TSE-NER: An Iterative Approach for Long-Tail Entity Extraction in Scientific Publications

Sepideh Mesbah, Christoph Lofi, Manuel Valle Torre, Alessandro Bozzon, Geert-Jan Houben

Delft University of Technology, Mekelweg 4, 2628 CD Delft, the Netherlands
{s.mesbah, c.lofi, m.valletorre, a.bozzon, g.j.p.m.houben}@tudelft.nl

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. "WebKB", "StatSnowball") are rare, often relevant only in specific knowledge domains, yet important for retrieval and exploration purposes. State-of-the-art NER approaches employ supervised machine learning models, trained on expensive type-labeled data laboriously 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 an iterative approach for training NER and NET classifiers in scientific publications that relies on minimal human input, namely a small seed set of instances for the targeted entity type. We introduce different strategies for training data extraction, semantic expansion, and result entity filtering. We evaluate our approach on scientific publications, focusing on the long-tail entities types *Datasets*, *Methods* in computer science publications, and *Proteins* in biomedical publications.

1 Introduction

The growth of domain-specific knowledge available as digital text demands more effective methods for querying, accessing, and exploring document collections. Scientific publications are a compelling example: online digital libraries (e.g. IEEE Xplore) contain hundreds of thousands documents; yet, the available retrieval functionality is often limited to keyword/faceted search on *shallow* metadata (e.g. title, terms in abstract). A query like *retrieve the publications that used a social media dataset for food recipe recommendation* is bound to return unsatisfactory results.[1]

Named entities, obtained through an analysis of a document's content, are an effective way to achieve better retrieval and exploration capabilities. Automatic *Named Entity* Recognition and Typing (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.

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

To perform well, state-of-the-art NER/NET methods [2,8] either require comprehensive domain knowledge (e.g. to specify matching rules), or rely on a large amount of human-labeled training data for machine learning – both solutions are expensive and time-consuming.

A cheaper alternative is to generate labeled training data by obtaining existing instances of the targeted entity type from Knowledge Bases (KBs) [2] - this of course requires that the desired entity type is well-covered in the KB.

Problem Statement. While achieving impressive performance with high-recall named entities (e.g. locations and age) [8], generic NER/NETs show their limits with domain-specific and long-tail entity types. Consider the following sentence: *"We evaluated the performance of SimFusion+ on the WebKB dataset"*. Despite *WebKB*[2] being a popular dataset in the Web research community, generic 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.

Literature [16,19,22,23] shows 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 [18]; and 2) the high cost associated with the creation of hand-crafted rules, or human-labeled training datasets for supervised machine learning techniques. Few approaches addressed these problems by relying on bootstrapping [23] or Entity Expansion [2,8] techniques, achieving promising performance. However, how to train high-performance *long-tail* Entity Extraction and Typing with minimal human supervision remains an open research question.

Original Contribution. We contribute TSE-NER, an iterative approach for training NER/NET classifiers for long-tail entity types that exploits Term and Sentence Expansion. TSE-NER relies on minimal human input – a seed set of instances of the targeted entity type. We introduce different strategies for training data extraction, semantic expansion, and result entity filtering. Different combinations of these strategies allow to tune the technique for either higher recall or higher precision scenarios.

We performed extensive evaluations comparing to state-of-the-art methods, and assess several sentence expansion and term filtering strategies. As our core use case, we focus on 15,994 data science publications from 10 conference series with the *Dataset* (e.g. `Imagenet`) and data processing *Methods* (e.g. `LSTM`) long-tail entity types. We show that our approach is able to consistently outperform previous low-cost supervision methods, even with small amount of training information: with a seed set of 100 entities, our approach can achieve precision up to 0.91 when tuned for precision, and recall up to 0.41 when tuned for recall, or 0.77 and 0.30 for a balanced setting. When applied in an iterative fashion, our approach can achieve comparable performance with an initial seed set of only 5 entities. We show that 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). To study the

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

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

performance of TSE-NER across scientific domains, we processed 4,525 biomedical publications focusing on *Protein* (e.g. Myoglobin) entity type. Evaluation on the Craft corpus [1] shows that TSE-NER can achieve performance comparable to existing dictionary-based systems, and obtain precision up to 0.40 and recall up to 0.28 with just 25 seed terms. TSE-NER is implemented in the *SmartPub* platform [13]; its source code is available on the companion Website [14], and its application shown in the video screencast at the following address: `https://youtu.be/zLLMwOT5sZc`.

