Explore and Exploit. Dictionary Expansion with Human-in-the-Loop

Anna Lisa Gentile, Daniel Gruhl, Petar Ristoski(✉), and Steve Welch

IBM Research Almaden, San Jose, CA, USA
{annalisa.gentile,petar.ristoski}@ibm.com, {dgruhl,welchs}@us.ibm.com

Abstract. Many Knowledge Extraction systems rely on semantic resources - dictionaries, ontologies, lexical resources - to extract information from unstructured text. A key for successful information extraction is to consider such resources as evolving artifacts and keep them up-to-date. In this paper, we tackle the problem of dictionary expansion and we propose a human-in-the-loop approach: we couple neural language models with tight human supervision to assist the user in building and maintaining domain-specific dictionaries. The approach works on any given input text corpus and is based on the *explore and exploit* paradigm: starting from a few seeds (or an existing dictionary) it effectively discovers new instances (explore) from the text corpus as well as predicts new potential instances which are not in the corpus, i.e. *"unseen"*, using the current dictionary entries (exploit). We evaluate our approach on five real-world dictionaries, achieving high accuracy with a rapid expansion rate.

1 Introduction

Dictionary expansion [17] is one area where close integration of humans into the discovery loop has been shown to enhance task performance substantially over more traditional post-adjudication methods. This is not surprising, as dictionary membership is often a fairly subjective judgment (e.g., should a fruit dictionary include tomatoes?) [18]. Thus even with a system which finds "similar" terms (e.g., word2vec [14]) guidance is important to keep the system focused on the subject matter expert's notion of the lexicon.

In this work we propose a feature agnostic approach for dictionary expansion based on neural language models, such as word2vec and bidirectional Long Short Term Memory (BiLSTM) models [19]. To prevent semantic drift during the dictionary expansion, we effectively include human-in-the-loop. Given as input a text corpus and a set of seed examples, the proposed approach runs in two phases, *explore* and *exploit* (*EnE*), to identify new potential dictionary entries. After each iteration the subject matter expert adjudicates the proposed candidates by model, and resumes the next iteration. The *explore* phase identifies terms in the corpus that are similar to the seed entries: similarity is calculated using term vectors from the neural language model. The *exploit* phase constructs

P. Hitzler et al. (Eds.): ESWC 2019, LNCS 11503, pp. 131–145, 2019.
https://doi.org/10.1007/978-3-030-21348-0_9

additional complex multi-token terms based on all the current dictionary items. Identifying similar multi-token (or n-gram) terms is challenging for any word2vec style system, as they need to be "known" prior to model creation. A common solution to identify multi-token terms is to identify tokens that often appear together and therefore are probably a single term [14]: defining a threshold on their co-occurrence score can help identify valid n-grams. Once all valid terms are detected the term vector model can be built and while the vectors can be always updated, the terms themselves will always remain unchanged after the model is built. There are countless situations where having a static set of terms is not desirable. Dictionaries which have been automatically constructed are rarely complete - and it has been shown that even manually built dictionaries, that might be considered "complete", will inevitably evolve with time [12]. The most simple cause of incompleteness is that, when automatically constructing a dictionary from a corpus, valid terms may simply not occur. This is particularly true when considering multi-token terms, where the single different tokens might be present in the corpus but not in all their valid combinations - as an example, a medical corpus may contain the term *acute joint pain* but not the term *chronic hip pain*, which is however likely to occur in future texts from the same source. Including those "unseen" but valid terms in the dictionary is beneficial for all subsequent information extraction operations which are going to leverage it as a resource.

While the *explore* phase of our novel approach is similar in fashion to existing vector space models techniques, the combination of the *exploit* phase allows the dictionary extraction to go beyond the terms that actually occur in the input corpus. New phrases are generated by analyzing all the tokens (taken separately) from all the terms already in the dictionary. We generate new multi-token terms in two ways: (i) by *modification*, i.e. replacing certain tokens in an existing term with similar ones from the text corpus - e.g., "abnormal behavior" can be modified to "strange behavior"; (ii) by *extension*, i.e. concatenating additional tokens to an existing term, with related tokens from the text corpus, e.g., staring from the existing terms "abnormal blood count", "blood clotting" and "clotting problems" we can generate "abnormal blood clotting problems". The rationale is that many valid multi-token terms might not appear in a text corpus but there is enough statistical evidence to support their addition to the dictionary.

The main contribution of this work is the combination of the discriminative and the generative approaches (*explore* and *exploit*) in a highly interactive supervised model. Completely unsupervised, or even infrequently supervised models, are not particularly effective for dictionary generation. Without targeted and frequent adjudication of the extracted items many spurious results would be generated, thus requiring a heavy post-processing phase. We argue that close supervision results in a much more performant system. We show that high promptness of the human-in-the-loop (tighter computer/human partnership) results in nearly perfect performance of the system, i.e., nearly all the candidates identified by the system are valid entries in the dictionary. Specifically, we build targeted dictionaries and show that when requesting feedback every small batch of added terms (e.g. every 10 terms) we are able to construct much bigger valid dictionaries (more than double the size) as opposed to a single human post-processing evaluation of an automatically created dictionary.

