknown instances; (3) the training data is *annotated* with the expanded instance list to (4) train a NER that is then applied on the document corpus to extract new named entities of the given types. In a final step, (5) the extracted entities are heuristically filtered to exclude likely misclassified instances, yielding the final result set. This automatic approach relies on minimal human input (the seed set of entities), and can operate in an iterative fashion by being repeated using the result set as seed for the next iteration.

We introduce and discuss different strategies for training data extraction, semantic expansion, and result entity filtering. We showcase and evaluate our approach using a corpus of 11,589 publications from ten conference series. We focus on two types of long-tail entities (*Datasets* and *Methods*) and performed extensive evaluation of different configurations, giving also insights into how our approach can be customised for different requirements.

To the best of our knowledge, this work is the first addressing the problem of long-tail entity recognition with minimal human input. We show that our approach is able to provide good performance across different entity types and small amount of training information: with a seed set of 100 entities, the approach can achieve precision up to 0.91 and recall up to 0.41. When applied in an iterative fashion, the approach can achieve comparable performance with an initial seed set of only 5 entities. Sentence expansion and filtering strategies can provide a spectrum of performance profiles, suitable for different retrieval applications such as search (high precision) and exploration (high recall).

[2]`http://www.cs.cmu.edu/~WebKB/`

[3]`https://www.textrazor.com/`

Outline. This paper is organized as follows. In Section 2 we first briefly cover related work. Section 3 presents our approach, and describes alternative data expansion and entity filtering strategies. The experimental setup and results are presented in Section 4. Section 5 concludes.

2. RELATED WORK

2.1 Deep Analysis Of Technical Texts

A considerable amount of literature published in recent years addressed the *deep* analysis of text in scientific domains such as Bio-Informatics [37], Computers Science [13], or Digital Libraries [1], etc. Common approaches for *deep* analysis of publications rely on techniques such as word-frequency analysis [34], co-citation analysis [5, 13], co-word analysis [14], and probabilistic methods like latent Dirichlet allocation [9]. In contrast to current research [10, 34] which limits the analysis of a publication's content to its title, abstract, references, and authors, we extract entity instances from the full text of scientific publications to avoid missing the entities of interest. In addition, our method does not rely on existing knowledge bases [32, 26] and it is not based on selecting the most frequent keywords (e.g. nouns, verbs set and proper nouns) [34].

More recent research [35] used both corpus-level statistics and local syntactic patterns of scientific publications to identify entities of interest. Our method uses only a small set of seed names (i.e 5-100) and the word representations to train a NER in iterative steps (i.e. 2-3) for the extraction of long tail entity instances.

2.2 Entity Instances Extraction

Named Entity Recognition (NER) has received significant attention in the past decade, and has been applied to identify the entity types of interest (e.g. Person, Location, Cell, Brand, etc) in general as way as more specific domains: from scientific text [30, 35], web [27, 44, 16], Social media [31, 21, 2, 8], patents [15], etc. NERs rely on different approaches such as dictionary-based, rule-based, machine-learning [35] or hybrid (combination of rule based and machine learning) [41, 26] techniques. Despite its high accuracy, a major drawback of dictionary-based approaches is that they require an exhaustive dictionary of domain terms. These dictionary are expensive to create for less relevant domain-specific entity types. The same holds for rule-based techniques, which rely on formal languages to express rules and require comprehensive domain knowledge and time to create.

The lack of the availability of large collection of labeled training data and the high cost of data annotation for a given domain is one of the main issues of machine learning approaches. In recent years, many attempts have been made to reduce annotation costs. Active learning techniques have been proposed, asking users to annotate a small part of a text for rule-based [24] and machine learning methods [33, 39, 7]. Transfer learning techniques [29] use the knowledge gained from one domain and apply it to a different but related named entity type.

The Entity Set Expansion is a technique aiming at finding similar entities to a given small set of seed entities [2, 16, 43, 44]. Our approach is inspired by this, but relies on different expansion strategies and has different goals. We focus on the extraction of *domain-specific long-tail entities* from scientific publication using a small set of seed names.

Differently from previous work, we do not require the presence of a large training corpus [29] for transfer learning, and we do not rely on the concepts already available in knowledge bases [2, 43]. Our approach also differs from works on high-recall entity extractors (e.g. Geonames, regular expression extractors) for detecting entity types such as location and age [16].

3. APPROACH

This section introduces our approach for domain-specific long-tail entity recognition in text documents. We focus on the following core hypothesis: there is a recurring pattern in the mentions of domain-specific named entities contained in a document set, and their appearance occur in similar contexts. If this hypothesis holds, by training a classifier on the texts containing the entities, we are able to extract the instances of the entity type of interest.

To pursue an answer to the research question of the paper, we designed the five steps approach depicted in Figure 1, and detailed below:

1. An initial set of seed terms is used to identify a set of sentences containing such terms, that are used as *training data* (Section 3.1).
2. *Expansion* strategies can be used to expand the set of initial seed terms, and the *training data* sentences (Section 3.2).
3. The *Training Data Annotation* step annotates the training data using the (possibly expanded) seed terms set (Section 3.3).
4. A new Named Entity Recognizer (NER) is *trained* using the annotated training data, and the newly trained NER is applied on the corpus to detect a candidate set of entities (Section 3.4).
5. The *Filtering* step refines the set candidate entities set, to improve the quality of outputted *Verified Terms* set Section 3.5).

The process can be iterated, by repeating the first step using the newly detected verified terms as seeds to generate new training data.

The following concepts are required to realise our approach. Some concepts are only relevant for the evaluation in Section 4.3, and could be omitted in setups where evaluation is not necessary.

Known Entity Terms $T_{all} := T_{seed} \cup T_{test}$: This represents a manually created set of instances of the entity type for which a NER classifier is to be trained. In this work, we split this set into a set of seed terms T_{seed} used for training, and test terms T_{test} used for evaluation purposes. In a real-life scenario not requiring a formal evaluation, of course only the seed terms would be necessary. T_{seed} may be small. In this work we consider seed sets $5 \leq |T_{seed}| \leq 100$. Creating T_{seed} is the only manual input required for NER training in our approach.

Document Corpus $D_{all} := \{d_1, ..., d_{|D|}\}$: This is the complete document corpus available to our system. Parts of it can potentially be used for training, others for testing. Each document is considered to be a sequence of sentences.

All Sentences $S_{all} := \{s | s \in d \wedge d \in D_{all}\}$: This represents all sentences of the whole document corpus. Each sentence is considered to be a sequence of terms.