Outline. The remainder of the 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

A considerable amount of literature published in recent years addressed the *deep* analysis of text. Common approaches for *deep* analysis of publications rely on techniques such as bootstrapping [23], word-frequency analysis [21], probabilistic methods like Latent Dirichlet Allocation [5], etc. In contrast to current research [21] which limits the analysis of a publication's content to its title, abstract, references, and authors, we extract entity instances from the much richer full text. In addition, our method does not rely on existing knowledge bases [19,16] and it is not based on selecting the most frequent keywords [21]. More recent research [22] 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 automatically trained distributed word representations to train a NER in iterative steps (i.e. 2-3).

Entity Instances Extraction Named Entity Recognition (NER) has been applied to identify both entity types of general interest (e.g. Person, Location, Cell, Brand, etc.) as well as for specific domains (e.g., medicine or other domain where resources for training a NER are easily available). NERs rely on different approaches such as dictionary-based, rule-based, machine-learning [22] or hybrid (combination of rule based and machine learning) [25] techniques. Despite its high accuracy, a major drawback of dictionary-based approaches is that they require an exhaustive dictionary of domain terms, which are expensive to create and many smaller domains lack the resources to do so. The same holds for rule-based techniques, which rely on formal languages to express rules and require comprehensive domain knowledge and time to create.

Bootstrapping and Entity Set Expansion. Most current NERs are based on Machine Learning techniques, which require a large corpus of labeled training text [6]. Again, the high costs of data annotation is one of the main challenges in adopting specialized NER for rare entity types in specialized domains [22]. 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 machine learning methods [4].

Transfer learning techniques [17] use the knowledge gained from one domain and apply it to a different but related named entity type. In contrast to previous work, we do not require a large training corpus [17] for transfer learning; also, our approach differs from works on high-recall entity extractors (e.g. with regular expression extractors) for detecting entity types such as location and age [8].

Entity Set Expansion is a technique finding similar entities to a given small set of seed entities [2,8]. Bootstrapping [23] is another approach similar to our method that uses seed terms and extracts features such as unigrams, bigrams, left unigram, closest verb, etc. These are used to annotate more concept mentions which leads to extracting new features. This step operates in an iterative fashion until no new features are detected. Our approach is inspired by Entity Set Expansion and bootstrapping, but relies on different expansion strategies and does not require concepts already being available in knowledge bases [2].

3 Approach

Our TSE-NER (Term and Sentence Expansion) approach for domain-specific long-tail entity recognition is organized in five steps, as shown in Figure 1.

❶ An initial set of seed terms is used to identify a set of sentences used as initial *training data* (Section 3.1). ❷ *Expansion* strategies can be used to expand the set of initial seed terms, and the *training data* sentences (Section 3.2). ❸ The *Training Data Annotation* step annotates the training data using the (possibly expanded) seed terms set (Section 3.3). ❹ 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). ❺ The *Filtering* step refines the set candidate entities set, to improve the quality of outputted *Verified Terms* set (Section 3.5).

TSE-NER operates under the hypothesis that there are recurring patterns in the mentions of domain-specific named entities, and that they appear 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. The process can be iterated, by repeating the first step using the newly detected terms as seeds to generate new training data. We rely on the following concepts (some are only relevant for the evaluation, and could be omitted in setups where evaluation is not necessary). The companion website [14] provides a complete unified algorithm covering the TSE-based NER training workflow.