We evaluate our approach on five real-world dictionaries, achieving high accuracy with a rapid expansion rate. Furthermore, we evaluate the importance of generating "future proof" dictionaries, i.e., we show that using a small subset of the entire text corpus can be used to generate dictionaries that have high coverage on the entire text corpus.

The rest of this paper is structured as follows. In Sect. 2, we give an overview of related work. In Sect. 3, we present our interactive dictionary expansion approach, followed by an evaluation in Sect. 4. We conclude with a summary and an outlook on future work.

2 Related Work

Dictionaries, ontologies and linguistic resources are the backbone of many NLP and information retrieval systems. The automatic construction of such resources has been the focus of research for many years. One of the first approaches to extract dictionaries from unstructured text is the one proposed by Riloff and Jones [17]: starting from a few seed terms it learns extraction patterns, which are then used to expand the seeds and iteratively repeat the process. Many similar approaches sprouted afterward [2,4,11,18]. Their main drawback is that they all require NLP parsing for feature extraction, and thus have a reliance on syntactic information for identifying quality patterns. Hence, such approaches underperform on less polished or grammatically incorrect text content, like user-generated text. Furthermore, completely automatic iterative methods can easily generate semantic drift - a few spurious extraction patterns can exponentially increase the inclusion of incorrect items in the dictionary.

A similar - and sometimes considered as preliminary - task is terminology extraction from text. Different from the dictionary extraction, *terminology extraction* aims at extracting the vocabulary that is specific of a certain domain, but extracted terms will belong to the several different concepts which are related to the domain. There are numerous solutions available that perform linguistic processing on a certain corpus and use various statistical techniques to understand which terms represent its domain-specific terminology. A comprehensive description (and implementation) of available techniques is available in [20,21]. In this work we compare to techniques based on term occurrence frequency [5]. A widely used example is the classic document-specific TFIDF (term frequency, inverse document frequency) [6] which has been adapted for this task [20]. The main reason for comparing with these methods is that they are (i) purely based on statistics and (ii) do not rely on external reference corpora apart from the input text. While terminology extraction can rightfully be considered as a preliminary step for dictionary extraction, we argue (and show with experiments) that performing initial candidate selection via terminology extraction can introduce unnecessary errors and limit the potential recall of the extraction, as opposed to performing the dictionary extraction directly on the raw corpus without any pre-processing. Removing the need of linguistic preprocessing can prove to be extremely beneficial when dealing with user-generated content, which is a valuable source of information for many domains. Even on technical domains such as pharmacovigilance, using a random Twitter stream as unlabeled training data has been proved successful for the recognition Adverse Drug Reaction [13].

Another challenge in the automatic creation of dictionaries is the fact that they can be highly dependent on the task at hand - one striking example is the creation of dictionaries of positive/negative words, which is highly influenced by the specific domain [9,16]. Many works tackle this hurdle by relying on machine learning techniques and tailor the algorithms to certain specific domains: these methods are in general expensive, requiring an annotated corpus and/or domain specific feature extraction (a comprehensive overview can be found in [15]). While it is clearly appealing to have completely automatic techniques, it comes at the expense of the need for annotations and fine-tuning for every new task/domain. We propose to mitigate the problem with a human-in-the-loop approach, where the "tuning" is an integral part of the process, i.e. the human works in partnership with the statistical method to drive the semantic of the task effectively and efficaciously.

Our work is closely related to *glimpse* [8], a statistical algorithm for dictionary extraction. The input is a large text corpus and a set of seed examples. Starting from these it evaluates the *contexts* (the set of words surrounding an item) in which the seeds occur and identifies "good" contexts. Contexts are scored retrospectively in terms of how many "good" results they generate. All contexts are kept which have a score over a given threshold and the candidates that appear in the most "good" contexts are provided first to the human-in-the-loop for adjudication. The approach has also been extended to *glimpseLD* [1], which is language agnostic and uses Linked Data to as a bootstrapping source. While both approaches have been proven to achieve high effectiveness for dictionary extension, both of the approaches can only identify new items that are present in the input text corpus. In this work, we adopt the *glimpse* computer/human partnership architecture and extend it with the *explore/exploit* algorithm for more effective dictionary expansion.

3 Approach

The input of the algorithm is a text corpus TC and a set of seed terms for the dictionary S. In the preprocessing step, we build a neural language model on the input text corpus TC. Our approach is based on the *explore and exploit* (*EnE*) paradigm to effectively discover new instances (explore) from the text corpus and generate new *"unseen"* instances based on human feedback (exploit). The approach is iterative: each iteration first runs the *explore* phase then the *exploit* one. After each iteration the subject matter expert performs adjudication on the proposed candidates. The *explore* phase uses all the terms currently in the dictionary to identify similar terms in the corpus (using the word2vec model), which are then accepted or rejected by the human-in-the-loop. All accepted terms are added to the dictionary and used in the *exploit* phase as well as all subsequent *explore* iterations. The *exploit* phase uses all terms in the dictionary and the neural language model to construct more complex terms that might be of interest for the user. The process stops once there are no more candidates to be discovered, or the user is satisfied with the results. The system architecture is shown in Fig. 1.