Test Sentences $S_{test} := \bigcup_{t \in T_{test}} \{s | s \in S_{all} \wedge t \in s\}$: These are all sentences containing any term from the test set, and they need to to be excluded from any training in order to ensure the validity of our later evaluations, resulting in the set of **Development Sentences** $S := S_{all} \setminus S_{test}$.

In the following, we already introduce the iterative version of our approach, representing the current iteration number as i whereas initially $i = 0$. Each iteration i uses its own term list T_i, which initially is $T_0 \subseteq T_{seed}$ (the size of the subset of T_{seed} depends on the desired usecase, as discussed in section 4.3).

3.1 Training Data Extraction

As a first step of our approach, a set of training data sentences S_i for the current iteration is created by extracting suitable sentences from S. At this stage, this is realized by selecting all sentences containing any of the seed terms. Therefore, S_i provides examples of the positive classification class as they are guaranteed to contain a desired entity instance. To better capture the usage context of the seed entity, we also extract surrounding sentences in the text: $S_i := \cup_{t \in T_i} \{s | s \in S \wedge (t \in s \vee t \in \mathit{successor}(s) \vee t \in \mathit{predecessor}(s))\}$.

3.2 Expansion

The small size of the seed term set T_{seed} has two obvious shortcomings: 1) the amount of training data sentences S_i is limited; and 2) there are only few examples of mentions of the entity instances of the given type. These two issues can greatly hinder the accuracy and recall of the trained NER. In addition, the *generalisation* capability of the NER for identifying new named entities can also be affected: an insufficient amount of positive examples can lead to a situation where entities of the sought type are labeled negatively; at the same time, the extraction of sentences in the training data that are related to seed terms will cause a shortage of negative examples for training purposes. To account for these issues, we designed two *expansion* strategies.

3.2.1 Term Expansion (TE)

Term Expansion is designed to increase the number of known instances of the desired entity type before training the NER. An expanded set of entities will provide more positive examples in the training data, thus ideally improving the precision of the NER. In technical documents, it is common for domain-specific named entities to be in close proximity, e.g. to enumerate alternative solutions, or list technical artifacts. The *Term Expansion* (TE) strategy is therefore designed to test and exploit this hypothesis.

We introduce the function interface $\mathit{expandTerms}(\mathit{terms}_s)$, with $\mathit{terms}_s \subseteq \mathit{terms}_i$. While many different implementations for this interface are possible, in this work we use *semantic relatedness*: terms which are semantically similar or related to terms in the seed list should be included in the expansion. For example, given the dataset seed terms `Clueweb` and `cim-10`, the expansion should add semantically related terms like `trec-2005` or `brassdl`. For this, we exploit the distributional hypothesis[11] stating that terms frequently occurring in similar context are semantically related [22], realised by the popular *word2vec* implementation of skip-n-gram word embeddings [25]. In essence, *word2vec* embeds each term of a large document corpus into low-dimensional vector space (100 dimensions in our case).

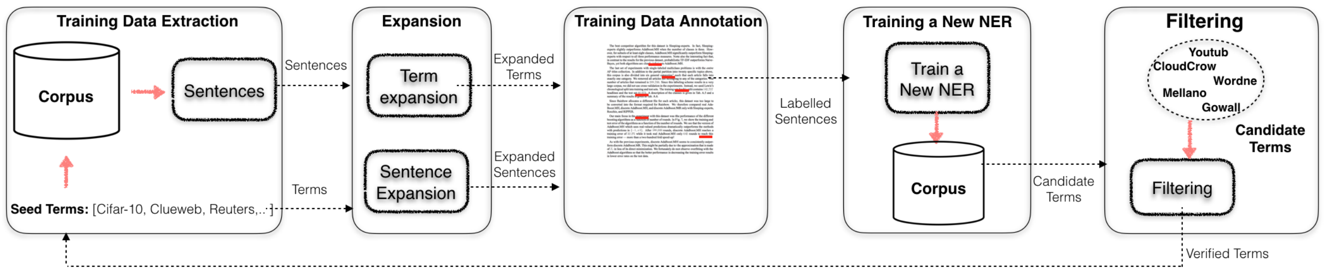

Figure 1: Overview of the domain-specific long-tail named entities recognition approach.

The cosine distance between two vectors has been shown to be a high-quality approximation of similarity semantics [22]. In our implementation, we trained the *word2vec* model on the whole development sentence collection S, as described in [25], learning all uni- and bi-gram word vectors of all terms in the corpus. Then, in its most basic version, we select all terms from all sentences, and cluster them with respect to their embedding vectors using K-means clustering. Silhouette analysis is used to find the optimal number k of clusters. Finally, clusters that contain at least one of the seed terms are considered to (only) contain entities the same type (e.g *Dataset*). Initial experiments have shown that this naive approach is slow, and that it can potentially introduce many false positives due to 1) the large number of considered terms, and 2) the assumption that terms in cluster are indeed similar. Therefore, as an necessary optimisation, we consider only terms which are likely to be a named entity instead of all terms in the corpus. For this, we use the NLTK entity detection to obtain a list of all entities E_{all} contained in S (the NLTK entity detection is based on grammatical context. It does not perform any typing, and due to it's simplicity, has high recall values.) This results in the following algorithm:

Algorithm 1 Term Expansion using Semantic Relatedness

```
function EXPANDTERMS($terms_s$)
    $T_{entity} := \{t | t \in s \wedge s \in S \wedge isEntity(t)\}$
                                        ▷ All entities in $S$
    $clusters := cluster(word2vec(T_{entity}))$
                                        ▷ Cluster the embeddings
    $clusters_{correct} := \{c | c \in clusters \wedge t \in terms_s \wedge t \in c\}$
                                        ▷ Select clusters containing any initial term
    return $\bigcup_{c \in clusters_{correct}}$
end function
```

3.2.2 *Sentence Expansion (SE)*

A second (optional) measure to increase the size and variety of the training set in a guided fashion is the *Sentence Expansion* (SE) strategy. It addresses the problem of the over-representation of positive examples resulting from selecting only sentences with instances of the desired type (see section 3.1). The goal is to include sentences which are unlikely to contain instances of the desired type, but are still very similar in semantics and vocabulary to serve as informative negative examples in order to boost the NER training accuracy.

