Latent Entities Extraction: How to Extract Entities that Do Not Appear in the Text?

Eylon Shoshan[1], **Kira Radinsky**[1,2]
{eylonsho999,kirar}@cs.technion.ac.il

[1]Technion - Israel Institute of Technology, Haifa, Israel
[2]eBay Research, Israel

Abstract

Named-entity Recognition (NER) is an important task in the NLP field , and is widely used to solve many challenges. However, in many scenarios, not all of the entities are explicitly mentioned in the text. Sometimes they could be inferred from the context or from other indicative words. Consider the following sentence: *"CMA can easily hydrolyze into free acetic acid."* Although water is not mentioned explicitly, one can infer that H2O is an entity involved in the process. In this work, we present the problem of *Latent Entities Extraction* (LEE). We present several methods for determining whether entities are discussed in a text, even though, potentially, they are not explicitly written. Specifically, we design a neural model that handles extraction of multiple entities jointly. We show that our model, along with multi-task learning approach and a novel task grouping algorithm, reaches high performance in identifying latent entities. Our experiments are conducted on a large biological dataset from the biochemical field. The dataset contains text descriptions of biological processes, and for each process, all of the involved entities in the process are labeled, including implicitly mentioned ones. We believe LEE is a task that will significantly improve many NER and subsequent applications and improve text understanding and inference.

1 Introduction

Named entity recognition (NER) is an important building block in many natural-language-processing algorithms and applications. For example, representing texts as a knowledge graph, where nodes are extracted entities, has been proved to be effective for question answering (Berant and Clark, 2014) as well as for summarization tasks (Ganesan et al., 2010). Other applications, such as semantic annotation (Marrero et al., 2013) require recognition of entities in the text as well. Babych and Hartley (2003) have also shown that identifying named entities correctly, has an effect both on the global syntactic and lexical structure, additionally to the local and immediate context.

NER today focuses on extracting existing entities in the text. However, many texts, contain "hidden" entities, which are not mentioned explicitly in the text, but might be inferred from the context. For example, special verbs could help a human reader infer the discussed entity implicitly. Consider the following textual passage of a biochemical reaction:

> *"At the plasma membrane, phosphatidylcholine is **hydrolyzed**, removing one of its acyl groups, to 1-acyl lysophosphatidylcholine by membrane-associated phospholipase b1. "*

The words *water* or *H2O* are not mentioned. Nonetheless, one could easily infer that water is involved in the process, since the word hydrolyzed refers to water. Therefore, water is a latent entity in this case. Other contexts, do not involve only indicating verbs. Consider the following sentence:

> *"The conversion of Testosterone to Estradiol is catalyzed by Aromatase associated with the endoplasmic reticulum membrane."*

Here, *Oxygen* is a latent entity. Aromatase is an enzyme that belongs to the Monooxygenases family. This family is characterized by requiring Oxygen when performing catalyzation.

Latent entities not only play a prominent role in biochemical and medical fields, but are also common in other domains. For example, consider the following snippet as published in business section, New York Times magazine in January 2017:

> *"The free app, which Facebook owns, is offering another vehicle to advertisers,*

who since late 2015 have been buying space on its original photo feed."

To an average human reader who is familiar with contemporary norms and trends, it is quite clear that *Instagram* app is discussed in the textual passage above. However, it is not explicitly written, thus it is practically a latent entity.

Identifying latent entities in texts, and gaining the ability to infer them from a context, will significantly enrich our ability to comprehend and perform inference over complex texts.

In this work, we formulate the novel problem of Latent-Entities Extraction (LEE). We study several deep and non-deep models for this task, that learn to extract latent entities from texts and overcome the fact that these are not mentioned explicitly. Specifically, we study a model that combines a neural recurrent network (Bi-GRUs) and multi-task learning, showing that joint prediction of correlated entities could refine the performance. We present a novel algorithm for task grouping in the multi-task learning setting for LEE. The algorithm chooses which latent entities to learn together. We show this approach reaches the best performance for LEE.

The contribution of our works is threefold: (1) We formulate a novel task of LEE, where the goal is to extract entities which are implicitly mentioned in the text. (2) We present a large labeled biological dataset to study LEE. (3) We present several algorithms for this task. Specifically, we find that learning multiple latent entities in a multi-task learning setting, while selecting the correct entities to learn together, reaches the best results for LEE. We share our code and data with the community to enable the community to develop additional algorithms for LEE: `https://github.com/EylonSho/LatentEntitiesExtraction`

2 Related Work

Entities Recognition Named-entity recognition (NER) aims at identifying different types of entities, such as people names, companies, locations, organizations, etc. within a given text. Such deduced information is necessary for many application, e.g. summarization tasks (Ganesan et al., 2010), data mining (Chen et al., 2004), and translation (Babych and Hartley, 2003).