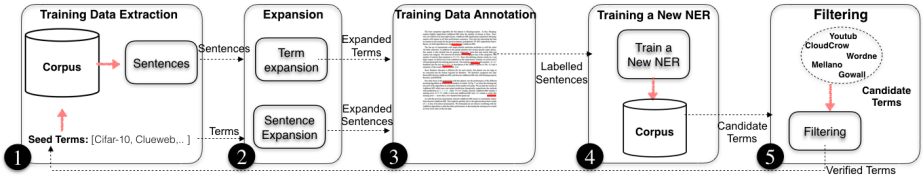

Fig. 1. Overview of the domain-specific long-tail named entities recognition approach.

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 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 use case, as discussed in section 4.3).

3.1 Training Data Extraction

As a first step, 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: $S_i := \cup_{t \in T_i} \{s | s \in S \wedge (t \in s \vee t \in successor(s) \vee t \in predecessor(s))\}$.

3.2 Expansion

The small size of the seed term set T_{seed} has two obvious shortcoming that can greatly hinder the accuracy and recall of the trained NERs: 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. In addition, the *generalization* capability of the NER for identifying new named entities can also be affected: an insufficient amount of positive examples can lead to entities of the targeted type being labeled negatively; while the extraction of sentences in the training data that are related to seed terms will cause a shortage of negative examples. To account for these issues, we designed two *expansion* strategies.

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 scientific 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 interface $expandTerms(terms_s)$, with $terms_s \subseteq terms_i$. While many different implementations for this interface are possible, in this work we use *semantic similarity*: terms which are semantically similar 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 similar terms like `trec-2005`.

We exploit the distributional hypothesis [7] stating that terms frequently occurring in similar context are semantically related, using the popular *word2vec* implementation of skip-n-gram word embeddings [15]. In essence, *word2vec* embeds each term of a large document corpus into low-dimensional vector space (100 dimensions in our case), and the cosine distance between two vectors has been shown to be a high-quality approximation of semantic relatedness [11]. In our implementation, we trained the *word2vec* model on the whole development sentence collection S, as described in [15], learning all uni- and bigram 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 sometimes faulty assumption that all terms in cluster are indeed similar as *word2vec* relatedness is not always reliable for similarity measurements [11]. To improve, in the following we only consider terms terms which are likely to be named entities by using NLTK entity detection to obtain a list of all entities E_{all} contained in S.[4] This results in the Algorithm 1.

Algorithm 1 TE 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

Sentence Expansion (SE). A second (optional) measure to increase the size and variety of the training set 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 negatives sentences not containing instances of the desired type, but are still very similar in semantics and vocabulary.

[4] NLTK entity detection is based on grammatical context. It does not perform any typing, and due to it's simplicity, has high recall values.

We rely on *doc2vec* document embeddings [10], a variant of *word2vec*, to learn vector representations of all sentences. For each sentence in S, we use the cosine distance to discover the most similar sentences filtered to those not containing any known instance of the targeted type. As such sentences might contain an unknown instance of that type, we always combine SE with term expansion to minimize the risk of accidentally mislabeling them as negative examples.

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 $A_{T_i} := annotate_{T_i}(S_i)$ in all training sentences if 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 can be trained.

3.4 NER Training

For training a new NER_i, we used the Stanford NER tagger[5] 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 [9]; 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 the companion website. 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.

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 false positives (like the expansion steps, but also the NER itself), the goal of this last (optional) step is to filter out candidate terms that are unlikely of the targeted type using a set of external heuristics with different assumptions:

Wordnet + Stopwords (WS) Filtering. In the domain-specific language of scientific documents, it is common for named entities to be "proper" of that 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*). This is achieved by performing lookup operations in WordNet[6] and in common lists of stopwords.[7] As both sources focus on general English language, only domain-specific terms should be preserved.

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

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

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

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) 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.

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 [20], as we manually compile a list of context terms / patterns CX 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.

Knowledge Base Lookup (KBL) Filtering. Our target are long-tail domain-specific entities, i.e. entities that are not part of existing knowledge bases. Named entities that could be linked to a knowledge base could be assumed 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.

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 preserve the entities that are passed through two out of three selected filtering strategies.