For this, we rely on *doc2vec* document embeddings [20], a variant of *word2vec*, to learn vector representations of the sentences in the corpus. For each sentence in the development set, we use *doc2vec* to discover the most similar sentence which does not contain any known instance of the targeted type (i.e., expanded terms). While indeed such sentences sometimes do contain an unknown instance of the targeted entity type, which would now be misclassified in the training set). To minimise such possibility, in our experiments sentence expansion always include the term expansion strategy.

Algorithm 2 Optional Sentence Expansion

```
function EXPANDSENTENCES($S_{org}$)
    return $sentences \cup \{s | s' \in S_{org} \wedge s \in S \wedge mostSimilar_{doc2vec}(s, s')\}$
end function
```

3.3 Training Data Annotation

After obtaining an (expanded) set of instances T_i (the current term list) and training sentences S_i, we annotate each term (denoted as $A_{T_i} := annotate_{T_i}(S_i)$) in all training sentences of they are a positive instance of the targeted entity type, i.e.if the term $\in T_i$. Using A_{T_i}, any state-of-the-art supervised NER algorithm can be trained.

3.4 NER Training

For training a new NER_i, we used the Stanford NER tagger[4] to train a Conditional Random Field (CRF) model. As the focus of this paper is the process of training data generation, we do not consider additional algorithms. CRF has shown to be an effective technique on different NER tasks [19]; the goal of CRF is to learn the hidden structure of an input sequence. This is done by defining a set of feature functions (e.g. word features, current position of the word labels of the nearby word), assigning them weights and transforming them to a probability to detect the output label of a given entity. The features used in the training of the model are listed in Table 1.

After a NER for the current iteration N_i is trained, it is used to annotate the whole development corpus S, i.e. $A_{NER_i} := annotate_{NER_i}(S)$. All positively annotated terms are considered newly discovered instances of our desired type.

[4]https://github.com/dat/stanford-ner

Table 1: Stanford NER training parameters.

useWord=true	useLastRealWord=true
useNGrams=true	useNextRealWord=true
usePrev=true	lowercaseNGrams=true
useNext=true	featuresuseTypeSeqs=true
useLemmas=true	useTypeSeqs2=true
normalize=true	useTypeySequences=true
useOccurrencePatterns=true	wordShape=chris2useLC

3.5 Filtering

After applying the NER to the development corpus, we obtain a list of new candidate terms. As our process relied on several steps which might have introduced noise and thus also false positives (like the expansion steps, but also the NER itself), the goal of this last (and optional) step is to filter out candidate terms that are not likely of the targeted entity type using a set of external heuristics with different underlying assumptions:

3.5.1 Wordnet + Stopwords (WS) Filtering

In the domain-specific language of technical documents, it is common for named entities to be proper of such a domain (like `Simlex-999`), or to be expressed as acronyms (like `Clueweb`, `SVM`, `RCV`). In this strategy, named entities are assumed to be not relevant if they are part of the "common" English language, either as proper nouns (e.g. *software*, *database*, *figure*), or a Stopwords (e.g. *on*, *at*).

We operationalize this strategy by performing look-up operations in WordNet[5] and in common lists of stopwords.[6] As both sources focus on general English language, only domain-specific terms should be preserved.

3.5.2 Similar Terms (ST) Filtering

In order to distinguish between different entity types that pertain to a given domain (e.g. `SVM` is of type `Method`, while `Clueweb` is of type `Dataset`), this filtering strategy employs an approach similar to the one used in the *Term Expansion* (TE – Section 3.2.1) strategy. The idea is to cluster entities based on their embedding feature using K-means clustering, and keep all the entities that appear in the cluster that contains a seed term.

3.5.3 Pointwise Mutual Information (PMI) Filtering

This filtering strategy adopts a semantic similarity measure derived from the number of times two given keywords appear together in a *sentence* in our corpus. The heuristic behind this filter is vaguely inspired by Hearst Patterns [12], as we manually compile a list of context terms / patterns XT which likely indicate the presence of an instance of our desired class (e.g., "we evaluate on x" typically indicates a dataset). Unlike the other filters, it does increase the manual resource costs for training.

Given a set of candidate entities CT_i and the context term set CX, we measure the PMI between them using $\log \frac{\mathrm{N(ct,cx)}}{\mathrm{N(ct)N(cx)}}$ with $ct \in CT_i \wedge cx \in CX$, and $N(ct, cx)$ being the number of sentences in which both a candidate entity (ct) and a given keyword (t) occur (analogously, $N(ct)$ counts the number of occurrences of ct). Finally, candidate terms are filtered and excluded if their PMI value is below a given threshold value.

[5]`http://wordnet.princeton.edu/`

[6]`http://www.nltk.org/book/ch02.html`

3.5.4 Knowledge Base Look-up (KBL) Filtering

Our target are long-tail domain-specific entities, i.e. entities that are not part of existing knowledge bases. We could therefore consider named entities that could be linked to a knowledge base as incorrect, and therefore amenable to exclusion from the final named entity set. In the KBL approach we exclude the entities that have a reference in the DBpedia.

3.5.5 Ensemble (EN) Filtering

Different filtering strategies are likely to remove different named entities. To reduce the likelihood of misclassification, the *Ensemble* (EN) filtering strategy combines the judgment of multiple filtering strategies, to preserve candidate entities that are considered correct by one or more strategy. Intuitively, if each strategy makes different errors, then a combination of the filters' judgment can reduce the total error. We make use of the 2 out of 3 majority vote, were we select the entities that are passed through two out of three selected filtering strategies.

3.6 Summary

In the following, we summarize the previous subsections into a unified algorithm covering the whole iterative NER training workflow.

Algorithm 3 Iterative NER Training

```
function LONGTAILTRAIN($T_{seed}$, $S_{all}$)
  $T_0 := T_{seed}$
  for $i \in \mathbb{N}_0$ do
    $S_i := \cup_{t \in T_i}\{s | s \in S \wedge (t \in s \vee t \in successor(s) \vee t \in predecessor(s))\}$
    $T_i := expandTerms(T_i)$
    (optional) $S_i := expandSentences(S_i)$
    $A_{T_i} := annotate_{T_i}(S_i)$
    $NER_i := trainNER(A_{T_i})$
    $A_{NER_i} := annotate_{NER_i}(S)$
    $CT_i := isPositiveTermIn(A_{NER_i})$
    $FT_i := filter(CT_i)$
    $T_{i+1} := FT_i \cup T_{seed}$
    if convergence then
      return $A_{NER_i}$
    end if
  end for
end function
```

4. EVALUATION

This section reports on an empirical evaluation to assess the performance of the approach (and its variants) described in Section 3, and the ability to utilise it for long-tail named entity recognition. Section 4.1 describes the experimental set-up, followed by the results (Section 4.2), and their discussion (Section 4.3).

