Lightly-supervised Representation Learning with Global Interpretability

Andrew Zupon, Maria Alexeeva, Marco A. Valenzuela-Escárcega,
Ajay Nagesh, and **Mihai Surdeanu**
University of Arizona, Tucson, AZ, USA
{zupon, alexeeva, marcov, ajaynagesh, msurdeanu}
@email.arizona.edu

Abstract

We propose a lightly-supervised approach for information extraction, in particular named entity classification, which combines the benefits of traditional bootstrapping, i.e., use of limited annotations and interpretability of extraction patterns, with the robust learning approaches proposed in representation learning. Our algorithm iteratively learns custom embeddings for both the multi-word entities to be extracted and the patterns that match them from a few example entities per category. We demonstrate that this representation-based approach outperforms three other state-of-the-art bootstrapping approaches on two datasets: CoNLL-2003 and OntoNotes. Additionally, using these embeddings, our approach outputs a globally-interpretable model consisting of a decision list, by ranking patterns based on their proximity to the average entity embedding in a given class. We show that this interpretable model performs close to our complete bootstrapping model, proving that representation learning can be used to produce interpretable models with small loss in performance. This decision list can be edited by human experts to mitigate some of that loss and in some cases outperform the original model.

1 Introduction

One strategy for mitigating the cost of supervised learning in information extraction (IE) is to bootstrap extractors with light supervision from a few provided examples (or seeds). Traditionally, bootstrapping approaches iterate between learning extraction patterns such as word n-grams, e.g., the pattern "`@ENTITY , former president`" could be used to extract person names,[1] and applying these patterns to extract the desired structures (entities, relations, etc.) (Carlson et al., 2010; Gupta and Manning, 2014, 2015, inter alia). One advantage of this direction is that these patterns are interpretable, which mitigates the maintenance cost associated with machine learning systems (Sculley et al., 2014).

On the other hand, representation learning has proven to be useful for natural language processing (NLP) applications (Mikolov et al., 2013; Riedel et al., 2013; Toutanova et al., 2015, 2016, inter alia). Representation learning approaches often include a component that is trained in an unsupervised manner, e.g., predicting words based on their context from large amounts of data, mitigating the brittle statistics affecting traditional bootstrapping approaches. However, the resulting real-valued embedding vectors are hard to interpret.

Here we argue that these two directions are complementary, and should be combined. We propose such a bootstrapping approach for information extraction (IE), which blends the advantages of both directions. As a use case, we instantiate our idea for named entity classification (NEC), i.e., classifying a given set of unknown entities into a predefined set of categories (Collins and Singer, 1999). The contributions of this work are:

(1) We propose an approach for bootstrapping NEC that iteratively learns custom embeddings for *both* the multi-word entities to be extracted and the patterns that match them from a few example entities per category. Our approach changes the objective function of a neural network language models (NNLM) to include a semi-supervised component that models the known examples, i.e., by *attracting* entities and patterns in the same category to each other and *repelling* them from elements in different categories, and it adds an external iterative process that "cautiously" augments the pools of known examples (Collins and Singer, 1999). In other words, our contribution is an example of combining representation learning and bootstrapping.

(2) We demonstrate that our representation learn-

[1] In this work we use surface patterns, but the proposed algorithm is agnostic to the types of patterns learned.

Proceedings of 3rd Workshop on Structured Prediction for NLP, pages 18–28
Minneapolis, Minnesota, June 7, 2019.

ing approach is suitable for semi-supervised NEC. We compare our approach against several state-of-the-art semi-supervised approaches on two datasets: CoNLL-2003 (Tjong Kim Sang and De Meulder, 2003) and OntoNotes (Pradhan et al., 2013). We show that, despite its simplicity, our method outperforms all other approaches.

(3) Our approach also outputs an interpretation of the learned model, consisting of a decision list of patterns, where each pattern gets a score per class based on the proximity of its embedding to the average entity embedding in the given class. This interpretation is global, i.e., it explains the entire model rather than local predictions. We show that this decision-list model performs comparably to the complete model on the two datasets.

(4) We also demonstrate that the resulting system can be understood, debugged, and maintained by non-machine learning experts. We compare the decision-list model edited by human domain experts with the unedited decision-list model and see a modest improvement in overall performance, with some categories getting a bigger boost. This improvement shows that, for non-ambiguous categories that are well-defined by the local contexts captured by our patterns, these patterns truly are interpretable to end users.

2 Related Work

Bootstrapping is an iterative process that alternates between learning representative patterns, and acquiring new entities (or relations) belonging to a given category (Riloff, 1996; McIntosh, 2010). Patterns and extractions are ranked using either formulas that measure their frequency and association with a category, or classifiers, which increases robustness due to regularization (Carlson et al., 2010; Gupta and Manning, 2015). While semi-supervised learning is not novel (Yarowsky, 1995; Gupta and Manning, 2014), our approach performs better than some modern implementations of these methids such as Gupta and Manning (2014).