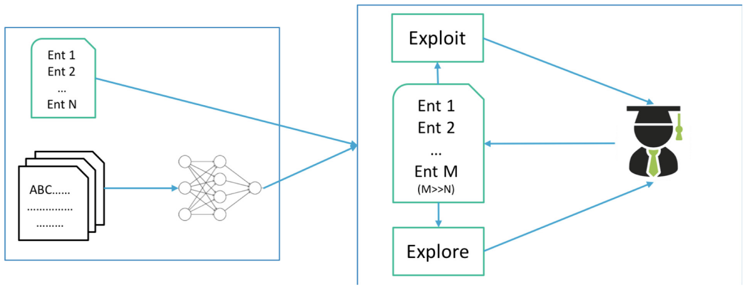


Fig. 1. EnE system architecture.

3.1 Explore

The *explore* phase uses all the terms currently in the dictionary to identify similar terms in the corpus (using the word2vec model). Word2vec is a computationally efficient two-layer neural network model for learning term embeddings from raw text. The output of the model is an embedding matrix W, where each term (single or multi token) from the corpus vocabulary V_{TC} is represented as an n-dimensional vector. When projecting this latent representations of words into a lower dimensional feature space, words which are semantically similar appear closer to each other. Therefore, the problem of calculating the similarity between two terms is a matter of calculating the distance between two instances in the given feature space. To do so we use the standard cosine similarity measure which is applied on the vectors of the instances. Formally, the similarity between two terms w_1 and w_2, with vectors V_1 and V_2, is calculated as the cosine similarity between the vectors V_1 and V_2:

$$sim(w_1, w_2) = \frac{V_1 \cdot V_2}{||V_1|| \cdot ||V_2||} \tag{1}$$

We calculate the similarity between the instances in the input dictionary and all the words in the corpus vocabulary V_{TC}. We sort the vocabulary in descending order using the cumulative similarity score, and choose the top-N candidates to present to the user. The accepted candidates are added in the input dictionary, which are then used in the *exploit* phase and the next iteration.

3.2 Exploit

In the *exploit* phase we try to identify more complex terms that don't appear in the corpus vocabulary, by analyzing the structure of the instances in the input dictionary. This is critical to help "future proof" a lexicon against new text. For a surveillance application (e.g., drug side effects mentioned on Twitter) it reduces how frequently a human needs to "tune up" the lexicon to make sure it is catching all relevant entity instances. We use two neural language models in this phase, i.e. word2vec [14] and bidirectional Long Short Term Memory (BiLSTM) models [19].

Exploit word2vec: Using the word2vec neural language model, we develop two phrase generation algorithms.

In the first approach, we first break each term in to a set of single tokens $T = \{t_1, t_2, \ldots, t_n\}$, then for each token t_i in T we identify a set of similar tokens $TS_{t_i} = \{ts_1, ts_2, \ldots, ts_s\}$ in the vocabulary V_{TC} using Eq. 1. In the next step, we build new terms by replacing t_i with a token ts_i from TS_{t_i}. The new terms are sorted based on the similarity score and the top-N are selected as candidates. For example, given the entry "abnormal behavior" the approach will identify "strange behavior", "abnormal attitude" and "strange attitude".

In the second approach, we generate new phrases by extending the instances with tokens from the text corpus that are related to the tokens in the instance. Related tokens are tokens that often share the same context, which means they often are surrounded by similar words. Given a word2vec model, we calculate the relatedness between two terms w_1 and w_2, as the probability $p(w_1|w_2)$ calculated using the softmax function,

$$p(w_1|w_2) = \frac{exp(v'^{T}_{w_1} v_{w_2})}{\sum_{w=1}^{V} exp(v'^{T}_{w} v_{w_2})}, \tag{2}$$

where v_w and v'_w are the input and the output vector of the word w, and V is the complete vocabulary of words.

As before, we first break each term in to a set of single tokens $T = \{t_1, t_2, \ldots, t_n\}$, then for each token t_i in T we identify a set of similar tokens $TR_{t_i} = \{tr_1, tr_2, \ldots, tr_r\}$ in the vocabulary V_{TC} using Eq. 2. In the next step, we build new multi-token term by appending a token tr_i from TR_{t_i} to each token t_i from T. The new phrases are sorted based on the relatedness score and the top-N are selected as candidates. For example, given the instance "clotting problems" in the input dictionary the approach first tries to identify related tokens in the text corpus for "clotting". For which the top word is "blood", because in many sentences "blood clotting" appears as a phrase, which can be used to generate new instances "blood clotting problems". In the next iteration the phrase can be further extended, by identifying new related words. For example, in the top-N related words for "blood" we will find "abnormal", which can be used to generate the instance "abnormal blood clotting problems".