4.1 Experimental Setup

4.1.1 Corpus Analysis

The evaluation is performed in the domain of scientific publications with a focus on data science and processing. In our corpus D_{all}, we have 11,589 papers from ten conference series: The Joint conference on Digital Libraries (JCDL –

1,416 papers from 2001 to 2016); the International Conference on Theory and Practice of Digital Libraries (TPDL – 276 papers from 2011 to 2016); the International Conference on Research and Development in Information Retrieval (SIGIR – 412 papers from 1971 to 2016); the Text Retrieval Conference (TREC – 1,444 papers from 1999 to 2015); the European Conference on Research and Advanced Technology on Digital Libraries (ECDL – 820 papers from 1997 to 2010); the International Conference on Software Engineering (ICSE – 1834 papers from 1976 to 2016); the Extended Semantic Web Conference (ESWC – 626 papers from 2005 to 2016); the International Conference On Web and Social Media (ICWSM – 810 papers from 2007 to 2016); the International Conference on Very Large Databases (VLDB – 1884 papers from 1975 to 2007); and the International World Wide Web Conference (The Web Conference – 2067 papers from 2001 to 2016). Scientific publications, typically available in PDF, are processed using state-of-art extraction engines, e.g. GeneRation Of BIbliographic Data (GROBID) [23]. GROBID extracts a structured full-text representation as Text Encoding Initiative(TEI)-encoded documents, thus providing easy and reliable access paragraphs and sentences.

4.1.2 Long tail entity types selection

Scientific publications contain a large quantity of long-tail named entities. Focusing on the (broad) computer science domain, in the following we address the entity types *Dataset* (i.e. dataset presented or used in a publication), and *Methods* (i.e. algorithms – novel or pre-existing – used to create/enrich/analyse a data set). Both entities types are scarcely represented in existing knowledge bases,[7] thus making the acquisition of labeled training data unfeasible.

To evaluate the performance of our approach, we create a set of 150 seed entity instances T_{all} for each targeted entity type, collected public from public websites.[8] For each type, 50 of those are selected as test terms for that type T_{test}, while 100 are used as seed terms T_{seed}.

4.1.3 Evaluation Dataset

As discussed in Section 3, in the training process all test sentences S_{test} (i.e. sentences mentioning terms in T_{test}) from the document corpus D_{all} are removed/ To measure the performance of our approach, we manually created a type-annotated test set: for each test term, we select all sentences in which they are contained including any adjacent sentence, forming the set of annotated sentences $S_{annotated} := \cup_{t \in T_{test}} \{s | s \in S_{test} \wedge (t \in s \vee t \in successor(s) \vee t \in predecessor(s))\}$. An expert annotator labeled each term as an instance of the target type to create the test annotation set used for evaluation $A_{test} := annotate_{expert}(S_{annotated})$. Statistics on sentences used for training and testing are shown in Table 2. The evaluation protocol is described in Algorithm 4, where the *seed_size* values can be initialized with different values.

[7]In DBPedia, the type `dbo:database` features 989 instances, but mostly related to biology, economy, and history. The type `dbo:software` contain names of several algorithms, but the list is clearly incomplete.

[8]For instance: `https://github.com/caesar0301/awesome-public-datasets`. The full list of seed entity instances, as well as the list of sources are available on the companion Website `https://sites.google.com/site/www2018101/`.

Algorithm 4 Evaluation Protocol

```
function EVALUATE($seed\_size$)
    $T \subseteq_{seed\_size} T_{seed}$
    $NER_{final} := longtailTrain(T, S_{all})$
    $A_{final} := annotate_{NER_{final}(S_{annotated})}$
    $result := analyze(A_{final}, A_{test})$
end function
```

Table 2: Size Statistics fof various seed set sizes #S

Entity type	*#S*	*#Sentences*	*#Words*
Dataset Training	5	198	2081
	10	358	4737
	25	799	13080
	50	1456	29015
	100	2863	63517
Method Training	5	617	15682
	10	1192	30354
	25	3620	86563
	50	7910	190026
	100	18543	449515
Dataset Testing	50	3149	69272
Method Testing	50	1097	26426

4.2 Results

For a given entity type, we test the performance with differently sized seed sets (i.e. 5, 10, 25, 50, 100), different expansion strategies (*No Expansion*, *Term Expansion*, and *Sentence Expansion*) to create the training data for generating the NER model, and different filtering strategies to filter the resulting set of recognized entities. We report the performance of the basic WS, PMI, KBL, and EN strategies, plus a combination of the WS, ST, and KBL strategies, as listed in the caption of Table 3. We also perform an experiment to test the performance of the approach when applied iteratively. We analyse the performance of the model on the manually annotated test set presented in the previous section.

Tables 3 – 6 summarise the performance achieved by our approach under different experimental configurations, respectively for *Dataset* and *Method* entity types. The *No Expansion* and *Term Expansion* figures for the *Method* entity type are omitted for brevity's sake. The approach is able to achieve excellent precision [89% – 91%] with both entity types, and good recall (up to 41%) with the *Dataset* type. The lower recall performance obtained with the *Method* entity type can be explained with the greater diversity (in terms of n-grams and use of acronyms) of method names. The expansion strategies lead to an average +200% (SE – *Dataset*) and +300% (TE – *Dataset*) increase in the recall performance, thus demonstrating their effectiveness for generalisation purposes. On average, the filtering strategies decrease recall, but with precision improvements up to +20% (PM – *Method*). These are promising figures, considering the minimal human supervision involved in the training of the NERs. We can also show in our experiments the different trade-offs our approach is able to strike: different configurations of filtering and expansion strategies lead to different results with respect to precision and recall values, allowing for example a high-precision slightly-lower recall setup for a digital library, and a higher recall lower precision setup for a Web retrieval system.

Table 3: *Dataset* entity type: Precision/Recall/F-score on evaluation dataset. Legend: *NF* – No Filtering; *WS* – Wordnet + StopWords; *SS* – Similar Terms + WS; *KB* – Knowledge Base Look-up; *KS* – KB + SS; *PM* – Point-wise Mutual Information; *EN* – Ensemble.