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 utilize 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

Corpora. Our main evaluation, shown in the following sections, is performed on the data science (15,994 papers from 10 conference series) domain. To assess the performance of TSE-NER in other scientific domains, at the end of the section

we describe an experiment over 4,525 publications from 10 biomedical journals. The full description of the corpora is described in the companion Web site [14]. Publications are processed using GROBID [12], to extract a structured full-text representation of their content.

Long tail entity types selection. Scientific publications contain a large quantity of long-tail named entities. Focusing on the data science domain, 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/analyze a dataset). Both entities types are scarcely represented in existing knowledge bases.[8] To evaluate the performance of our approach, we create a set of 150 seed instances T_{all} for each targeted type, collected public from public websites.[9]

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}.

Evaluation Dataset. As discussed in Section 3, in the training process all test sentences S_{test} (i.e. sentences mentioning terms in T_{test}) in the corpus D_{all} are removed. For evaluation, 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 $A_{test} := annotate_{expert}(S_{annotated})$.

Details of statistics on sentences used for training and testing can be found in the companion Web site. For training, depending on the seed set size between 5 and 100, we used between 198 and 2863 sentences for the *dataset* entity type and 617 to 18545 sentences for the *Method* entity type.

Algorithm 2 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

For testing 50 seed terms were used for both dataset (i.e. 3149 sentences) and method (i.e. 1097 sentences) entity type. The evaluation protocol is described in Algorithm 2, where the $seed\_size$ values can be initialized with different values. Our analysis was not limited to the 50 test seed terms, we further evaluated 200 entities recognized by TSE-NER via a pooling technique.

4.2 Results

For a given entity type (*Dataset* and *Method*), we test the performance with differently sized seed sets and expansion strategies to create the training data

[8] 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.

[9] 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.

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, and EN strategies, plus a combination of the WS, ST, and KBL strategies, as listed in the caption of Table 1. The complete evaluation results for all the seed set size and the filtering techniques can be found in the companion Web site. We also perform an experiment to test the performance of our approach when applied iteratively. We analyze the performance of the model on the manually annotated test set presented in the previous section.

Tables 1 and 2 summarize the performance achieved for *Dataset* and *Method* entity types. In Table 2, the *No Expansion* and *Term Expansion* figures for the *Method* type are omitted for brevity's sake. Our 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 obtained with the *Method* 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 recall, thus demonstrating their effectiveness for generalization. On average, filtering 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 the different trade-offs our approach can strike: different configurations of filtering and expansion 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.

Expansion Strategies. Expansion strategies increase the size and variety of training datasets, thus improving the precision and recall. Both strategies achieve the expected results, although with different performance increase: compared to *NE* strategy, both TE and SE achieve a considerable performance boost (μ = +190%) for recall, but at cost of lower precision ($\mu = -8.7\%$). We account the

Table 1. *Dataset* entity type: Precision/Recall/F-score on evaluation dataset. Legend: *NE* – No Expansion; *TE* – Term Expansion; *SE* – Sentence Expansion; *NF* – No Filtering; *WS* – Wordnet + StopWords; *SS* – Similar Terms + WS; *KS* – Knowledge Base Lookup + SS; *PM* – Point-wise Mutual Information; *EN* – Ensemble.

Strategy	***#S***	***NF***	***WS***	***SS***	***KS***	***PM***	***EN***
NE	5	.83/**.05**/**.10**	.84/.04/.08	.86/.03/.07	.75/.01/.01	**.90**/.04/.09	.86/.04/.08
	25	.84/**.08**/**.16**	.83/.07/.13	.86/.07/.13	.78/.01/.03	**.91**/**.08**/.15	.85/.07/.13
	100	.85/**.15**/**.26**	.85/.13/.22	.87/.12/.22	.82/.03/.07	**.91**/.13/.24	.86/.12/.22
TE	5	.76/**.14**/**.25**	.78/.13/.22	.79/.11/.20	.74/.04/.09	**.83**/.13/.23	.80/.13/.22
	25	.72/**.29**/**.42**	.73/.28/.40	.75/.27/.40	.73/.17/.28	**.77**/.27/.40	.75/.27/.40
	100	.69/**.41**/**.51**	.70/.39/.50	.71/.38/.50	.71/.28/.40	**.74**/.38/.50	.72/.38/.50
SE	5	.83/**.07**/**.14**	.84/.06/0.12	.86/.05/.10	.82/.01/.02	**.91**/**.07**/.13	.86/.06/.11
	25	.81/**.22**/**.35**	.80/.18/.29	.83/.17/.29	.77/.04/.08	**.89**/.20/.33	.82/.18/.29
	100	.77/**.30**/**.43**	.77/.24/.37	.80/.23/.36	.78/.07/.13	**.86**/.26/.40	.79/.24/.37