Distributed representations of words (Deerwester et al., 1990; Mikolov et al., 2013; Levy and Goldberg, 2014) serve as underlying representation for many NLP tasks such as information extraction and question answering (Riedel et al., 2013; Toutanova et al., 2015, 2016; Sharp et al., 2016). Mrkšić et al. (2017) build on traditional distributional models by incorporating synonymy and antonymy relations as supervision to fine tune word vector spaces, using an Attract/Repel method similar to our idea. However, most of these works that customize embeddings for a specific task rely on some form of supervision. In contrast, our approach is lightly supervised, with a only few seed examples per category. Batista et al. (2015) perform bootstrapping for relation extraction using pre-trained word embeddings. They do not learn custom pattern embeddings that apply to multi-word entities and patterns. We show that customizing embeddings for the learned patterns is important for interpretability.

Recent work has focused on explanations of machine learning models that are model-agnostic but local, i.e., they interpret individual model predictions (Ribeiro et al., 2018, 2016a). In contrast, our work produces a global interpretation, which explains the entire extraction model rather than individual decisions.

Lastly, our work addresses the interpretability aspect of information extraction methods. Interpretable models mitigate the technical debt of machine learning (Sculley et al., 2014). For example, it allows domain experts to make manual, gradual improvements to the models. This is why rule-based approaches are commonly used in industry applications, where software maintenance is crucial (Chiticariu et al., 2013). Furthermore, the need for interpretability also arises in critical systems, e.g., recommending treatment to patients, where these systems are deployed to aid human decision makers (Lakkaraju and Rudin, 2016). The benefits of interpretability have encouraged efforts to either extract interpretable models from opaque ones (Craven and Shavlik, 1996), or to explain their decisions (Ribeiro et al., 2016b).

As machine learning models are becoming more complex, the focus on interpretability has become more important, with new funding programs focused on this topic.[2] Our approach for exporting an interpretable model (§3) is similar to Valenzuela-Escárcega et al. (2016), but we start from distributed representations, whereas they started from a logistic regression model with explicit features.

[2] DARPA's Explainable AI program: http://www.darpa.mil/program/explainable-artificial-intelligence.

3 Approach

Bootstrapping with representation learning

Our algorithm iteratively grows a pool of multi-word entities (entPool$_c$) and n-gram patterns (patPool$_c$) for each category of interest c, and learns custom embeddings for both, which we will show are crucial for both performance and interpretability.

The entity pools are initialized with a few seed examples (seeds$_c$) for each category. For example, in our experiments we initialize the pool for a `person names` category with 10 names such as *Mother Teresa*. Then the algorithm iteratively applies the following three steps for T epochs:

(1) *Learning custom embeddings*: The algorithm learns custom embeddings for all entities and patterns in the dataset, using the current entPool$_c$s as supervision. This is a key contribution, and is detailed in the second part of this section.

(2) *Pattern promotion*: We generate the patterns that match the entities in each pool entPool$_c$, rank those patterns using point-wise mutual information (PMI) with the corresponding category, and select the top ranked patterns for promotion to the corresponding pattern pool patPool$_c$. In this work, we use use surface patterns consisting of up to 4 words before/after the entity of interest, e.g., the pattern "`@ENTITY , former president`" matches any entity followed by the three tokens ,, *former*, and *president*. However, our method is agnostic to the types of patterns learned, and can be trivially adapted to other types of patterns, e.g., over sytactic dependency paths.

(3) *Entity promotion*: Entities are promoted to entPool$_c$ using a multi-class classifier that estimates the likelihood of an entity belonging to each class (Gupta and Manning, 2015). Our feature set includes, for each category c: (a) edit distance over characters between the candidate entity e and current e_cs $\in$ entPool$_c$, (b) the PMI (with c) of the patterns in patPool$_c$ that matched e in the training documents, and (c) similarity between e and e_cs in a semantic space. For the latter feature group, we use the set of embedding vectors learned in step (1). These features are taken from Gupta and Manning (2015). We use these vectors to compute the cosine similarity score of a given candidate entity e to the entities in entPool$_c$, and add the average and maximum similarities as features. The top 10 entities classified with the highest confidence for each class are promoted to the corresponding entPool$_c$ after each epoch.

Learning custom embeddings

We train our embeddings for both entities and patterns by maximizing the objective function J:

$$J = \text{SG} + \text{Attract} + \text{Repel} \tag{1}$$

where SG, Attract, and Repel are individual components of the objective function designed to model both the unsupervised, language model part of the task as well as the light supervision coming from the seed examples, as detailed below. A similar approach is proposed by (Mrkšić et al., 2017), who use an objective function modified with Attract and Repel components to fine-tune word embeddings with synonym and antonym pairs.