Strategy	*#S*	*NF*	*WS*	*SS*	*KB*	*KS*	*PM*	*EN*
No Exp. (NE)	5	.83/**.05/.10**	.84/.04/.08	.86/.03/.07	.71/.01/.01	.75/.01/.01	**.90**/.04/.09	.86/.04/.08
	10	.84/**.07/.14**	.83/.06/.12	.85/.06/.11	.70/.01/.03	.78/.01/.02	**.90/.07**/.13	.85/.06/.11
	25	.84/**.08/.16**	.83/.07/.13	.86/.07/.13	.71/.02/.04	.78/.01/.03	**.91/.08**/.15	.85/.07/.13
	50	.85/**.12/.21**	.84/.10/.18	.87/.10/.18	.73/.03/.06	.80/.02/.05	**.92**/.11/.20	.86/.10/.18
	100	.85/**.15/.26**	.85/.13/.22	.87/.12/.22	.77/.04/.08	.82/.03/.07	**.91**/.13/.24	.86/.12/.22
Term Exp. (TE)	5	.76/**.14/.25**	.78/.13/.22	.79/.11/.20	.70/.05/.09	.74/.04/.09	**.83**/.13/.23	.80/.13/.22
	10	.72/**.24/.36**	.74/.21/.33	.76/.21/.33	.68/.10/.18	.70/.10/.18	**.78**/.22/.35	.76/.21/.33
	25	.72/**.29/.42**	.73/.28/.40	.75/.27/.40	.71/.18/.29	.73/.17/.28	**.77**/.27/.40	.75/.27/.40
	50	.70/**.36/.47**	.71/.33/.46	.73/.33/.45	.69/.22/.34	.71/.21/.33	**.75**/.33/.46	.73/.33/.45
	100	.69/**.41/.51**	.70/.39/.50	.71/.38/.50	.69/.29/.41	.71/.28/.40	**.74**/.38/.50	.72/.38/.50
Sentence Exp. (SE)	5	.83/**.07/.14**	.84/.06/0.12	.86/.05/.10	.75/.01/.03	.82/.01/.02	**.91/.07**/.13	.86/.06/.11
	10	.81/**.15/.26**	.81/.13/.22	.84/.12/.21	.66/.02/.05	.73/.02/.05	**.89**/.14/.25	.84/.12/.21
	25	.81/**.22/.35**	.80/.18/.29	.83/.17/.29	.70/.05/.09	.77/.04/.08	**.89**/.20/.33	.82/.18/.29
	50	.78/**.27/.40**	.78/.22/.35	.81/.21/.34	.69/.07/.12	.76/.06/.11	**.87**/.24/.38	.80/.22/.34
	100	.77/**.30/.43**	.77/.24/.37	.80/.23/.36	.70/.08/.15	.78/.07/.13	**.86**/.26/.40	.79/.24/.37

Table 4: *Method* entity type: Precision/Recall/F-score on evaluation dataset. Legend as in Table 3.

Strategy	*#S*	*NF*	*WS*	*SS*	*KB*	*KS*	*PM*	*EN*
Sentence Exp. (SE)	5	.76/**.04/.08**	.77/.03/.07	.77/.01/.01	.84/.02/.04	.84/.01/.01	**.86**/.01/.03	.84/.03/.05
	10	.80/**.11/.19**	.81/.10/.18	.81/.03/.07	.85/.06/.12	**.89**/.02/.04	.87/.04/.07	.86/.08/.14
	25	.77/**.14/.24**	.77/.12/.21	.79/.09/.16	.84/.08/.14	**.87**/.05/.09	.86/.05/.09	.85/.09/.17
	50	.74/**.15/.25**	.73/.13/.22	.74/.11/.20	.81/.08/.16	**.85**/.06/.12	.83/.05/.10	.83/.10/.18
	100	.68/**.15/.25**	.67/.14/.23	.65/.12/.20	.80/.09/.17	.84/.07/.13	**.85**/.05/.10	.83/.10/.19

4.2.1 Expansion Strategies

Expansion strategies were designed to increase the size and variety of training datasets, thus improving the precision and recall of the NERs. In this respect, both strategies achieve the expected results, although with different performance increase: compared to *No Expansion* (NE) strategy, both the TE (term expansion) and SE (sentence expansion with term expansion) strategies achieve a considerable performance boost ($\mu = +190\%$) in terms of recall, but at a cost of lower precision ($\mu = -8.7\%$). We account the better recall performance of TE to the contextual similarity (and proximity) of named entities of the same type in technical documents (e.g. `Gov2`, `Robust04`, `ClueWeb` and `Wt10g`). The precision decrease in TE can be accounted to treating some terms incorrectly as positive instance examples due to their presence in the same embedding clusters as the seed terms (see also section 3.2.1). The SE strategy shows lower recall performance ($\mu = +210\%$ over NE), but with less precision loss ($\mu = -5.2\%$ than NE). We account this positive behaviour to the presence of more good-quality negative examples in the training set, that helps maintaining the generalization capabilities of the NER, while refining the quality of its recognition.

4.2.2 Filtering Strategies

This section presents the result of different filtering techniques to reduce the number of false positives detected by the noisy NER. We can observe no significant improvement in precision with the WS filtering approach. Manual inspection of results reveal that most of the false positives are already domain-specific terms (e.g. `Pagerank`, `Overcite` for *Dataset*, and `NDCG` for *Method*) which are not included in Wordnet, but that are of the wrong type. SS slightly increases the precision by keeping only the entities that appear in the same cluster as the seed names; however, this comes at a cost, as the recall is also penalised by the exclusion of entities of interest that are in other clusters. KB excludes popular entities that are contained in the knowledge base (e.g. `Wordnet`, `Dailymed`), but also some rare entities that are mistyped.

Table 5: #*Dataset* entities (out of the 150 in T_{all}) recognized. Legend as in Table 3.

Strategy	*#S*	*NF*	*WS*	*SS*	*KB*	*KS*	*PM*	*EN*
NE	5	*41*	**35**	22	9	8	34	31
	10	*37*	**35**	25	11	9	34	26
	25	*57*	50	35	17	15	**51**	44
	50	*93*	**86**	78	33	30	81	76
	100	*122*	**114**	101	45	45	107	101
TE	5	*83*	**72**	45	21	21	69	62
	10	*101*	**87**	45	26	26	86	75
	25	*118*	105	97	40	40	**108**	94
	50	*130*	**115**	103	43	42	110	100
	100	*148*	**132**	97	53	52	125	115
SE	5	*61*	**54**	52	56	51	53	48
	10	*91*	**81**	73	30	30	78	72
	25	*111*	**102**	92	35	35	94	88
	50	*120*	**110**	100	41	40	101	95
	100	*135*	**125**	113	56	56	116	110

Table 6: #*Method* entities (out of the 150 T_{all}) recognized. Legend as in Table 3.