This problem has been widely researched. Several benchmark data sets such as CoNLL-2003 (Tjong Kim Sang and De Meulder, 2003) and OntoNotes 5.0 (Hovy et al., 2006; Pradhan et al., 2013) were published. Traditional approaches label each token in texts as part of named-entity, and achieve high performance (Ratinov and Roth, 2009; Passos et al., 2014; Chiu and Nichols, 2016).

However, these approaches are relying on the assumption that entities are necessarily mentioned in the text. To best of our knowledge, the problem of latent entities extraction, where entities could potentially not be mentioned in the text at all, is yet to be researched.

Multi-Task Learning Multitask learning (Caruana, 1998) was extensively used across many NLP fields, including neural speech translation (Anastasopoulos and Chiang, 2018), neural machine translation (Domhan and Hieber, 2017), and summarization tasks (Isonuma et al., 2017). In this work we study several approaches for LEE, including multi-task learning. We observe that the vanilla approach of multi-task learning is reaching limited results in our setting (Section 6). Previous work (Liu and Pan, 2017; Zhong et al., 2016; Jeong and Jun, 2018) have suggested that multi-task learning should be applied on related tasks. We present an extension to the multi-task learning setting by performing clustering to related tasks to improve performance.

3 The Reactome Dataset

It is quite common to have implicit entities in texts in the biomedical field. *Reactome* (Croft et al.) is a large publicly-available biological dataset of human biological pathways and reactions. The data consists of 9,333 biochemical reaction diagrams and their textual description. Each reaction is labeled by experts regarding its reactants and products. We consider each reactant or product of a reaction as an entity. If an entity is not mentioned in the textual description, it will be considered as a *latent entity*. In more than 90% of the reactions, there are 3–5 involved entities. We have performed an exploration to find latent frequency, i.e., how many times the entity was found as latent, among all of its occurrences in the dataset. We identify that 97.53% of the texts contain at least one latent entity and that 80.65% of the entities are latent at least 10% of the times. The analysis results for several entities are shown in Table 1. We observe an interesting phenomena – several entities, such as ATP, mostly appear as a latent en-

Entity	Times Not Mentioned	Total Occurrences	Latent Frequency (%)
ATP	1177	1448	81.28
ADP	1221	1326	92.08
H2O	949	1087	87.30
PI	249	492	50.61
H+	296	487	60.78
O2	159	275	57.82
NADPH	145	254	57.09
NADP+	175	253	69.17
COA-SH	123	191	64.40
PPI	105	190	55.26
ADOMET	120	181	66.30
GTP	40	181	22.10
GDP	60	179	33.52
ADOHCY	155	169	91.72
UB	0	157	0.00
CO2	75	145	51.72
NAD+	56	134	41.79
AMP	53	128	41.41
NADH	46	112	41.07
NA+	23	94	24.47
2OG	33	93	35.48
L-GLU	66	82	80.49
AC-COA	43	75	57.33

Table 1: Latent frequency of top common entities

tity in the descriptions, i.e., most of the times they are not mentioned explicitly in the text.

4 One-vs-all Algorithms for LEE

Given a single entity which frequently tends to be latent, we need to classify whether it is involved within a given text. We train a classifier per entity using multiple techniques. We then apply the classifier on each text passage that may discuss several latent entities, and output their prediction in a one-vs-all approach.

We present several models which predict whether a given entity is implicitly (or explicitly) involved in a textual paragraph. We construct a classifier per entity which detects the entity in texts, and overcomes the cases where it is latent. We devise a few simple yet relatively powerful algorithms presented in Sections 4.1 – 4.5.

4.1 Bag-of-words (TF-IDF)

To tackle the LEE problem, we try to leverage the context to infer latent entities. We transform a text to a TF-IDF vector representation (applied on bigrams). Using these vectors we train several supervised classification models. We did not observe a significant difference between the models, and present results for Support Vector Machine (SVM) model (Cortes and Vapnik, 1995) that have shown the highest performance on a validation set. The models are trained to predict whether a given entity is involved or not. As can be observed in Table 1, most of the entities are latent enough, thus this data set is appropriate to the LEE task.

4.2 Weighted Document Embedding

One of the state-of-the-art approaches for modeling text was presenting by Arora et al. (2017). We leverage pre-trained word embedding vectors (Chiu et al., 2016) to generate an embedding for a text which might contain implicit entities. Based on these embeddings, a supervised classifier per entity is trained as before, i.e., we create a classifier per entity to predict whether it is implicitly mentioned in the text.