The SG term is formulated identically to the original objective function of the *S*kip-*G*ram model of Mikolov et al. (2013), but, crucially, adapted to operate over multi-word entities and contexts consisting not of bags of context words, but of the patterns that match each entity. Thus, intuitively, our SG term encourages the embeddings of entities to be similar to the embeddings of the patterns matching them:

$$\begin{aligned}\text{SG} = \sum_{e} [&\log(\sigma(V_e^\top V_{pp})) + \\ &\sum_{np} \log(\sigma(-V_e^\top V_{np}))]\end{aligned} \tag{2}$$

where e represents an entity, pp represents a positive pattern, i.e., a pattern that matches entity e in the training texts, np represents a negative pattern, i.e., it has not been seen with this entity, and σ is the sigmoid function. Intuitively, this component forces the embeddings of entities to be similar to the embeddings of the patterns that match them, and dissimilar to the negative patterns.

The second component, Attract, encourages entities or patterns in the same pool to be close to each other. For example, if we have two entities in the pool known to be person names, they should be close to each other in the embedding space:

$$\text{Attract} = \sum_{P} \sum_{x1,x2 \in P} \log(\sigma(V_{x1}^\top V_{x2})) \tag{3}$$

where P is the entity/pattern pool for a category, and $x1, x2$ are entities/patterns in said pool.

Lastly, the third term, Repel, encourages that the pools be mutually exclusive, which is a soft version of the counter training approach of Yangarber (2003) or the weighted mutual-exclusive bootstrapping algorithm of McIntosh and Curran (2008). For example, person names should be far from organization names in the semantic embedding space:

$$\text{Repel} = \sum_{P1,P2 \text{ if } P1 \neq P2} \sum_{x1 \in P1} \sum_{x2 \in P2} \log(\sigma(-V_{x1}^{\top} V_{x2})) \quad (4)$$

where $P1, P2$ are different pools, and $x1$ and $x2$ are entities/patterns in $P1$, and $P2$, respectively.

We term the complete algorithm that learns and uses custom embeddings as *Emboot* (*Em*beddings for *boot*strapping), and the stripped-down version without them as *EPB* (*E*xplicit *P*attern-based *B*ootstrapping). EPB is similar to Gupta and Manning (2015); the main difference is that we use pretrained embeddings in the entity promotion classifier rather than Brown clusters. In other words, EPB relies on pretrained embeddings for both patterns and entities rather than the custom ones that Emboot learns.[3]

Interpretable model

In addition to its output (entPool$_c$s), Emboot produces custom entity and pattern embeddings that can be used to construct a decision-list model, which provides a global, deterministic interpretation of what Emboot learned.

This interpretable model is constructed as follows. First, we produce an average embedding per category by averaging the embeddings of the entities in each entPool$_c$. Second, we estimate the cosine similarity between each of the pattern embeddings and these category embeddings, and convert them to a probability distribution using a softmax function; $prob_c(p)$ is the resulting probability of pattern p for class c.

After being constructed, the interpretable model is used as follows. First, each candidate entity to be classified, e, receives a score for a given class c from all patterns in patPool$_c$ that match it. The entity score aggregates the relevant pattern probabilities using Noisy-Or:

$$Score(e, c) = 1 - \prod_{\{p_c \in patPool_c | matches(p_c, e)\}} (1 - prob_c(p_c)) \quad (5)$$

Each entity is then assigned to the category with the highest overall score.

4 Experiments

We evaluate the above algorithms on the task of named entity classification from free text.

Datasets: We used two datasets, the CoNLL-2003 shared task dataset (Tjong Kim Sang and De Meulder, 2003), which contains 4 entity types, and the OntoNotes dataset (Pradhan et al., 2013), which contains 11.[4] These datasets contain marked entity boundaries with labels for each marked entity. Here we only use the entity boundaries but *not* the labels of these entities during the training of our bootstrapping systems. To simulate learning from large texts, we tuned hyper parameters on development, but ran the actual experiments on the *train* partitions.

Baselines: In addition to the EPB algorithm, we compare against the approach proposed by Gupta and Manning (2014)[5]. This algorithm is a simpler version of the EPB system, where entities are promoted with a PMI-based formula rather than an entity classifier.[6] Further, we compare against label propagation (LP) (Zhu and Ghahramani, 2002), with the implementation available in the `scikit-learn` package.[7] In each bootstrapping epoch, we run LP, select the entities with the lowest entropy, and add them to their top category. Each entity is represented by a feature vector that contains the co-occurrence counts of the entity and each of the patterns that matches it in text.[8]

Settings: For all baselines and proposed models, we used the same set of 10 seeds/category, which were manually chosen from the most frequent entities in the dataset. For the custom embedding

[3] For multi-word entities and patterns, we simply average word embeddings to generate entity and pattern embeddings for EPB.

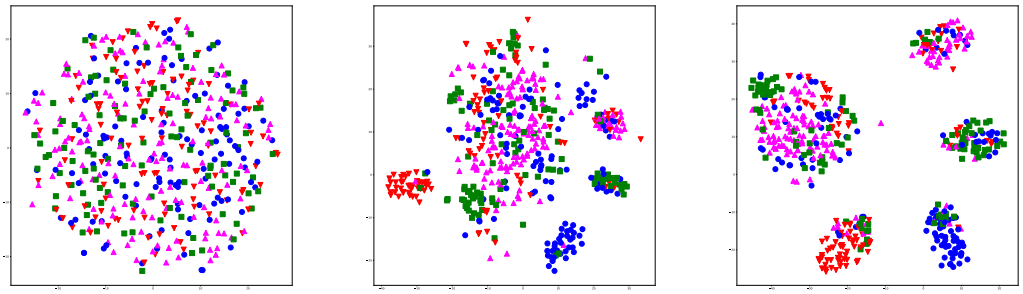
[4] We excluded numerical categories such as `DATE`.