Exploit BiLSTM: LSTM neural networks are special type of recurrent neural networks (RNN) [3] used in many NLP application. RNNs are able to model sequential data with temporal dependencies, like text. RNNs perform the same task for every element in a sequence, e.g., word in a sentence, conditioning the output of the model on the previous computations. One limitation of the basic RNN models is learning time-dependencies more than a few time-steps long [10]. LSTM networks solve this issue and are capable of learning long-term dependencies. Bidirectional LSTMs are an extension of the traditional LSTMs, which are able to capture the backwards dependences in the sequential data. Instead of training one LSTM model the BiLSTMs train two models, one for the data in the original direction, and one for the reversed copy of the input sequence. As the BiLSTM networks perform well on analyzing sequences of values and predicting the next one, they have been successfully applied for the task of text generation, i.e., given a sequence of words, the model can successfully predict the next word in the sequence. We use this capability for generating new multi-token dictionary candidates that are similar to the input seeds. To build such multi-token terms (phrases), we make use of the LSTM models in both directions, i.e., expanding existing terms in both forward and backward directions.

For the forward expansion, for each instance $T = \{t_1, t_2, \ldots, t_n\}$ in the input dictionary, we generate forward sub-sequences with size in the range $[1, n-1]$ starting from the first token t_1 till token t_{n-1}, resulting in a set of sequences $FS = \{t_1, t_1{-}t_2, \ldots, t_1{-}t_2{-}\cdots{-}t_{n-1}, t_2{-}t_3, \ldots, t_{n-1}\}$. Each of these sequences is then fed in the forward LSTM network to calculate the probability for each word in the corpus vocabulary V_{TC} to be the next word in the input sequence using the softmax function. For each input instance we select the top-100 generated new phrases, then we aggregate across all input instances, and select the top-N to present to the user.

For the backward expansion, for each instance $T = \{t_1, t_2, \ldots, t_n\}$ in the input dictionary, we generate backward sub-sequences with size in the range $[1, n]$ starting from the last token t_n to the first token t_1, resulting in a set of sequences $BS = \{t_n, t_n - t_{n-1}, \ldots, t_n - t_{n-1} - \cdots - t_1, t_{n-1} - t_{n-2}, \ldots, t_1\}$. Each of these sequences is then fed in the backward LSTM network to calculate the probability for each word in the corpus vocabulary V_{TC} to be the next word in the backward input sequence using the softmax function. For each input instance we select the top-100 generated new phrases, then we aggregate across all input instances, and select the top-N to present to the human-in-the-loop, which are reversed in the original order. In both modes, in cases where the phrases end with a stop-word (e.g. “a” and “the”) we do a second forward pass in the network for the current sequence. Furthermore, rejected phrases that have high confidence value are copied in the next iteration, as many phrases might need multiple iterations to form a complete phrase.

For example, given the input instance “high blood pressure”, the set of forward sequences will be $FS\{high, high\ blood, blood\}$ and the backwards sequences will be $BS = \{pressure, pressure\ blood, pressure\ blood\ high, blood, blood\ high, high\}$. Then each of these instances is fed in the forward and backward model, respectively.

For the forward sequence “high blood”, the model proposes phrases like “high blood count”, “high blood cell”. While the phrase “high blood count” is a valid phrase, the phrase “high blood cell” although it has high confidence value it is not complete and cannot be accepted as valid. However, we can foresee that in the next iteration this phrase might result in a valid phrase with high confidence value. Therefore, in each iteration we keep a set of rejected phrases that have high confidence value, and we extend them once more in the next iteration. For example, the phrase “high blood cell” will be extended to a valid phrase “high blood cell count”.

For the backward sequence “pressure blood”, the backward model proposes phrases like “pressure blood low” and “pressure blood varying”, which are then reversed to form human readable phrases, i.e., “low blood pressure” and “varying blood pressure”.

4 Evaluation

To evaluate our approach we conduct three experiments, i.e., (i) count the number of newly discovered dictionary entries per iteration; (ii) the impact of the promptness of the human-in-the-loop on the system performance; (iii) capability of the system for identifying dictionary instances that are not present in the analyzed corpus, but might appear in future texts.

4.1 Dictionary Growth

We consider 3 different text corpora:

– *ADE*: A dataset from the healthcare domain, specifically tackling the problem of identifying Adverse Drug Events (ADE) in user generated data. The input text corpus consists of user blogs extracted from http://www.askapatient.com[1], containing a total of 1, 384, 716 sentences. For this corpus we build a dictionary of ADE and the adjudication is performed by a medical doctor.
– *Twitter Food*: A dataset consisting of 4, 203, 922 tweets, sampled from Twitter,[2] in the period of January 2017, filtered using an existing vocabulary of food items from [7]. For this corpus we build a dictionary of food items.
– *IBM Call Center*: A dataset from the IBM call center consisting of customer’s surveys, containing a total of 6, 004, 804 sentences. For this corpus we build 3 dictionaries containing terms expressing customers’ satisfaction (*Satisfied*), terms expressing customers’ being confused by the process (*Confusing*) and terms expressing customers’ complaints about the phone connection (*Bad Connection*). The adjudication is performed by a team of subject matter experts that have many years experience working with costumer satisfaction data and maintaining call center systems.

[1] A forum where patients report their experience with medication drugs.
[2] https://twitter.com/.