4.3 Element-Wise Document Embedding

We study several additional methods of representing a document using several word embedding compositions (De Boom et al., 2016). We leverage pre-trained word embedding vectors, that were trained on Pubmed data (Chiu et al., 2016), and suggest the following composition techniques: (1) We compute the element-wise maximum vector of each word from the text, denoted as v_{max}; (2) We compute the element-wise minimum vector of word embedding, denoted as v_{min}. (3) We compute the element-wise mean vector, denoted as v_{avg}.

We concatenate these three vectors into the final document representation: $v = [v_{max}; v_{min}; v_{avg}]$. This is the feature vector which is fed as an input to the SVM classifier, built for each entity separately.

4.4 Combined Document Embedding

In this approach, we attempt to combine several ways of representing a document into a single representation. We concatenate the feature vectors for each document as generated in sections 4.2, 4.3. Classification model is then trained similarly to the previous sections and applied on the new representation.

4.5 Deep Document Embedding

Instead of disregarding word order as in the previous approaches (Sections 4.2– 4.4), we leverage pre-trained word embedding vectors that were trained on Pubmed data (Chiu et al., 2016), and learn an unsupervised deep model to produce a document embedding. We experiment

with several deep models, including Bi-LSTM and Bi-GRU unit: each textual description is translated to sequence of pre-trained embeddings. That sequence is fed into a Bi-Directional Long Short Term Memory (Bi-LSTM) (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997) or Bi-GRU (Cho et al., 2014), and based on the final cell state, we perform a binary prediction whether the given entity is implicitly mentioned or not.

5 Multi-Task-Learning Algorithms for LEE

Given a predefined list of entities, we wish to classify whether *one entity or more* from that list, are involved in a given text passage. We train a single multi-task-learning classifier that outputs the set of latent entities relevant to the text. Intuitively, the model might capture correlation of entities that tend to be involved (or not) together, and therefore their latent behavior might be similar. For each entity which is listed in a predefined list, the model will output a probability as an estimation for its likelihood to be involved in a given text.

5.1 Multi-Task-Learning Model Architecture

Figure 1 illustrates the general design of our architecture: an embedding layer, a Bi-GRU components that are fed by the embedding, and ultimately a prediction layer containing as many outputs as the total number of latent entities to be extracted.

Embedding The embedding layer first embeds a sequence of words into a sequence of embedding vectors of 200 dimension.

Bidirectional GRU The output vectors from the last layer are fed into a RNN unit to capture context out of the text. This unit is capable of analyzing context that is spanned over sub-sequences in texts. This is done when the RNN component is sequentially fed by the embedding vectors $\{v_t\}$, and iteratively compute a hidden state vector $\{h_t\}$ based on the previous hidden state and the current input embedding vector, using some function f. Moreover, the output of this unit $\{o_t\}$ in each timestamp t, is computed based on the current hidden state using a function g.

Specifically we use a GRU unit as a RNN as presented by Cho et al. (2014). Hidden state's dimension is set to 200, with sigmoid as an activation function. Additionally, we use the bidirectional version (Schuster and Paliwal, 1997) of GRU.

We also apply natural dropout (Srivastava et al., 2014) of 0.5 on the input embedding vectors. Another refinement is dropout that is applied on the recurrent neural network hidden layers, as Gal and Ghahramani (2016) have suggested. This recurrent dropout is set to 0.25.

Classifier The outputs of the Bi-GRU unit, of the first and last cell, are considered during classification phase. The classifier unit is a fully connected layer with a sigmoid activation layer with k outputs, where k is the number of all tasks, i.e., entities being predicted.

Loss Function We define a loss function to address the multi-task learning approach. Currently, we present a loss function for multi-task prediction that joins all of the entities together into a single prediction unit. Denote m as the number of training samples, and k as the number of latent entities that are intended to be extracted. We define the following loss function:

$$L(y, \hat{y}) = \\ -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k}\left(y_j^{(i)}\log\hat{y}_j^{(i)} + \left(1-y_j^{(i)}\right)\log\left(1-\hat{y}_j^{(i)}\right)\right)$$

where y and $\hat{y}$ are the labeled and predicted values, respectively. Practically, we aggregate the log-losses over all of the training samples and latent entities, and then averaging to get the final loss.

Note that we address all of the entities as they were related in here, since the loss is calculated based on them all with no exceptions.

Training Model optimization was carried out using standard backpropagation and an Adam optimizer (Kingma and Ba, 2014). We have trained our model with 300 epochs and a batch size of 128. Backpropagation is allowed through all layers, except the embedding layer, which is set using pre-trained embeddings.