[5] https://nlp.stanford.edu/software/patternslearning.shtml

[6] We did not run this system on OntoNotes dataset as it uses a builtin NE classifier with a predefined set of labels which did not match the OntoNotes labels.

[7] http://scikit-learn.org/stable/modules/generated/sklearn.semi_supervised.LabelPropagation.html

[8] We experimented with other feature values, e.g., pattern PMI scores, but all performed worse than raw counts.



(a) Embeddings initialized randomly (b) Bootstrapping epoch 5 (c) Bootstrapping epoch 10

Figure 1: t-SNE visualizations of the entity embeddings at three stages during training.
Legend: ● = LOC. ■ = ORG. ▲ = PER. ▼ = MISC. This figure is best viewed in color.

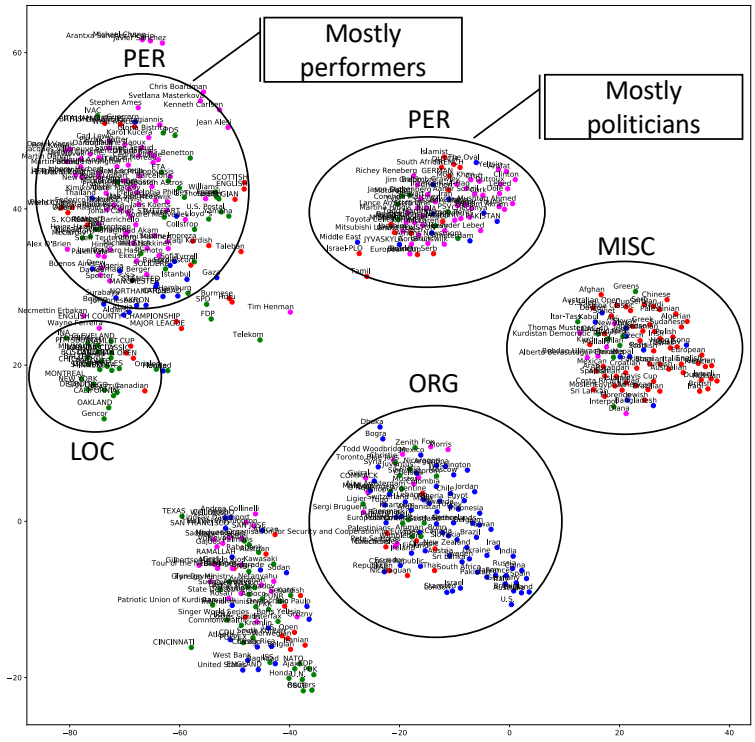


Figure 2: t-SNE visualizations of the entity embeddings learned by Emboot after training completes.
Legend: ● = LOC. ● = ORG. ● = PER. ● = MISC.

features, we used randomly initialized 15d embeddings. Here we consider patterns to be n-grams of size up to 4 tokens on either side of an entity. For instance, "`@ENTITY , former President`" is one of the patterns learned for the class `person`. We ran all algorithms for 20 bootstrapping epochs, and the embedding learning component for 100 epochs in each bootstrapping epoch. We add 10 entities and 10 patterns to each category during every bootstrapping epoch.

5 Discussion

Qualitative Analysis

Before we discuss overall results, we provide a qualitative analysis of the learning process for Emboot for the CoNLL dataset in Figure 1. The figure shows t-SNE visualizations (van der Maaten and Hinton, 2008) of the entity embeddings at several stages of the algorithm. This visualization matches our intuition: as training advances, entities belonging to the same category are indeed

grouped together. In particular, Figure 1c shows five clusters, four of which are dominated by one category (and centered around the corresponding seeds), and one, in the upper left corner, with the entities that haven't yet been added to any of the pools.

Figure 2 shows a more detailed view of the t-SNE projections of entity embeddings after Emboot's training completes. Again, this demonstrates that Emboot's semi-supervised approach clusters most entities based on their (unseen) categories. Interestingly, Emboot learned two clusters for the PER category. Upon a manual inspection of these clusters, we observed that one contains mostly performers (e.g., athletes or artists such as Stephen Ames, a professional golfer), while the other contains many politicians (e.g., Arafat and Clinton). Thus, Emboot learned correctly that, at least at the time when the CoNLL 2003 dataset was created, the context in which politicians and performers were mentioned was different. The cluster in the bottom left part of the figure contains the remaining working pool of patterns, which were not assigned to any category cluster after the training epochs.