Table 1. Accuracy of newly discovered dictionary instances per iteration and the average length, using W2V, EnE - W2V and EnE-BiLSTM approaches on 5 dictionaries.

Dataset	#sentences	#iter	W2V		EnE-W2V		EnE-BiLSTM	
			Acc.	Avg. Len.	Acc.	Avg. Length	Acc.	Avg. Len.
ADE	1,384,716	50	60.2	1.29	**80.3**	2.14	79.0	**2.66**
Twitter Food	4,203,922	20	67.5	1.74	**79.5**	2.34	58.3	**2.46**
Satisfied	6,004,804	20	61.3	2.39	**87.3**	3.26	75.0	**4.15**
Confusing	6,004,804	20	42.8	1.87	68.5	3.43	**71.5**	**3.79**
Bad Connection	6,004,804	10	66.0	2.26	**85.0**	2.85	81.5	**3.37**

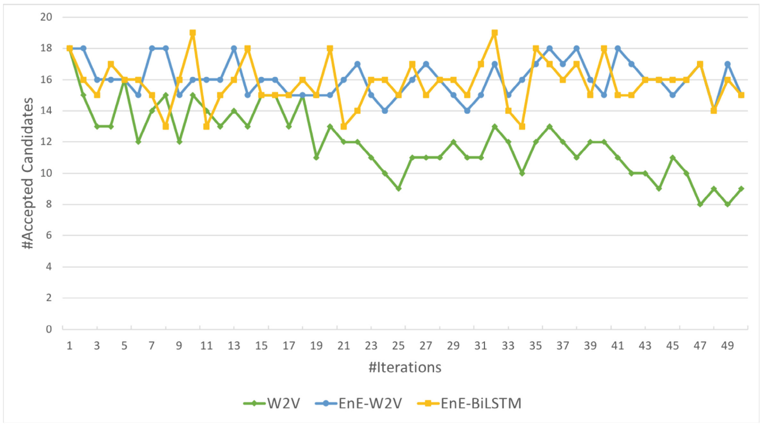
We use the EnE approach to build the 5 dictionaries, using word2vec in the explore phase and word2vec (EnE-W2V) or BiLSTM (EnE-BiLSTM) in the explore phase. We use a standard word2vec model as a baseline, i.e., running only the explore phase of the EnE approach (W2V). For the baseline *W2V* approach in pre-processing we identify terms of maximum 3 tokens, using the phrase detection algorithm based on words co-occurrence proposed in [14]. We use the skip-gram algorithm with window size 10, 10 training iterations and 200 dimensional vectors.

The BiLSTM network has 4 stacked 256-unit LSTM layers, each followed by dropout layer with probability of 0.2 where the last layer is a fully connected Softmax layer. The models were implemented in TensorFlow,[3] and trained on Tesla P100 16 GB GPU.

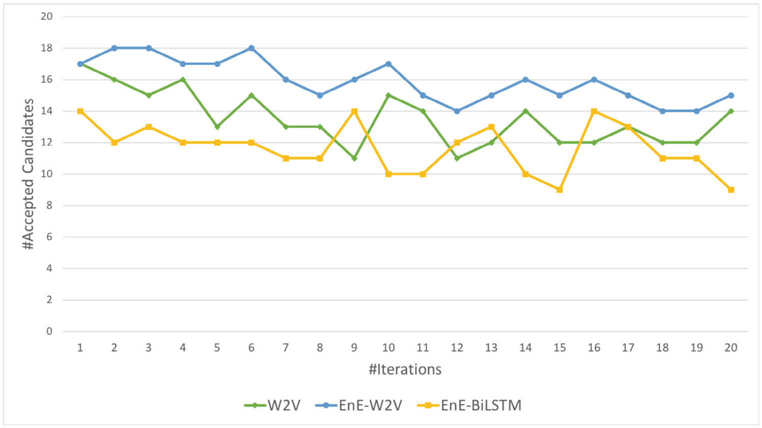
We run the evaluation in 50 iterations for the *ADE* dictionary, 20 iterations for the *Twitter Food*, *Satisfied* and *Confusing* dictionary, and 10 iterations for the *Bad Connection* dictionary. In each iteration the algorithm proposes 20 candidates. The number of iterations was decided by the subject matter expert that is performing the adjudication, i.e., once the dictionary has sufficient number of entries the dictionary expansion stops. After each iteration we count how many new instances are discovered in the top 20 proposed candidates by the algorithm. The accepted instances are then added in the dictionary and used for the next iteration. For the *EnE* approach we run *explore* to identify 10 candidates, and *exploit* to identify another 10 candidates, while for the baseline *W2V* approach we propose directly 20 candidates. The accuracy results per iteration for each of the datasets are shown in Fig. 2, i.e., the fraction of accepted candidates by the adjudicator. The accuracy of all the approaches after the end of the process are shown in Table 1, including the average length of the discovered dictionary entries.