Word Embedding Initialization We use pre-trained word embedding to represent each text passage. Note that fine-tuning is also not practical from similar reasons, hence we directly use *word2vec* trained vectors[1]. These were trained

[1]Trained embedding is available online at: https://github.com/cambridgeltl/BioNLP-2016

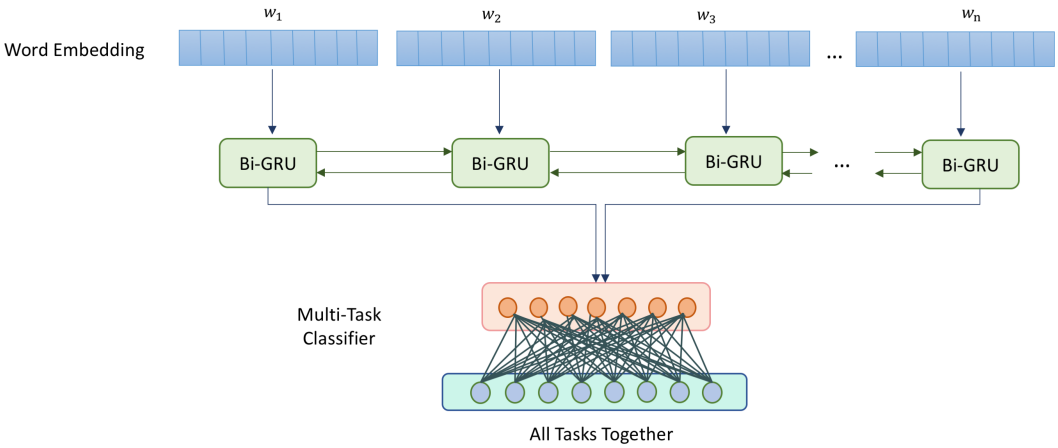

Figure 1: The multi-task model architecture for latent entities extraction. Word embeddings are fed to several Bi-GRU units which are connected via a multi-task learning approach to numerous outputs, each representing a different latent entity prediction

over large corpora, the PubMed archive of the biochemical, biological and medical field by Chiu et al. (2016). We fit this choice to the nature of our data set, Reactome, which is consisted of biochemical reactions and biological processes.

5.2 Task-Grouping Model Architecture

The common approach in multi-task learning is handle all tasks altogether (Evgeniou and Pontil, 2004; Rai and Daume III, 2010). Therefore, a possible approach could possibly suggest that all of the entities should be predicted together as a single multi-task classification process. However, this method is based on the assumption that all entities are necessarily related to one another (as presented in section 5.1).

Several studies have shown that separation of tasks into disjoint groups could boost classification performance. Intuitively, multi-task learning among tasks that are mutually related reduces noise in prediction (Liu and Pan, 2017; Zhong et al., 2016; Jeong and Jun, 2018). We present an algorithm that divides all of the tasks, i.e., all entities predictions, into task groups according to their inherent relatedness. Capturing these connections is performed using a co-occurrence matrix that we compute based on training-set information and behavior. Conceptually, latent entities that are labeled many times together in processes would be considered as related, thus grouped together in a joint multi-task classification unit.

Our tasks are divided into groups based on a co-occurrence matrix M which is computed as follows:

$$M_{ij} = \frac{\#\text{ mutual occurrences of } e_i, e_j}{\#\text{ occurrences of } e_i}$$

where e_i is the i-th latent entity that should be predicted. Additionally, note the elements of M are normalized. Figure 2 presents an example of such a co-occurrence matrix for 5 sampled entities.

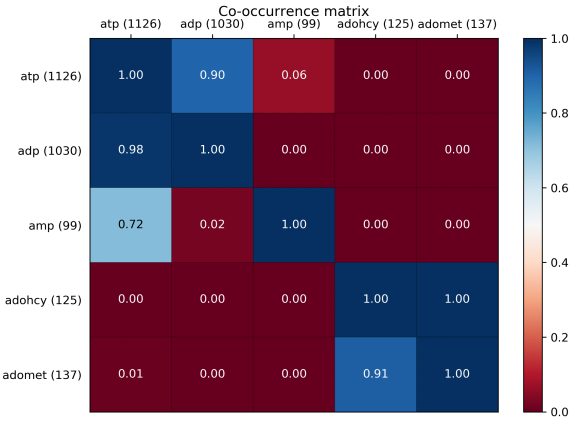

Figure 2: An example of co-occurrence matrix which describes the relatedness of entities to one another. Numbers in parentheses next to entities' names are an indication for their frequency in the training-set. As follows from the distribution, ATP and ADP are high correlated. AMP also tends to co-exist with ATP (not reciprocally though). Similarly, ADOHCY and ADOMET are quite related to one another.