Quantitative Analysis

A quantitative comparison of the different models on the two datasets is shown in Figure 3.

Figure 3 shows that Emboot considerably outperforms LP and Gupta and Manning (2014), and has an occasional improvement over EPB. While EPB sometimes outperforms Emboot, Emboot has the potential for manual curation of its model, which we will explore later in this section. This demonstrates the value of our approach, and the importance of custom embeddings.

Importantly, we compare Emboot against: (a) its interpretable version ($\text{Emboot}_{\text{int}}$), which is constructed as a decision list containing the patterns learned (and scored) after each bootstrapping epoch, and (b) an interpretable system built similarly for EPB (EPB_{int}), using the pretrained Levy and Goldberg embeddings[9] rather than our custom ones. This analysis shows that $\text{Emboot}_{\text{int}}$ performs close to Emboot on both datasets, demonstrating that most of the benefits of representation learning are available in an interpretable model. Please see the discussion on the edited interpretable model in the next section.

Importantly, the figure also shows that EPB_{int}, which uses generic entity embeddings rather than the custom ones learned for the task performs considerably worse than the other approaches. This highlights the importance of learning a dedicated distributed representation for this task.

Interpretability Analysis

Is the list of patterns generated by the interpretable model actually interpretable to end users? To investigate this, we asked two linguists to curate the models learned by $\text{Emboot}_{\text{int}}$, by editing the list of patterns in a given model. First, the experts performed an independent analysis of all the patterns. Next, the two experts conferred with each other and came to a consensus when their independent decisions on a particular pattern disagreed. These first two stages took the experts 2 hours for the CoNLL dataset and 3 hours for the OntoNotes dataset. The experts did not have access to the original texts the patterns were pulled from, so they had to make their decisions based on the patterns alone. They made one of three decisions for each pattern: (1) no change, when the pattern is a good fit for the category; (2) changing the category, when the pattern clearly belongs to a different category; and (3) deleting the pattern, when the pattern is either not informative enough for any category or when the pattern could occur with entities from multiple categories. The experts did not have the option of modifying the content of the pattern, because each pattern is associated with an embedding learned during training. Table 1 shows several examples of the patterns and decision made by the annotators. A summary of the changes made for the CoNLL dataset is given in Figure 4, and a summary of the changes made for the OntoNotes dataset is given in Figure 5.

As Figure 3 shows, this edited interpretable model ($\text{Emboot}_{\text{int-edited}}$) performs similarly to the unedited interpretable model. When we look a little deeper, the observed overall similarity between the unchanged and the edited interpretable Emboot models for both datasets depends on the specific categories and the specific patterns involved. For example, when we look at the CoNLL dataset, we observe that the edited model outperforms the unchanged model on PER entities, but performs worse than the unchanged model on ORG enti-

[9] For multi-word entities, we averaged the embeddings of the individual words in the entity to create an overall entity embedding.