Strategy	*#S*	*NF*	*WS*	*SS*	*KB*	*KS*	*PM*	*EN*
SE	5	*44*	**44**	41	12	11	26	28
	10	*691*	**69**	64	16	14	37	40
	25	*91*	**91**	83	21	18	41	50
	50	*101*	**101**	93	26	24	46	56
	100	*108*	**108**	99	29	28	49	60

For instance, the *Dataset* entities `Ratebeer`[9] or `Jester` can be retrieved from DBpedia using the lookup search, although the result points to another entity. This is a clear limitation with the adopted lookup technique, which could be avoided with a more precise implementation of the lookup function. PMI usually gets the highest precision; the strategy proved effective in removing false positives, but penalises recall by excluding entities that do not appear with the words in the context list XT. For instance, `Unigene` (*Dataset*) often appears in our corpus with the word *data source*, which is not in our context list and thus filtered out. The EN strategy keeps only the entities that are preserved by two out of three (WS, KB and PMI) filtering strategies. While reducing the number of false positives, the technique proves to be too restrictive; for instance *Dataset* names such as `Yelp`, `Twitter`, `Foursquare` and `Nasdaq` are removed by both the WS and KB strategies.

4.2.3 Seed Set Size

We now analyze the performance of our approach when a different amount of seed entities are provided. We randomly initialize $T \subseteq T_{seed}$ with $|T| = 5, 10, 25, 50, 100$ (see Algorithm 4). We execute the evaluation cycle 10 times for each size of T, and again vary expansion and filtering strategies. The recall performance sharply increase with the number of seeds term ($\mu = +340\%$ from 5 to 100 seeds): this is due by the increase in the number of sentences available for NER training, and it is an expected behaviour. The decrease in precision performance is an average of -6% from 5 to 100 seeds, with an average value of -5.1% for *Dataset* and -6.9% for *Methods*. Noteworthy are the good performance that can be achieved with as little as 5 seed entities (*Datasets*: 0.25 F-score with TE strategy and no filtering).

4.2.4 Iterative NER Training

Given the results from the previous section, we investigate the ability of our approach to achieve good performance when applied in an iterative fashion. Figure 2 shows the result of the iterative NER training using Sentence Expansion with 5 seeds. We report the results with the PMI (*Dataset*) and EN (*Methods*) filtering techniques, as they are the ones offering the most balance performance in both precision and recall. Despite the small initial seed seed, in only 2 iterations it is possible to achieve precision and recall performance comparable to the ones obtained with an initial set of 100 seeds.

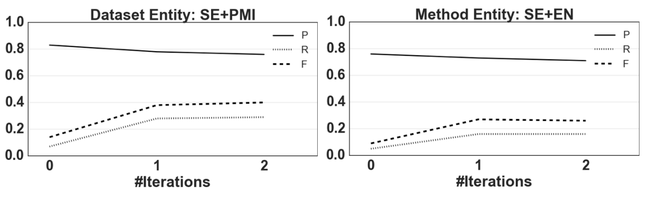

Figure 2: Dataset(left) and Method(right) entity: Iterative NER training using 5 initial seeds.

[9] `http://lookup.dbpedia.org/api/search/KeywordSearch?QueryClass=&QueryString=ratebeer`

4.2.5 Analysis of recognized entities

To improve of the analysis of the performance of our approach, we extended our result analysis beyond the 150 named entities in T_{all} that make the seed and test term sets. We manually investigated up-to-now unknown named entities which have been recognized by the NER after training. We applied a method inspired by the pooling technique typically used in information retrieval research: given a list of seed terms T_{seed} of a given type, and a list of recognized potential filtered terms FT (see Algorithm 3) of an yet unknown type, the idea is to rank the items in the list of candidate terms FT according to their embedding similarity to the items in the seed set T_{seed} and collect the top K. The similarity is measured based on the cosine similarity between the *word2vec* embedding vectors. Each entity in the lists has been manually checked and annotated by an expert. Figures 3, and 4 show the precision and recall for the entities recognized using the SE approach. As in the previous experiment, we used the PMI and EN filtering strategies respectively for *Dataset* and *Method* types. Figure 4 shows the precision and recall of the top K = 10,25,50,100, and 200 retrieved entities. Precision performance are consistently high at all level of recall, further demonstrating the quality of the approach. Table 7 shows examples of retrieved entities, where we can find *Dataset* long-tail entities such as `mslr-web10` (a benchmark collection for learning to rank method) and `ocris` (Online catalogue and repository interoperability study); and *Method* entities such as *TimedTextTank* and *StatSnowball*. Some examples of incorrect detected entities are due to ambiguous nature of the sentence. Consider the following sentence: *"The implementation of scikitlearn toolkit was adopted for these methods"*, since its similar to a sentence that contains a method entity, the entity *scikitlearn* was detected as a method although its a software library. In another sentence: *"The Research Support Libraries Programme (RSLP) Collection Description Project developed a model."*, the *RSPL* was detected as a dataset entity despite being a project due to its surrounding words like *collection* and *libraries*.

Table 7: Retrieval examples for *Dataset* and *Method*

Dataset Entities	`ukgov, ocris, ohloh, mslr-web10k, jstor, acmdl, hathitrust, pictureaustralia, europarl, ace2004`
Method Entities	`timedtextrank, keycluster, comborank, statsnowball, milnewitten, sbatcg, pwotd, guidearch, revrank, llrnumsubst`

4.3 Discussion

The core design goal of our NER approach was the minimisation of training costs in scenarios where the targeted entity types are rare, and little to no resources (for manual annotations) are available. In these cases, relying on exhaustive dictionaries or knowledge-bases is not possible, and common techniques like supervised learning cannot be applied. We believe to have successfully reached that goal, as we could show that even with small seed lists T_{seed} with little as 5 or 25 terms, high-precision NERs could be trained for our entity types with the right filtering techniques.