After generating the co-occurrence matrix, we leverage it to select task groups. We denote α as a

minimum threshold in order to group a pairwise of tasks together ($0 \leq \alpha \leq 1$). Then, two prediction tasks (a pair of entities) e_i and e_j will be grouped together if $M_{ij} > \alpha$ or $M_{ji} > \alpha$. Later, we would like to avoid from multi-task group that contains one task only. Therefore, if any singletons remain, we attach each one of them to its most related entity's group, according to the same co-occurrence distribution. This reunion phase comes with the exception of $\alpha/2$ as a minimum threshold rather than α as was done previously.

Clusters of tasks are computed according to $\alpha = 0.65$. This value is chosen empirically such that groups are divided fairly, in terms of size, both subjectively and objectively.

This process induces a division of the tasks to T disjoint groups of tasks, where each group is consisted of k_r prediction tasks (a task per latent entity), where $r \in \{1, 2, \ldots, T\}$. Note that each group is potentially of different size, i.e., k_r is not fixed. Ultimately, these groups are going to represent as the multi-task units of classification in our model.

Figure 3 illustrates the design of our architecture along with the task-grouping layer. It contains an embedding layer, a Bi-GRU components that are fed by the embedding, and ultimately T multi-task classification units, one per task group.

Classifier Similarly to the classifier in Section 5.1, the first and last cell of the GRU are connected to several disjoint groups of prediction tasks. These outputs represent the features for several multi-task classifier units, one such unit per a group of tasks. For the r-th ($r \in \{1, 2, ..., T\}$) task group, we define a classifier unit as a fully connected layer with a sigmoid activation layer with k_r outputs, where k_r is the size of the r-th task group.

Loss Function As opposed to the loss function previously presented, here we would like to preserve relatedness among prediction tasks when they are actually related only. Therefore, we use task grouping feature to recognize T disjoint sets of entities as prediction task groups. For each task group, we force the preservation of entities' known correlation using a unique loss function that is designated for the classifier of that specific group. Denote m as the number of training samples, and k_r as the number of entities that are grouped in the r-th task-group ($r \in \{1, 2, \ldots, T\}$). The need for latent entities from the same task group to retain their relatedness, will be forced using the following loss function:

$$L(y, \hat{y}) = \\ -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k_r}\left(y_j^{(i)}\log\hat{y}_j^{(i)} + \left(1-y_j^{(i)}\right)\log\left(1-\hat{y}_j^{(i)}\right)\right)$$

where y and $\hat{y}$ are the labeled and predicted values, respectively. Whereas the concept is similar to the presented loss function in the vanilla multi-task approach, now each task group classifier has a customized loss function that learns the behavior of its relevant entities it is responsible of.

Note that the penalty for each predicted value is equal while in the same task group, whereas, between different task-groups the loss value may be different. In that way, we refine the classification per each task-group, and thus per each latent entity.

6 Experimental Results

In this section, we evaluate the algorithms for LEE. We first show the performance of the algorithms for a single entity extraction, focusing on the ATP entity. We then present results for the general task of LEE, extracting multiple entities from a text. We then conclude this section by a few qualitative examples illustrating the feature importance considered for the LEE task over several texts.

6.1 Single Latent Entity Extraction

We start by exploring the performance of the different classifiers for the task of identifying a single latent entity. As a representative test-case we consider the extraction of the *A*TP entity. The entity is considered of high importance to many biological processes. Additionally, it has the highest frequency in the dataset, i.e., there are many data examples (biochemical reactions) where ATP is involved in. In more than 81% of its existences in reactions, it is not explicitly mentioned in the text, which practically makes it to a pure latent entity.

The results of all the algorithms for the prediction of the latent entity ATP are shown in Table 2. We emphasize that here, training, validating and testing were all performed on pure latent samples, i.e., texts that did contain the examined entity were filtered out. The last row stands for the multi-task approach with grouping, where ADP was selected