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 instances due to their presence in the same embedding clusters as the seed terms (see also Section 3.2). The SE strategy shows lower recall ($\mu = +210\%$ over NE), but with less precision loss ($\mu = -5.2\%$ than NE). We account this positive behaviour to the presence of more quality negative examples, helping to maintain the generalization capabilities of the NER, while refining the quality of its recognition.

Filtering Strategies. We 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 penalized 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.

For instance, the *Dataset* entities Ratebeer[10] 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 penalizes recall by excluding entities that do not appear with the words in the context list CX. For instance, Unigene (*Dataset*) often appears in with the term *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, this 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.

Seed Set Size. We randomly initialize $T \subseteq T_{seed}$ with $|T| = 5, 10, 25, 50, 100$ (see Algorithm 2). 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 to the increase in the number of sentences available for NER training, and is an expected behaviour. The decrease in precision is an average of -6%

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

Table 2. *Method* entity type: Precision/Recall/F-score. Legend as in Table 1.

Strategy	***#S***	***NF***	***WS***	***SS***	***KS***	***PM***	***EN***
SE	5	.76/**.04**/**.08**	.77/.03/.07	.77/.01/.01	.84/.01/.01	**.86**/.01/.03	.84/.03/.05
	25	.77/**.14**/**.24**	.77/.12/.21	.79/.09/.16	**.87**/.05/.09	.86/.05/.09	.85/.09/.17
	100	.68/**.15**/**.25**	.67/.14/.23	.65/.12/.20	.84/.07/.13	**.85**/.05/.10	.83/.10/.19

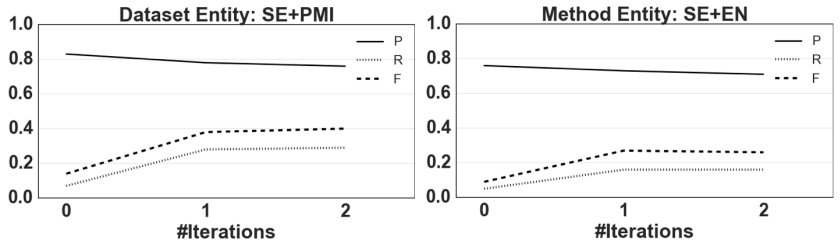

Fig. 2. Dataset(L) and Method(R) entity: Iterative NER training using 5 initial seeds.

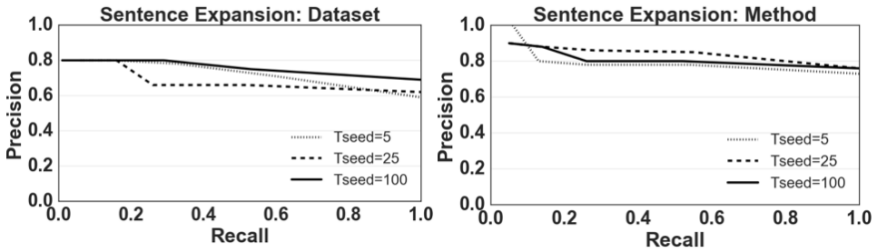

Fig. 3. *Dataset* (L) and *Method* (R): Precision and Recall for ranked top 10, 25, 50, 100 and 200 entities, varying seeds sizes.

from 5 to 100 seeds, with an average value of −5.1% for *Dataset* and −6.9% for *Methods*. Noteworthy are the good performance with as little as 5 seed entities (*Datasets*: 0.25 F-score with TE strategy and no filtering).