The results show that using the *EnE-W2V* and *EnE-BiLSTM* approach we are able to discover a steady number of instances in each iteration, significantly outperforming the baseline *W2V* approach. The *EnE-W2V* approach leads to better performance on most of the dataset, except the "IBM Call Center - Confusing" dataset where the *EnE-BiLSTM* app-

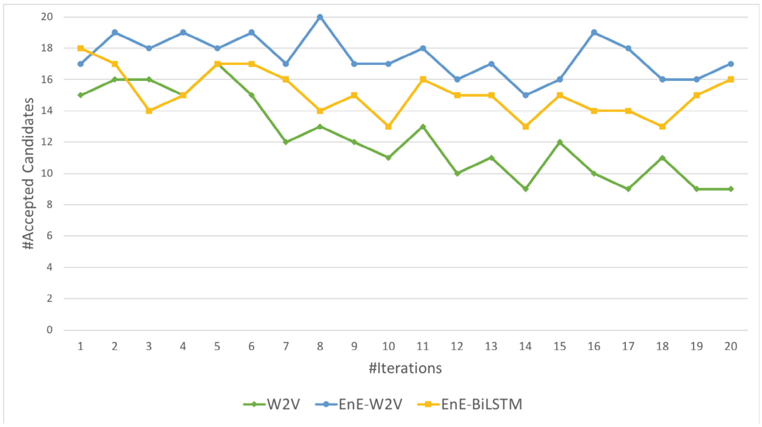
[3] https://www.tensorflow.org/.



a) *ADE*

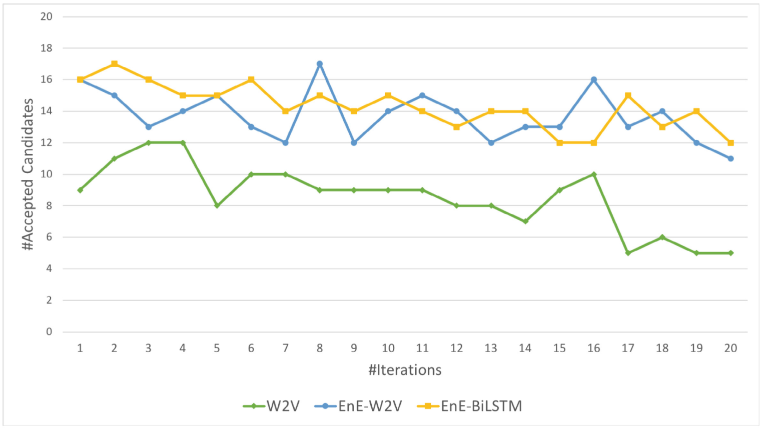


b) *Twitter Food*

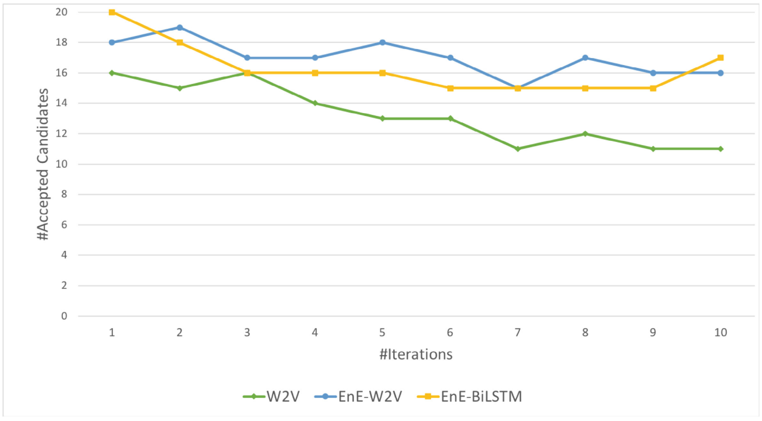


c) *IBM Call Center - Satisfied*

Fig. 2. Number of discovered new dictionary instances per iteration, using W2V, EnE - W2V and EnE-BiLSTM approaches on 5 dictionaries



d) *IBM Call Center- Confusing*



e) *IBM Call Center - Bad Connection*

Fig. 2. (*continued*)

roach delivers higher accuracy. We can observe that on the *Twitter Food* dataset the *EnE-BiLSTM* approach performs poorly. There are two reasons for the low performance, i.e., the tweets are very short for the LSTM model to learn strong contexts for the words; (ii) the tweets contain a lot of syntactic irregularities, misspellings and incorrect grammar. We can observe that when using the *W2V* approach the number of newly discovered instances quickly decreases as the number of available instances in the whole corpus is decreasing in each iteration.

Furthermore, we can see that the *W2V* is able to identify only uni-grams and short phrases, while the *EnE* approaches can identify significantly longer phrases on all the datasets.

For the sake of completeness we compare our dictionary generation approach with fully automated terminology extraction methods. The two techniques are not directly comparable for the reason that while dictionary extraction aims at extracting terms that are cohesive and belong to the same specific semantic concept, terminology extraction aims at extracting the general vocabulary of a certain domain, but extracted terms will belong to the several different concepts. Nonetheless, given a corpus, all of terms extracted via terminology extraction would potentially contain all the terms which are specific to the corpus domain, in one big set. Therefore, one meaningful comparison that we can perform is the following: given a (validated) dictionary constructed with our approach, check how many are present in the set extracted with terminology extraction software. This would indicate if it is useful to perform terminology extraction as preliminary step and then extract all the single dictionaries for the domain.