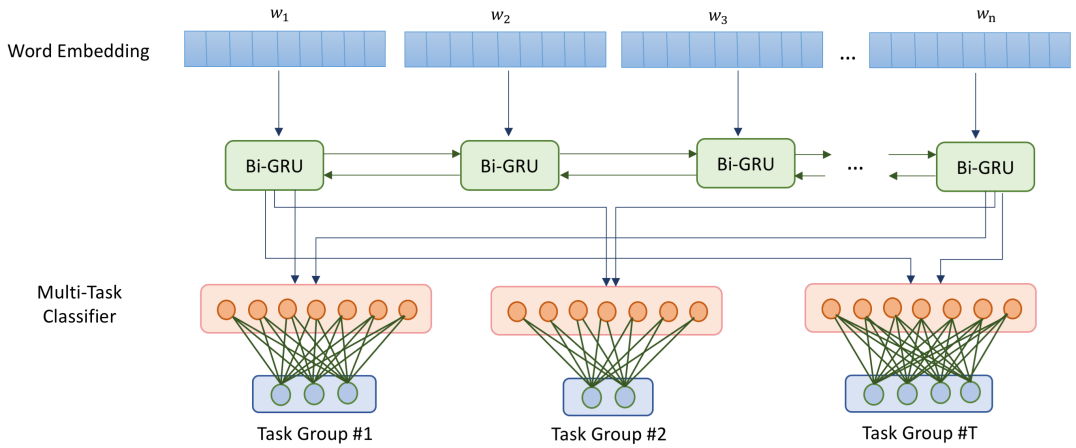

Figure 3: Multi-Task model architecture for latent entities extraction, based on task grouping approach. Word embeddings are fed to several Bi-GRU units which are connected via a multi-task learning approach to numerous groups of tasks, each representing a different group of related latent entities sharing a similar loss.

to be in the same prediction task group along with ATP (ADP is known to be high correlated with ATP as also can be deduced from Figure 2). The improved empirical results in that experiment suggest that using multi-task learning for related tasks could be beneficial for the performance.

6.2 Multiple Latent Entities Extraction

In this section, we consider the full problem of LEE of extracting multiple entities jointly. The results are presented in Table 3.

We measure the performance in two metrics: *micro-average* and *macro-average*. Whereas micro method is considered to be a measurement for the quality of all the predictions altogether, macro stands for the performance of predictions per each task. Note that the average is calculated over the number of latent entities to extract.

Among the one-vs-all possible methods (Section 4), the most successful method, in terms of macro metric, is the bag-of-words & SVM model (section 4.1). At first sight, it could be surprising that such a simple approach outperforms more sophisticated methods, and mainly the deep-learning techniques. We speculate that this is an outcome of the data-set imbalance. That imbalance holds in the sense that different entities could occur in different frequencies in data examples. For example, there are quite many training examples of ATP and ADP (both are involved in more than 14% of the reactions), while other entities may be significantly less frequent (e.g. Oxygen, NADPH, NADP+ and more occurs in less than 3% of the reactions). Therefore, many classes of entities have very little training examples. This does not allow deep-learning models to train well, and therefore the macro score of SVM methods tends to be higher. The reason the SVM with BOW performs better than the more semantic embeddings (Section 4.2–4.4) with SVM might also be due to the low amount of training examples that cause the contribution of semantics to be limited for the LEE task in this dataset.

The vanilla multi-task approach as described in Section 5.1, performs well according to micro-averaging metric, but fails in terms of macro measurement.

Ultimately, our proposed multi-task GRU based model with task-grouping (Section 5.2), outperforms all other baselines in both metrics: micro and macro. Thus, not only generally extracting entities with high performance, but also preserving fairness among different prediction tasks. We conclude that selecting the tasks to learn together in the a multi-task approach is critical for the LEE task.

6.3 Multi-Task Approach Contribution in Multiple Latent Entity Extraction

It should be noted that multi-task learning approach is much more effective in the multiple latent entity extraction (Table 3) compared to the single latent entity extraction case (Table 2). Specifically, multi-task learning approach along

Model	Precision	Recall	F1
Bag-of-Words (TF-IDF) & SVM (Section 4.1)	0.803	0.873	0.837
Weighted-average Embedding & SVM (Section 4.2)	0.770	0.746	0.758
Element-wise Document Embedding & SVM (Section 4.3)	0.817	0.817	0.817
Combined Document Embedding & SVM (Section 4.4)	0.823	0.810	0.816
Pre-Trained PubMed Word Embedding & Bi-LSTM (Section 4.5)	0.888	0.867	0.877
Pre-Trained PubMed Word Embedding & Bi-GRU (Section 4.5)	0.899	0.836	0.866
Multitask - Embedding based Bi-GRU (Section 5.1)	0.869	**0.883**	0.876
Multitask - Embedding based Bi-LSTM (Section 5.1)	0.869	**0.883**	0.876
Multitask with Grouping - Embedding based Bi-LSTM (Section 5.2)	0.909	0.859	**0.884**
Multitask with Grouping - Embedding based Bi-GRU (Section 5.2)	**0.914**	0.828	0.869

Table 2: Extraction of ATP as a latent entity. Statistically significant results are shown in bold.