Iterative NER Training. 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, as they are the ones offering the most balanced performance in both precision and recall. Despite the small initial seed seed, it is possible to achieve precision and recall comparable to the ones obtained with an initial set of 100 seeds in only 2 iterations.

Analysis of recognized entities. To widen the scope of our evaluation, we extended our result analysis beyond the 150 named entities in T_{all}. 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 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. As a result, the obtained precision and recall measurements are only approximate values. The similarity is measured based on the cosine similarity between the *word2vec* embedding vectors. Each entity in the lists has been manually checked by an expert. Figure 3 shows the precision and recall of the top $K = 10$, 25, 50, 100, and 200 retrieved entities using the SE approach. As in the previous experiment, we used the PMI and EN filtering strategies respectively for *Dataset* and *Method* types. Precision performance are consistently high at all level of recall.

The *Dataset* entities `mslr-web10` (a benchmark collection for learning to rank method) and `ace2004` (ACE 2004 Multilingual Training Corpus); and *Method* entities such as *TimedTextTank* and *StatSnowball* are a sample of extracted entities. More examples can be found in the companion website. 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 it is 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."*, *RSPL* (a project) was detected as a dataset due to its surrounding words (e.g. *collection*, *libraries*).

Comparison with State-of-the-art. We compared our method with: 1) the BootStrapping (BS) based concept extraction approach [23], a commonly used state-of-the-art technique in scientific literature; the experiments where executed with the code and the parameters (k, n, t) to (2000, 200, 2) provided in [23], and with 100 seeds. And, 2) improved and expanded Hearst Pattern (HP) [20] for automatically building or extending knowledge bases extracting type-instance relations e.g., X such as Y as in "we used datasets such as twitter". Intuitively, the performance of BS decreases with less number of seed terms. For the HP we kept type-instance pairs related to dataset or method (i.e. the context words in CX). Experiments on our evaluation dataset shown that TSE-NER achieved higher performance in terms of precision/recall/fscore for the *dataset* entity type (0.77/0.30/0.43) compared to *BS* (0.08/0.13/0.10) and *HP* (0.92/0.15/0.27) as well as for the *method* entity (*TSE-NER*: 0.68/0.15/0.25, *BS*: 0.11/0.32/0.16, *HP*: 0.64/0.04/0.07). The high precision and low recall in *HP* is explained by the limited set of *HP* patterns. We infer that different expansion strategies augment the performance of our technique compared to the *BS* which just relies on features such as unigrams, bigrams, closest verb, etc.

Biomedical Domain. To test the performance of TSE-NER on another scientific domain, we processed 4,525 biomedical publications from 10 journals focusing on the *Protein* entity type. The seed terms were selected from the protein ontology.[11] We excluded the test terms appearing in the Craft corpus [1] (a manually annotated corpus containing 67 full-text biomedical journals) and kept only the ones that have a reference in the publications. The list of seed terms used for the *Protein* entity are listed on the companion site. We randomly initialized $T \subseteq T_{seed}$ with $|T| = 5, 25, 100$ and employed the SE strategy and a simple WS filtering. The evaluation cycle has been executed 10 times for each size of T, and results are averaged. TSE-NER can achieve precision/recall/f-score of 0.57/0.08/0.14 using 5 seeds, 0.40/0.28/0.32 using 25 seeds, and 0.38/0.46/0.41 with 100 seeds. The latter results are comparable to extensive dictionary-based systems [24] (0.44/0.43/0.43) [3] (0.57/0.57/0.57) where existing ontologies in the biomedical domain are used for matching *Protein* entities of the text.

[11] `http://obofoundry.org/ontology/pr.html`

4.3 Discussion

The design goal of our TSE-NER approach was minimizing the 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.