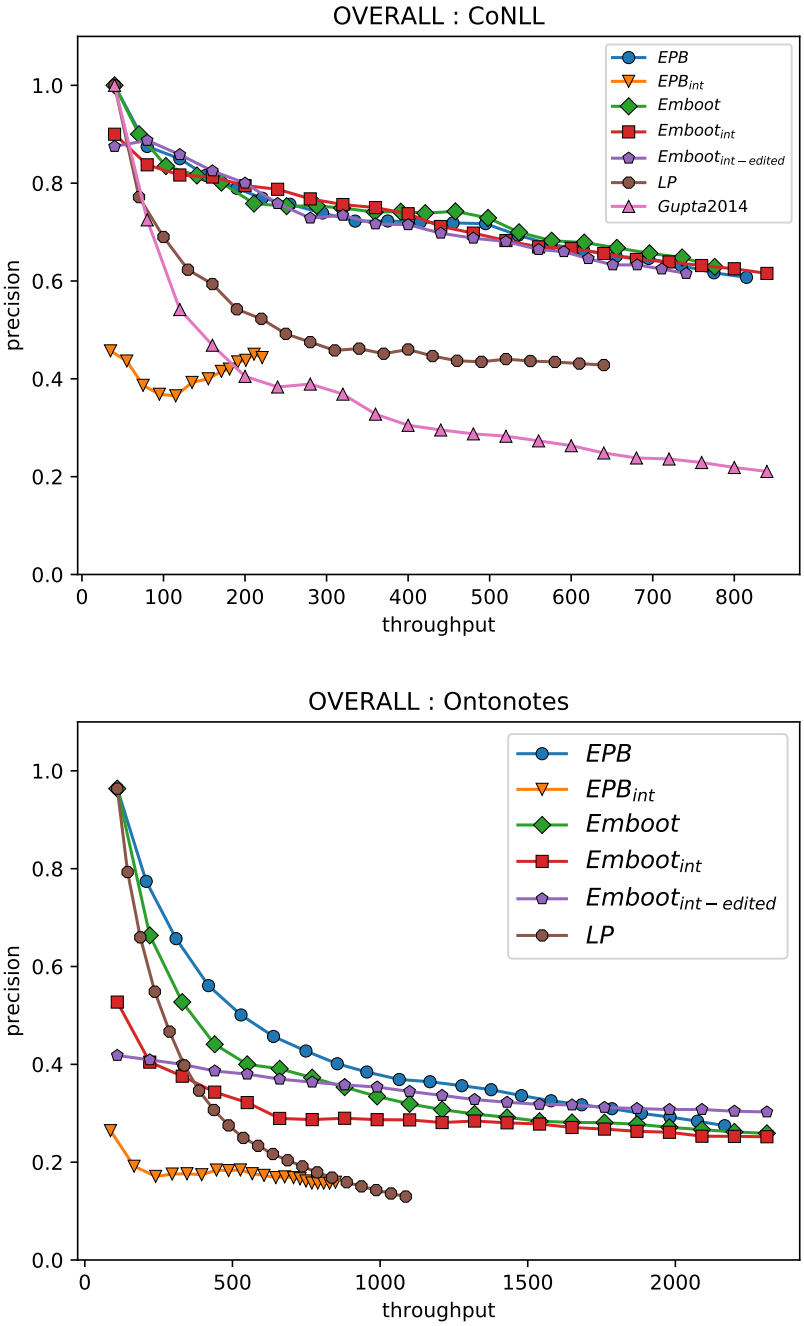


Figure 3: Overall results on the CoNLL and OntoNotes datasets. Throughput is the number of entities classified, and precision is the proportion of entities that were classified correctly. Please see Sec. 4 for a description of the systems listed in the legend.

Pattern	Original Label	Decision	Rationale
@ENTITY was the	LOC	delete	the pattern is too broad
@ENTITY) Ferrari	LOC	delete	the pattern is uninformative
citizen of @ENTITY	MISC	change to LOC	the pattern is more likely to occur with a location
According to @ENTITY officials	ORG	no change	the pattern is likely to occur with an organization

Table 1: Examples of patterns and experts' decisions and rationales from the CoNLL dataset.

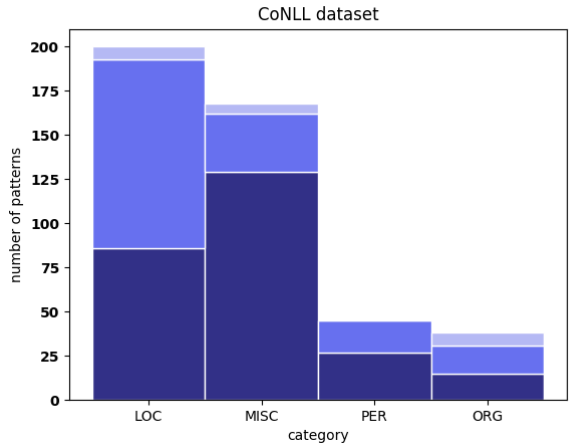


Figure 4: Summary of expert decisions when editing the Emboot$_{\text{int}}$ model, for the CoNLL dataset by original category. Dark blue (bottom) is no change, medium blue (middle) is deletions, light blue (top) is change of category.

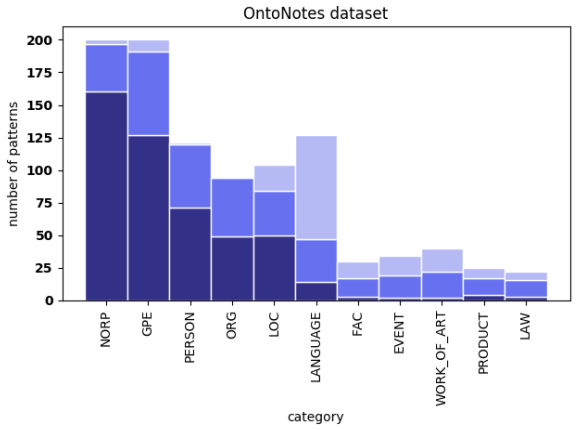


Figure 5: Summary of expert decisions when editing the Emboot$_{\text{int}}$ model, for the OntoNotes dataset by original category. Dark blue (bottom) is no change, medium blue (middle) is deletions, light blue (top) is change of category.