Model	*micro*			*macro*		
	Prec.	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
Bag-of-Words (TF-IDF) & SVM (Section 4.1)	0.784	0.746	0.795	0.785	0.750	0.754
Weighted-average Embedding & SVM (Section 4.2)	0.682	0.649	0.665	0.636	0.565	0.580
Element-wise Document Embedding & SVM (Section 4.3)	0.740	0.728	0.734	0.696	0.647	0.648
Combined Document Embedding & SVM (Section 4.4)	0.743	0.729	0.736	0.707	0.651	0.656
Pre-Trained PubMed Word Embedding & Bi-LSTM (Section 4.5)	0.817	0.773	0.794	0.707	0.645	0.664
Pre-Trained PubMed Word Embedding & Bi-GRU (Section 4.5)	0.798	0.817	0.808	0.707	0.690	0.687
Multitask - Embedding based Bi-GRU (Section 5.1)	0.798	0.820	0.809	0.671	0.664	0.662
Multitask & Task-Grouping - Embedding based Bi-GRU (Section 5.2)	**0.822**	**0.849**	**0.835**	**0.809**	**0.839**	**0.811**

Table 3: Results of multiple latent entities extraction of top 40 frequent entities. Left side is *micro* metric based, while the right side is according to *macro* metric. Statistically significant results are shown in bold.

with task-grouping performs much better than the other baselines. Naturally, the wins are significant in terms of macro-metric, as our loss-function (as defined in Section 5.2) is aimed for macro optimization. However, we notice that the method also improves performance in terms of micro-metric. To motivate this, consider an example of a sentence with water as a latent entity. Let us assume water is not appearing many times in the corpora, but appears many times in the data with Oxygen. As water is not appearing in many sentences it would be hard to learn indicative features in a sentence to predict it. However, in many cases it is possible to infer Oxygen. The prior of having an Oxygen as latent entity in the sentence can be considered as an indicative feature that also helps to predict water as a latent entity. As those entities do not appear many times in the corpus, learning the indicative features for a multi-task learner is hard. However, when only grouping relevant entities, we then overcome this issue and scores are improved.

Table 2 provides results on the extraction of the ATP entity only. Since there are many training examples for this entity in the corpus (most frequent latent entity), it is possible to learn indicative features even in non-multitask models, which therefore perform well. Thus, there is a small difference between multitask and non-multitask approaches in Table 2. On the other hand, in Table 3 we examine the performance over the top-40 frequent entities, including very frequent entities (such ATP and ADP), and less frequent (such Oxygen, NADPH, NADP+ and water) as well. This leads to the results over all entities both frequent and infrequent to be much better in multitask learning settings with task-grouping specifically.

We observed this phenomena across all latent entities prediction, and present the results for the top frequent entities in the dataset, for the two best performing classifiers (bag-of-words embedding with SVM classification and multi task with grouping) in Table 4.

6.4 Qualitative Examples

To help understand the LEE problem, we present several examples of prominent words that con-

Entity	Bag-of-Words AUC	Grouped-MTL AUC
ATP	0.906	**0.938**
ADP	0.910	**0.965**
H2O	0.864	**0.928**
PI	0.872	**0.937**
H+	**0.924**	0.889
O2	0.904	**0.928**
NADPH	0.917	**0.998**
NADP+	0.918	**0.972**
COA-SH	0.960	**0.998**

Table 4: AUC scores for bag-of-words vectors & SVM baseline performance compared to the multi-task learner with task-grouping. The results are shown for top frequent entities in the data set. Statistically significant results are shown in bold.

tribute to the prediction of a latent entity. We leverage LIME algorithm (Ribeiro et al., 2016) to explain the multi task learning algorithm and present feature importance for ATP and NADPH in Figure 4.

The model inferred that words such as *phosphorylation* or *phosphorylates* are good indicators for the existence of ATP. Phosphorylation is the process through which a phosphate group, which is usually provided by ATP, is transferred from one molecule to a protein.

To infer NADPH, the algorithm gives a high importance to the words P450 and reductase. Cytochrome P450 are proteins that use a variety of molecules as substrates in enzymatic reactions. They usually serve as oxidase enzymes in electron transfer chains. One of the common system they are involved in are microsomal P450 systems, where electrons are transferred from NADPH via cytochrome P450 reductase.