Nonetheless, this ease-of-training of course comes at a price: recall values are low, and are unlikely to be able

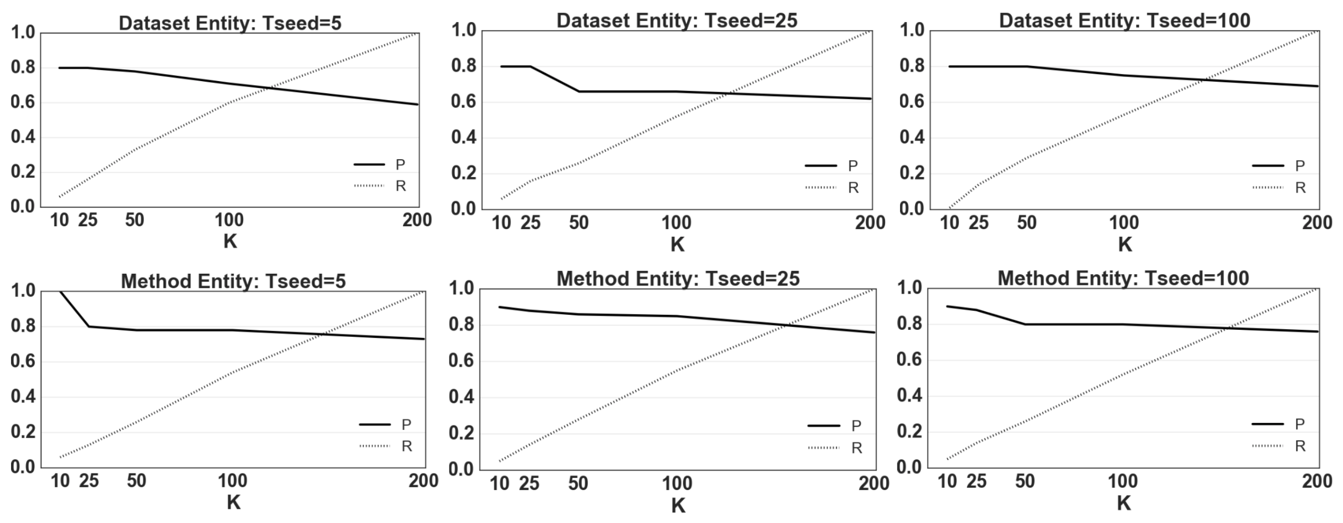

Figure 3: Dataset(top) and Method(bottom) entity:Precision and Recall@k using different number of seeds for the Sentence Expansion strategy

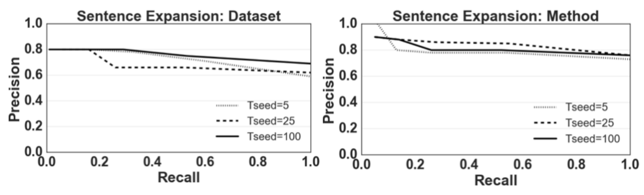

Figure 4: *Dataset* (left) and *Method* (right): Precision and Recall for ranked top 10, 25, 50, 100 and 200 entities, varying seeds sizes and Expansion strategy.

to compete with recall measures known from much more elaborately trained NERs for popular types. However, by selecting different configurations for filtering and expansion strategies, recall can be moderately improved at the cost of precision. Also, the effectiveness of such changes of configurations seems to slightly differ between our two chosen entity types. As a result, we cannot identify on clear best configuration as our approach seems to require some tuning in the field in order to achieve desired result (however, this also provides some flexibility to tune with respect to different quality requirements).

Furthermore, some of our underlying assumptions, heuristics and implementation choices, are designed as a simplistic prove-of-concepts, and deserve further discussion and refinement. As an example, consider WS WordNet filtering: we assumed domain-specific named entities would not be part of common English language. While this is true for many relevant domain-specific entities, several datasets do indeed carry common names like the `census dataset`. For a production system, more complex implementations and tailored crafting is necessary for reaching higher performance values.

Another restriction is related to the core heuristics found in the term and sentence expansion, where we assume that similar types of entities occur in similar contexts.

Threats To Validity. Our evaluation has been performed on an extensive document corpus, but on a single domain and on a limited set of entity types. While the hypothesis described in Section 3 seems to hold for both *Datasets* and *Methods*, we cannot ensure the same for other entity types in the same domain (e.g. *Software*) or in other domains. Despite the good performance achieved, it could already be noted that even between those two types – both from the same domain – no single configuration is clearly the best. In order to obtain a complete understanding of the full capabilities, limitations, and trade-offs of our approach, more studies addressing additional domains and entity types are necessary.

5. CONCLUSION

We presented a novel approach for the extraction of domain-specific long-tail entities from scientific publications. A limiting factor in this scenario is the lack of resources and/or available explicit knowledge to allow for established techniques for NER training. We explored techniques able to limit the reliance on human supervision, resulting in an iterative approach that requires a small set of seed terms of the targeted type. Our core contributions, in addition to the overall approach, are a set of expansion strategies relying on semantic similarity and relatedness between terms to increase the size and labelling quality of the training dataset generated from the seed terms, as well as several filtering techniques to control the noise introduced by the expansion.

In our evaluation, we could show that we can reach a precision of up to 0.91 and a recall of 0.41 – a good result considering the very cheap training costs. Furthermore, we could show that recall can be traded for more precision to a moderate extend by changing the configuration of our NER training process.

For future works, additional evaluation addressing more domains and entity types is of great importance, to better understand the range of applicability of the approach. Furthermore, many of our currently still simplistic heuristics and implementation choices can benefit from (domain-specific) improvement.

6. REFERENCES

[1] E. Afiontzi, G. Kazadeis, L. Papachristopoulos, M. Sfakakis, G. Tsakonas, and C. Papatheodorou. Charting the digital library evaluation domain with a semantically enhanced mining methodology. In *Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries*, pages 125–134. ACM, 2013.

[2] M. Brambilla, S. Ceri, E. Della Valle, R. Volonterio, and F. X. Acero Salazar. Extracting emerging knowledge from social media. In *Proceedings of the 26th International Conference on World Wide Web*, pages 795–804. International World Wide Web Conferences Steering Committee, 2017.

[3] K. Chakrabarti, S. Chaudhuri, V. Ganti, and D. Xin. An efficient filter for approximate membership checking. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 805–818. ACM, 2008.

[4] A. Chandel, P. Nagesh, and S. Sarawagi. Efficient batch top-k search for dictionary-based entity recognition. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pages 28–28. IEEE, 2006.

[5] C. Chen. CiteSpace II: Detecting and visualizing emerging trends and transient patterns in scientific literature. *Journal of the American Society for information Science and Technology*, 57(3):359–377, 2006.