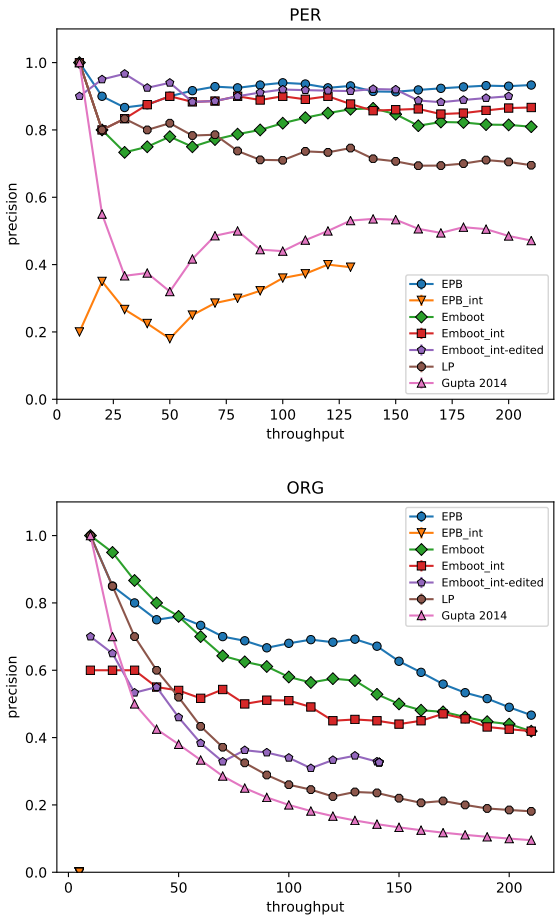


Figure 6: CoNLL results for PER and ORG. The Emboot$_{\text{int-edited}}$ model generally outperforms the Emboot$_{\text{int}}$ model when it comes to PER entities, but not for ORG entities. This discrepancy seems to relate to the amount of local information available in PER patterns versus ORG patterns that can aid human domain experts in correcting the patterns. The EPB$_{\text{int}}$ model (incorrectly) classifies very few entities as ORG, which is why it only shows up as a single point in the bottom left of the lower plot.

ties (Figure 6). We observe a similar pattern with the OntoNotes dataset, where the Emboot$_{\text{int-edited}}$ model outperforms the Emboot$_{\text{int}}$ model greatly for GPE but not for LAW (Figure 7). Overall, for OntoNotes, Emboot$_{\text{int-edited}}$ outperforms Emboot$_{\text{int}}$ for 5 categories out of 11. For the categories where Emboot$_{\text{int-edited}}$ performs worse, the data is sparse, so few patterns are promoted (30 FAC patterns compared to 200 GPE patterns), and many of them were deleted or changed by the two linguists (14 FAC deletions and 13 FAC changes, with only 3 FAC patterns remaining).

This difference in outcome partially has to do with the amount of *local* vs. *global* information available in the patterns. For example, local patterns are common for the PER and MISC categories in CoNLL, and for the GPE category in OntoNotes, e.g., the entity "Syrian" is correctly classified as MISC (which includes demonyms) due to two patterns matching it in the CoNLL dataset: "`@ENTITY President`" and "`@ENTITY troops`". In general, the majority of predictions are triggered by 1 or 2 patterns, which makes these decisions explainable. For the CoNLL dataset, 59% of Emboot$_{\text{int}}$'s predictions are triggered by 1 or 2 patterns; 84% are generated by 5 or fewer patterns; only 1.1% of predictions are generated by 10 or more patterns.

On the other hand, without seeing the full source text, the experts were not able to

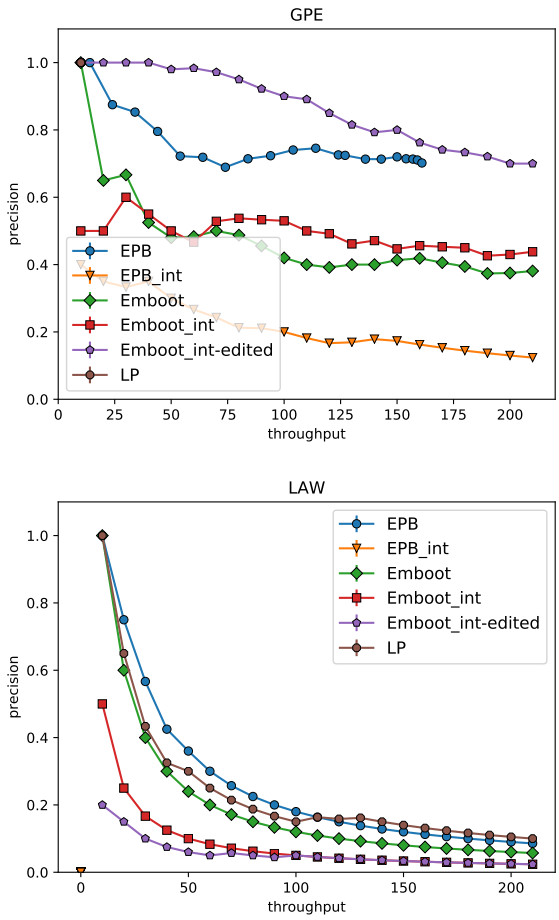


Figure 7: OntoNotes results for GPE and LAW. The $\text{Emboot}_{\text{int-edited}}$ model greatly outperforms both Emboot and $\text{Emboot}_{\text{int}}$ models when it comes to GPE entities, but not for LAW entities. This discrepancy seems to relate to the amount of local information available in GPE patterns versus LAW patterns that can aid human domain experts in correcting the patterns. EPB_{int} does not classify any entities as LAW.

make an accurate judgment on the validity of some patterns—for instance, while the pattern `@ENTITY and Portugal` clearly indicates a geo-political entity, the pattern `@ENTITY has been` (labeled `facility` originally when training on OntoNotes documents) can co-occur with entities from any category. Such patterns are common for the LAW category in OntoNotes, and for the ORG category in CoNLL (due to the focus on sport events in CoNLL, where location names are commonly used as a placeholder for the corresponding team), which partially explains the poor curation results on these categories in Figures 6 and 7. Additionally, lower performance on certain categories can be partially explained by the small amount of data in those categories and the fact that the edits made by the experts drastically changed the number of patterns that occur with some categories (see Figures 4 and 5).