Nonetheless, this ease-of-training comes at a price: recall values are low, and are unlikely to be able to compete with known much more elaborately trained NERs for popular types. However, by selecting different configurations for filtering and expansion, recall can be moderately improved at the cost of precision. Also, the effectiveness of such changes of configurations seems to slightly differ between the *Dataset* and *Method* entity types. As a result, we cannot identify one clear best configuration as TSE-NER seems to benefit from some entity type-specific tuning. However, this also provides some flexibility to tune with respect to different quality and application 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 (for instance) 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 – which is not necessarily always the case.

Threats To Validity. Our evaluation has been performed on an extensive document corpus, covering two distinctively different domains. However, we focused only on a limited set of entity types. The hypothesis described in Section 3 hold for *Datasets*, *Methods*, and *Proteins*, but further experiments are needed for other entity types in the same domains (e.g. *Software*) or in other domains. Despite the good performance achieved, it could already be noted that even between those three types, no single TSE-NER 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 NER

training techniques. We explored techniques able to limit the reliance on human supervision, resulting in an iterative approach that requires only 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 exploiting 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, or a recall of up to 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 work, additional evaluation addressing more domains and entity types is of importance to better understand the range of applicability of our approach. Also, many of our currently still simplistic heuristics and implementation choices can benefit from (domain-specific) improvement and optimization.

References

1. M. Bada, M. Eckert, D. Evans, K. Garcia, K. Shipley, D. Sitnikov, W. A. Baumgartner, K. B. Cohen, K. Verspoor, J. A. Blake, et al. Concept annotation in the craft corpus. *BMC bioinformatics*, 13(1):161, 2012.
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. C. Funk, W. Baumgartner, B. Garcia, C. Roeder, M. Bada, K. B. Cohen, L. E. Hunter, and K. Verspoor. Large-scale biomedical concept recognition: an evaluation of current automatic annotators and their parameters. *BMC bioinformatics*, 15(1):59, 2014.
4. 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.
5. T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235, 2004.
6. M. Habibi, L. Weber, M. Neves, D. L. Wiegandt, and U. Leser. Deep learning with word embeddings improves biomedical named entity recognition. *Bioinformatics*, 33(14):i37–i48, 2017.
7. Z. Harris. Distributional Structure. *Word*, 10:146–162, 1954.
8. 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.
9. J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Int. Conf. on Machine Learning*, volume 951, pages 282–289, 2001.
10. 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.

11. C. Lofi. Measuring semantic similarity and relatedness with distributional and knowledge-based approaches. *Information and Media Technologies*, 10(3):493–501, 2015.
12. P. Lopez. GROBID: Combining Automatic Bibliographic Data Recognition and Term Extraction for Scholarship Publications. In *European Conference on Digital Library (ECDL)*, Corfu, Greece, 2009.
13. S. Mesbah, C. Lofi, A. Bozzon, and G.-J. Houben. Smartpub: A platform for long-tail entity extraction from scientific publications. In To Appear *in Proceedings of The Web Conference 2018 Companion Volume*, 2018.
14. S. Mesbah, C. Lofi, A. Bozzon, and G.-J. Houben. Tse-ner companion page. In `https://sites.google.com/view/iswc2018/`, 2018.
15. 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.
16. 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.
17. L. Qu, G. Ferraro, L. Zhou, W. Hou, and T. Baldwin. Named entity recognition for novel types by transfer learning. In *EMNLP*, 2016.
18. 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.
19. 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.
20. J. Seitner, C. Bizer, K. Eckert, S. Faralli, R. Meusel, H. Paulheim, and S. P. Ponzetto. A large database of hypernymy relations extracted from the web. In *LREC*, 2016.
21. 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.
22. T. Siddiqui, X. Ren, A. Parameswaran, and J. Han. Facetgist: Collective extraction of document facets in large technical corpora. In *Int. Conf. on Information and Knowledge Management*, pages 871–880. ACM, 2016.
23. 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.
24. E. Tseytlin, K. Mitchell, E. Legowski, J. Corrigan, G. Chavan, and R. S. Jacobson. Noble–flexible concept recognition for large-scale biomedical natural language processing. *BMC bioinformatics*, 17(1):32, 2016.
25. 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.