We use the JATE software [20], which implements all state-of-the-art techniques for terminology extraction. Given that we are checking the recall of these techniques, we report results for the most comprehensive one, the TFIDF adaptation from JATE, which is generating the biggest set of terms.

We performed terminology extraction with JATE on the 5 dictionaries, with a very inclusive threshold i.e. considering any term with a score above zero as valid. For the IBM Call Center corpus, JATE extracted $118,176$ terms. When considering the 3 dictionaries that we built with *EnE* for this corpus, only 17% from *Satisfied*, 10% from *Confusing* and 6% from *Bad Connection* are contained in the comprehensive JATE created terminology. Similarly for the *ADE* corpus, JATE extracted a set of 636,264 terms, and only 26% of the terms of the *ADE* dictionary built with *EnE* are present in this set. For the *Twiter food* corpus, JATE extracted a set of $1,164,329$ terms, and only 43% of the terms of the *food* dictionary built with *EnE* are present in this set.

This experiment shows that although a fully automated terminology extraction method is able to generate a large number of lexicon entries, the coverage of relevant terms for specific tasks is very low. Therefore, using automatic approaches is not suitable for dictionary extension, rather a human-in-the-loop is required.

4.2 Impact of the Human-in-the-loop on the Dictionary Growth

In this experiment we show the importance of the interaction of the human-in-the-loop on the number of newly discovered instances, i.e., we evaluate if the user gives their feedback to the system sooner it will improve the performance of the system. To do so, we run the *EnE-W2V* approach with different feedback intervals on the *ADE* dataset. The feedback interval indicates how many candidates the system needs to identify before the user gives their feedback to the system. For example, when using feedback interval of 10, the user gives their feedback after 10 candidates are identified by the system. We evaluate feedback intervals of 10, 50, 100, 250 and 500. After each iteration we count the number of accepted candidates, and include them in the dictionary to be used for the next iteration. The results are shown in Fig. 3.

The results show that the tighter the human-in-the-loop integration is, the more quickly new instances are discovered. We see that with a large 500 examples

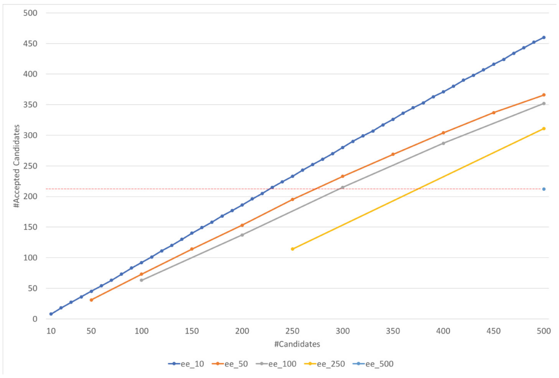


Fig. 3. Impact of the human-in-the-loop promptness on the number of newly discovered dictionary instances using EnE - W2V on the ADE dataset

feedback interval the human-in-the-loop system discovers 212 new instances, but requires the human to consider 500 candidates.

A more tightly integrated system with a 10 examples feedback interval finds 212 new instances in just 23 iterations, requiring the human to consider only 230 candidates. After 50 iterations the system discovered 460 new dictionary entries, compared to only 212 new entries when using 500 examples feedback interval. That yields 216% improvement in effectiveness of the system.

4.3 Generating "Future Proof" Dictionaries

In this experiment we evaluate if generating "unseen" instances to "future proof" a dictionary is useful. To do so, we take a small portion of the dataset to generate the vocabularies, and then we count how many of the newly generated instances that don't appear in the training set appear in the remaining text corpus that was not used for training. For the evaluation we use the 3 vocabularies from the *IBM Call Center* dataset. We randomly select 10% of the text corpus for training the neural language model and run both versions of the *EnE* algorithm to generate the vocabularies. Then we identify how many of the instances generated in the *Exploit* phase of the *EnE* approach don't occur in the training dataset, but occur in the remaining 90% of the text corpus.

The number of newly discovered entries that don't appear in the training set (#new entries), and the number of entries that appear in the future text (#future hits) on the 3 vocabularies are shown in Table 2. We can see that on the *Satisfied* dataset, both *EnE-W2V* and *EnE-BiLSTM* approaches identify a significant number of new entries that don't appear in the training dataset, but more than 50% of these entries appear in the remaining text corpus. The number of "future hits" slightly reduces on the other 2 vocabularies, but it is still significant.

These experiments prove that generating "future proof" dictionaries is useful, and it will certainly reduce how frequently a human needs to "tune up" the dictionaries in order to keep high coverage.

Table 2. Number of dictionary entries that don't occur in the training text corpus, but appear in a future text corpus. Tested on the IBM call center vocabularies.