6 Conclusion

This work introduced an example of representation learning being successfully combined with traditional, pattern-based bootstrapping for information extraction, in particular named entity classification. Our approach iteratively learns custom embeddings for multi-word entities and the patterns that match them as well as cautiously augmenting the pools of known examples. This approach outperforms several state-of-the-art semi-supervised approaches to NEC on two datasets, CoNLL 2003 and OntoNotes.

Our approach can also export the model learned into an interpretable list of patterns, which human domain experts can use to understand why an extraction was generated. These patterns can be manually curated to improve the performance of the system by modifying the model directly, with minimal effort. For example, we used a team of two linguists to curate the model learned for OntoNotes in 3 hours. The model edited by human domain experts shows a modest improvement over the unedited model, demonstrating the usefulness of these interpretable patterns. Interestingly, the manual curation of these patterns performed better for some categories that rely mostly on local context that is captured by the type of patterns used in this work, and less well for categories that require global context that is beyond the n-gram patterns used here. This observation raises opportunities for future work such as how to learn global context in an interpretable way, and how to adjust the amount of global information depending on the category learned.

7 Acknowledgments

We gratefully thank Yoav Goldberg for his suggestions for the manual curation experiments.

This work was supported by the Defense Advanced Research Projects Agency (DARPA) under the Big Mechanism program, grant W911NF-14-1-0395, and by the Bill and Melinda Gates Foundation HBGDki Initiative. Marco Valenzuela-Escárcega and Mihai Surdeanu declare a financial interest in lum.ai. This interest has been properly disclosed to the University of Arizona Institutional Review Committee and is managed in accordance with its conflict of interest policies.

References

David S Batista, Bruno Martins, and Mário J Silva. 2015. Semi-supervised bootstrapping of relationship extractors with distributional semantics. In *In Empirical Methods in Natural Language Processing*. ACL.

Andrew Carlson, Justin Betteridge, Richard C Wang, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 101–110. ACM.

Laura Chiticariu, Yunyao Li, and Frederick R Reiss. 2013. Rule-based information extraction is dead! long live rule-based information extraction systems! In *EMNLP*, October, pages 827–832.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Mark W Craven and Jude W Shavlik. 1996. Extracting tree-structured representations of trained networks. *Advances in neural information processing systems*, pages 24–30.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

Sonal Gupta and Christopher D Manning. 2014. Improved pattern learning for bootstrapped entity extraction. In *CoNLL*, pages 98–108.

Sonal Gupta and Christopher D. Manning. 2015. Distributed representations of words to guide bootstrapped entity classifiers. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

Himabindu Lakkaraju and Cynthia Rudin. 2016. Learning cost-effective treatment regimes using markov decision processes. *CoRR*, abs/1610.06972.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *ACL (2)*, pages 302–308.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *The Journal of Machine Learning Research*, 9(2579-2605):85.

Tara McIntosh. 2010. Unsupervised discovery of negative categories in lexicon bootstrapping. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 356–365. Association for Computational Linguistics.

Tara McIntosh and James R Curran. 2008. Weighted mutual exclusion bootstrapping for domain independent lexicon and template acquisition. In *Proceedings of the Australasian Language Technology Association Workshop*, volume 2008.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. 2017. Semantic specialization of distributional word vector spaces using monolingual and cross-lingual constraints. *Transactions of the Association for Computational Linguistics*, 5:309–324.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Bjrkelund, 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, Sofia, Bulgaria. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016a. "why should i trust you?": Explaining the predictions of any classifier. In *Knowledge Discovery and Data Mining (KDD)*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016b. 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.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of NAACL-HLT*.

Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proceedings of the national conference on artificial intelligence*, pages 1044–1049.

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, and Michael Young. 2014. Machine learning: The high interest credit card of technical debt. In *SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop)*.

Rebecca Sharp, Mihai Surdeanu, Peter Jansen, Peter Clark, and Michael Hammond. 2016. Creating causal embeddings for question answering with

minimal supervision. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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 CoNLL-2003*, pages 142–147. Edmonton, Canada.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*, volume 15, pages 1499–1509. Citeseer.

Kristina Toutanova, Xi Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge bases and text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1434–1444.

Marco A Valenzuela-Escárcega, Gus Hahn-Powell, Dane Bell, and Mihai Surdeanu. 2016. Snaptogrid: From statistical to interpretable models for biomedical information extraction. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*, pages 56–65.

Roman Yangarber. 2003. Counter-training in discovery of semantic patterns. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

X. Zhu and Z. Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University.