Weight	Feature
0.0939	phosphorylation
0.0741	phosphorylates
0.0534	kinase
0.0449	phosphorylate
0.0273	autophosphorylation
0.0214	synthetase
0.0184	serine
0.0181	activation loop
0.0181	binds
0.0170	phosphorylated

(a) ATP Extraction Top Features

Weight	Feature
0.0609	p450
0.0369	reductase
0.0357	atral trans
0.0341	cytochrome
0.0319	oxidation
0.0308	arachidonic
0.0307	associated endoplasmic
0.0302	estradiol
0.0294	nadp
0.0289	p450s

(b) NADPH Extraction Top Features

Figure 4: An example of prominent words when inferring latent entities.

7 Conclusions

In this paper, we presented a new task of latent entities extraction from text, which gives a new insight over the original named-entity recognition task. Specifically, we focus on how to extract an entity when it is not explicitly mentioned in the text, but rather implicitly mentioned in the context.

We developed several methods to detect existence of such entities in texts, and present a large labeled dataset for exploring the LEE task, and perform an extensive evaluation of our methods. We explore one-vs-all methods with numerous methods to embed the text and a multi-task learning approach that attempts to predict several entities at once. We observe that learning *highly-relevant* entities together when during LEE prediction substantially boosts detection performance. We present several explanations of the classification, as they are taken into account behind the scenes of the best-performing classifier for LEE.

For future work, we consider learning the LEE in an end-to-end fashion, learning to weight which tasks to group together to improve LEE.

We believe the LEE task would spur additional research in the field to improve NER when entities are implicitly mentioned and help better comprehend complex texts.

References

Antonios Anastasopoulos and David Chiang. 2018. Tied multitask learning for neural speech translation. *CoRR*, abs/1802.06655.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings.

Bogdan Babych and Anthony Hartley. 2003. Improving machine translation quality with automatic named entity recognition. In *Proceedings of the 7th International EAMT workshop on MT and other*

Language Technology Tools, Improving MT through other Language Technology Tools: Resources and Tools for Building MT, pages 1–8. Association for Computational Linguistics.

Jonathan Berant and Peter Clark. 2014. Modeling Biological Processes for Reading Comprehension. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1510.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, pages 95–133. Springer.

Hsinchun Chen, Wingyan Chung, Jennifer Jie Xu, Gang Wang, Yi Qin, and Michael Chau. 2004. Crime data mining: a general framework and some examples. *computer*, 37(4):50–56.

Billy Chiu, Gamal Crichton, Anna Korhonen, and Sampo Pyysalo. 2016. How to Train Good Word Embeddings for Biomedical NLP. pages 166–174.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Kyunghyun Cho, Bart van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

David Croft, Gavin O 'kelly, Guanming Wu, Robin Haw, Marc Gillespie, Lisa Matthews, Michael Caudy, Phani Garapati, Gopal Gopinath, Bijay Jassal, Steven Jupe, Irina Kalatskaya, Shahana Mahajan, Bruce May, Nelson Ndegwa, Esther Schmidt, Veronica Shamovsky, Christina Yung, Ewan Birney, Henning Hermjakob, Peter D 'eustachio, and Lincoln Stein. Reactome: a database of reactions, pathways and biological processes.

Cedric De Boom, Steven Van Canneyt, Thomas Demeester, and Bart Dhoedt. 2016. Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recognition Letters*, 80(C):150–156.

Tobias Domhan and Felix Hieber. 2017. Using target-side monolingual data for neural machine translation through multi-task learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1500–1505.

Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: A graph-based approach to abstractive summarization of highly redundant opinions. pages 340–348.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*, pages 57–60. Association for Computational Linguistics.

Masaru Isonuma, Toru Fujino, Junichiro Mori, Yutaka Matsuo, and Ichiro Sakata. 2017. Extractive summarization using multi-task learning with document classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2101–2110.

Jun-Yong Jeong and Chi-Hyuck Jun. 2018. Variable selection and task grouping for multi-task learning. *arXiv preprint arXiv:1802.04676*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Sulin Liu and Sinno Jialin Pan. 2017. Adaptive group sparse multi-task learning via trace lasso. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2358–2364. AAAI Press.

Mónica Marrero, Julián Urbano, Sonia Sánchez-Cuadrado, Jorge Morato, and Juan Miguel Gómez-Berbís. 2013. Named entity recognition: fallacies, challenges and opportunities. *Computer Standards & Interfaces*, 35(5):482–489.

Alexandre Passos, Vineet Kumar, and Andrew D McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *CoNLL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152.

Piyush Rai and Hal Daume III. 2010. Infinite predictor subspace models for multitask learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 613–620.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Shi Zhong, Jian Pu, Yu-Gang Jiang, Rui Feng, and Xiangyang Xue. 2016. Flexible multi-task learning with latent task grouping. *Neurocomputing*, 189:179–188.