[6] R. Gaizauskas, G. Demetriou, P. J. Artymiuk, and P. Willett. Protein structures and information extraction from biological texts: the pasta system. *Bioinformatics*, 19(1):135–143, 2003.

[7] S. Goldberg, D. Z. Wang, and C. Grant. A probabilistically integrated system for crowd-assisted text labeling and extraction. *Journal of Data and Information Quality (JDIQ)*, 8(2):10, 2017.

[8] D. Graus, D. Odijk, and M. de Rijke. The birth of collective memories: Analyzing emerging entities in text streams. *arXiv preprint arXiv:1701.04039*, 2017.

[9] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235, 2004.

[10] S. Gupta and C. D. Manning. Analyzing the dynamics of research by extracting key aspects of scientific papers. In *In Proceedings of IJCNLP*. Citeseer, 2011.

[11] Z. Harris. Distributional Structure. *Word*, 10:146–162, 1954.

[12] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics, 1992.

[13] A. Hoonlor, B. K. Szymanski, and M. J. Zaki. Trends in computer science research. *Communications of the ACM*, 56(10):74–83, 2013.

[14] P. Isenberg, T. Isenberg, M. Sedlmair, J. Chen, and T. Möller. Visualization as Seen Through its Research Paper Keywords. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):771–780, 2017.

[15] A. Judea, H. Schütze, and S. Brügmann. Unsupervised training set generation for automatic acquisition of technical terminology in patents.

[16] M. Kejriwal and P. Szekely. Information extraction in illicit web domains. In *Proceedings of the 26th International Conference on World Wide Web*, pages 997–1006. International World Wide Web Conferences Steering Committee, 2017.

[17] M. L. Khodra, D. H. Widyantoro, E. A. Aziz, and B. R. Trilaksono. Automatic tailored multi-paper summarization based on rhetorical document profile and summary specification. *Journal of ICT Research and Applications*, 6(3):220–239, 2012.

[18] S. Kim, Z. Lu, and W. J. Wilbur. Identifying named entities from pubmed® for enriching semantic categories. *BMC bioinformatics*, 16(1):57, 2015.

[19] J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, volume 951, pages 282–289, 2001.

[20] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014.

[21] X. Liu, F. Wei, S. Zhang, and M. Zhou. Named entity recognition for tweets. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(1):3, 2013.

[22] C. LOFI. Measuring semantic similarity and relatedness with distributional and knowledge-based approaches. *Information and Media Technologies*, 10(3):493–501, 2015.

[23] P. Lopez. GROBID: Combining Automatic Bibliographic Data Recognition and Term Extraction for Scholarship Publications. In *European Conference on Digital Library (ECDL)*, Corfu, Greece, 2009.

[24] M. MARRERO and J. URBANO. A semi-automatic and low-cost method to learn patterns for named entity recognition. *Natural Language Engineering*, pages 1–37, 2017.

[25] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[26] F. Osborne, H. de Ribaupierre, and E. Motta. Techminer: extracting technologies from academic publications. In *Knowledge Engineering and Knowledge Management: 20th International Conference, EKAW 2016, Bologna, Italy, November 19-23, 2016, Proceedings 20*, pages 463–479. Springer, 2016.

[27] M. Paşca. Weakly-supervised discovery of named entities using web search queries. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 683–690. ACM, 2007.

[28] K. Pastra, D. Maynard, O. Hamza, H. Cunningham, and Y. Wilks. How feasible is the reuse of grammars for named entity recognition. In *In Proceedings of the*

3rd Conference on Language Resources and Evaluation (LREC), Canary Islands. Citeseer, 2002.

[29] L. Qu, G. Ferraro, L. Zhou, W. Hou, and T. Baldwin. Named entity recognition for novel types by transfer learning. In *EMNLP*, 2016.

[30] R. Reinanda, E. Meij, and M. de Rijke. Document filtering for long-tail entities. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 771–780. ACM, 2016.

[31] A. Ritter, S. Clark, O. Etzioni, et al. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics, 2011.

[32] B. Sateli and R. Witte. What's in this paper?: Combining rhetorical entities with linked open data for semantic literature querying. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1023–1028. ACM, 2015.

[33] D. Shen, J. Zhang, J. Su, G. Zhou, and C.-L. Tan. Multi-criteria-based active learning for named entity recognition. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 589. Association for Computational Linguistics, 2004.

[34] K. Shubankar, A. Singh, and V. Pudi. A frequent keyword-set based algorithm for topic modeling and clustering of research papers. In *Data Mining and Optimization (DMO), 2011 3rd Conference on*, pages 96–102. IEEE, 2011.

[35] T. Siddiqui, X. Ren, A. Parameswaran, and J. Han. Facetgist: Collective extraction of document facets in large technical corpora. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 871–880. ACM, 2016.

[36] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine learning*, 34(1):233–272, 1999.

[37] M. Song, G. E. Heo, and S. Y. Kim. Analyzing topic evolution in bioinformatics: investigation of dynamics of the field with conference data in DBLP. *Scientometrics*, 101(1):397–428, 2014.

[38] Y. Tateisi, Y. Shidahara, Y. Miyao, and A. Aizawa. Annotation of Computer Science Papers for Semantic Relation Extrac-tion. In *LREC*, pages 1423–1429, 2014.

[39] K. Tomanek and U. Hahn. Reducing class imbalance during active learning for named entity annotation. In *Proceedings of the fifth international conference on Knowledge capture*, pages 105–112. ACM, 2009.

[40] C.-T. Tsai, G. Kundu, and D. Roth. Concept-based analysis of scientific literature. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1733–1738. ACM, 2013.

[41] S. Tuarob, S. Bhatia, P. Mitra, and C. L. Giles. Algorithmseer: A system for extracting and searching for algorithms in scholarly big data. *IEEE Transactions on Big Data*, 2(1):3–17, 2016.

[42] W. Yu, X. Lin, W. Zhang, Y. Zhang, and J. Le. Simfusion+: Extending simfusion towards efficient estimation on large and dynamic networks. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 365–374. ACM, 2012.

[43] X. Zhang, Y. Chen, J. Chen, X. Du, K. Wang, and J.-R. Wen. Entity set expansion via knowledge graphs. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1101–1104. ACM, 2017.

[44] Z. Zhang, L. Sun, and X. Han. A joint model for entity set expansion and attribute extraction from web search queries. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.