Dataset	#iterations	EnE-W2V		EnE-BiLSTM	
		#new entries	#future hits	#new entries	#future hits
Satisfied	20	94	55 (58.51%)	83	44 (50.00%)
Confusing	20	79	32 (40.50%)	76	33 (43.42%)
Bad Connection	10	34	14 (41.17%)	52	19 (36.53%)

5 Conclusions

This paper proposes an interactive dictionary expansion tool using two neural language models. Our algorithm is iterative and purely statistical, hence does not require any feature extraction beyond tokenization. It incorporates human feedback to improve performance and control semantic drift at every iteration cycle. It does not require phrase detection prior building the model, but rather uses the neural language models to identify phrases with high confidence that might not appear in the input text corpus. This is critical to help "future proof" the dictionary to capture new terms in previously unseen text, thus reducing the how frequently the subject matter expert needs to "tune up" the dictionary. The evaluation shows that our approach is able to rapidly extend the input dictionaries with high accuracy. Furthermore, the experiments show the critical value of tight human-in-the-loop integration, which leads to higher efficiency and effectiveness for the task of dictionary expansion.

References

1. Alba, A., Coden, A., Gentile, A.L., Gruhl, D., Ristoski, P., Welch, S.: Multi-lingual concept extraction with linked data and human-in-the-loop. In: Proceedings of the Knowledge Capture Conference, p. 24. ACM (2017)
2. Ando, R.K.: Semantic lexicon construction: learning from unlabeled data via spectral analysis. Technical report, IBM Thomas J Watson Research Center, Yorktown Heights, NY (2004)
3. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**(2), 157–166 (1994)
4. Blohm, S., Cimiano, P.: Using the web to reduce data sparseness in pattern-based information extraction. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 18–29. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74976-9_6

5. Bourigault, D.: Surface grammatical analysis for the extraction of terminological noun phrases. In: Proceedings of the 14th Conference on Computational Linguistics - Volume 3, Stroudsburg, PA, USA, pp. 977–981 (1992)
6. Church, K., Gale, W.: Inverse document frequency (IDF): a measure of deviations from poisson. In: Armstrong, S., Church, K., Isabelle, P., Manzi, S., Tzoukermann, E., Yarowsky, D. (eds.) Natural Language Processing Using Very Large Corpora. Text, Speech and Language Technology, vol. 11, pp. 283–295. Springer, Dordrecht (1999). https://doi.org/10.1007/978-94-017-2390-9_18
7. Coden, A., Danilevsky, M., Gruhl, D., Kato, L., Nagarajan, M.: A method to accelerate human in the loop clustering. In: Proceedings of the 2017 SIAM International Conference on Data Mining, pp. 237–245. SIAM (2017)
8. Coden, A., Gruhl, D., Lewis, N., Tanenblatt, M., Terdiman, J.: SPOT the drug! An unsupervised pattern matching method to extract drug names from very large clinical corpora. In: Proceedings of the 2012 IEEE 2nd Conference on Healthcare Informatics, Imaging and Systems Biology, HISB 2012, pp. 33–39 (2012)
9. Hamilton, W.L., Clark, K., Leskovec, J., Jurafsky, D.: Inducing domain-specific sentiment lexicons from unlabeled corpora. In: Conference on Empirical Methods in Natural Language Processing, Austin, Texas, pp. 595–605 (2016)
10. Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al.: Gradient flow in recurrent nets: the difficulty of learning long-term dependencies (2001)
11. Igo, S.P., Riloff, E.: Corpus-based semantic lexicon induction with web-based corroboration. In: Proceedings of the Workshop on Unsupervised and Minimally Supervised Learning of Lexical Semantics, pp. 18–26. ACL (2009)
12. Kuriki, I., et al.: The modern Japanese color lexicon. J. Vis. **17**, 1 (2017)
13. Lee, K., et al.: Adverse drug event detection in tweets with semi-supervised convolutional neural networks (2017)
14. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
15. Pazienza, M.T., Pennacchiotti, M., Zanzotto, F.M.: Terminology extraction: an analysis of linguistic and statistical approaches. In: Sirmakessis, S. (ed.) Knowledge Mining. STUDFUZZ, vol. 185, pp. 255–279. Springer, Heidelberg (2005). https://doi.org/10.1007/3-540-32394-5_20
16. Pröllochs, N., Feuerriegel, S., Neumann, D.: Generating domain-specific dictionaries using Bayesian learning. In: ECIS 2015, pp. 0–14 (2015)
17. Riloff, E., Jones, R., et al.: Learning dictionaries for information extraction by multi-level bootstrapping. In: AAAI/IAAI, pp. 474–479 (1999)
18. Riloff, E., Wiebe, J., Wilson, T.: Learning subjective nouns using extraction pattern bootstrapping. In: HLT-NAACL 2003, pp. 25–32 (2003)
19. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Process. **45**(11), 2673–2681 (1997)
20. Zhang, Z., Gao, J., Ciravegna, F.: JATE 2.0: Java automatic term extraction with apache solr. In: LREC 2016, Portorož, Slovenia (2016)
21. Zhang, Z., Gao, J., Ciravegna, F.: SemRe-Rank: Improving automatic term extraction by incorporating semantic relatedness with personalised pagerank. TKDD **12**(5), 57:1–57:41 (2018)