Embedding Methods for Fine Grained Entity Type Classification

Dani Yogatama
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213
dyogatama@cs.cmu.edu

Dan Gillick, Nevena Lazic
Google Research
1600 Amphitheatre Parkway
Mountain View, CA 94043
{dgillick,nevena}@google.com

Abstract

We propose a new approach to the task of fine grained entity type classifications based on label embeddings that allows for information sharing among related labels. Specifically, we learn an embedding for each label and each feature such that labels which frequently co-occur are close in the embedded space. We show that it outperforms state-of-the-art methods on two fine grained entity-classification benchmarks and that the model can exploit the finer-grained labels to improve classification of standard coarse types.

1 Introduction

Entity type classification is the task of assigning type labels (e.g., `person`, `location`, `organization`) to mentions of entities in documents. These types are useful for deeper natural language analysis such as coreference resolution (Recasens et al., 2013), relation extraction (Yao et al., 2010), and downstream applications such as knowledge base construction (Carlson et al., 2010) and question answering (Lin et al., 2012).

Standard entity type classification tasks use a small set of coarse labels, typically fewer than 20 (Hirschman and Chinchor, 1997; Sang and Meulder, 2003; Doddington et al., 2004). Recent work has focused on a much larger set of fine grained labels (Ling and Weld, 2012; Yosef et al., 2012; Gillick et al., 2014). Fine grained labels are typically subtypes of the standard coarse labels (e.g., `artist` is a subtype of `person` and `author` is a subtype of `artist`), so the label space forms a tree-structured *is-a* hierarchy. See Figure 1 for the label sets used in our experiments. A mention labeled with type `artist` should also be labeled with all ancestors of `artist`. Since we allow mentions to have multiple labels, this is a multi-label classification task. Multiple labels typically correspond to a single path in the tree (from root to a leaf or internal node).

An important aspect of context-dependent fine grained entity type classification is that mentions of an entity can have different types depending on the context. Consider the following example: *Madonna starred as Breathless Mahoney in the film Dick Tracy*. In this context, the most appropriate label for the mention *Madonna* is `actress`, since the sentence talks about her role in a film. In the majority of other cases, *Madonna* is likely to be labeled as a `musician`.

The main difficulty in fine grained entity type classification is the absence of labeled training examples. Training data is typically generated automatically (e.g. by mapping Freebase labels of resolved entities), without taking context into account, so it is common for mentions to have noisy labels. In our example, the labels for the mention *Madonna* would include `musician`, `actress`, `author`, and potentially others, even though not all of these labels apply here. Ideally, a fine grained type classification system should be robust to such noisy training data, as well as capable of exploiting relationships between labels during learning. We describe a model that uses a ranking loss—which tends to be more robust to label noise—and that learns a joint representation of features and labels, which allows for information sharing among related labels.[1] A related idea to learn output representations for multiclass document classification and part-of-speech tagging was considered in Srikumar and Manning (2014). We show that it outperforms state-of-the-art methods on two fine grained entity-classification benchmarks. We also evaluate our model on standard coarse type classification and find that training embedding models on all fine grained labels gives better results than training it on just the coarse

[1] Turian et al. (2010), Collobert et al. (2011), and Qi et al. (2014) consider representation learning for *coarse* label named entity recognition.

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers), pages 291–296,
Beijing, China, July 26-31, 2015.

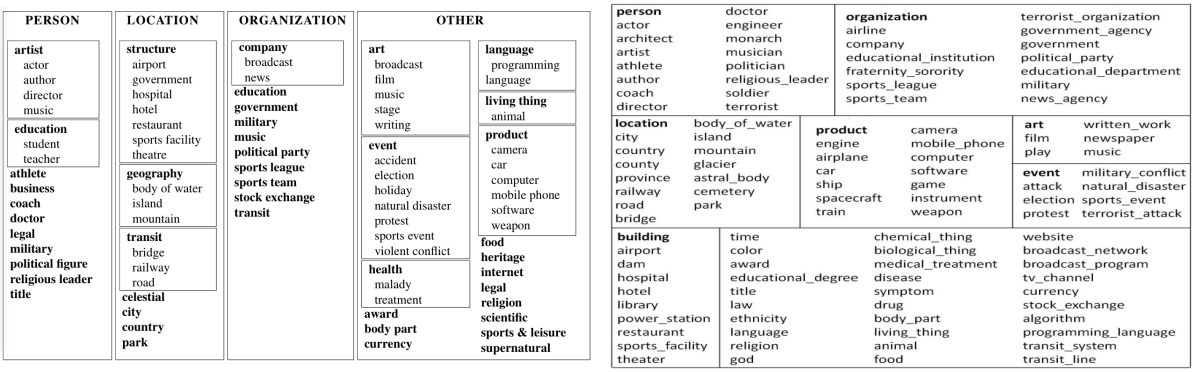

Figure 1: Label sets for Gillick et al. (2014)—left, GFT—and Ling and Weld (2012)—right, FIGER.

types of interest.

2 Models

In this section, we describe our approach, which is based on the WSABIE (Weston et al., 2011) model.

Notation We use lower case letters to denote variables, bold lower case letters to denote vectors, and bold upper case letters to denote matrices. Let $\mathbf{x} \in \mathbb{R}^D$ be the feature vector for a mention, where D is the number of features and x_d is the value of the d-th feature. Let $\mathbf{y} \in \{0,1\}^T$ be the corresponding binary label vector, where T is the number of labels. $y_t = 1$ if and only if the mention is of type t. We use $\mathbf{y}_t$ to denote a one-hot binary vector of size T, where $y_t = 1$ and all other entries are zero.

Model To leverage the relationships among the fine grained labels, we would like a model that can learn an embedding space for labels. Our model, based on WSABIE, learns to map both feature vectors and labels to a low dimensional space $\mathbb{R}^H$ (H is the embedding dimension size) such that each instance is close to its label(s) in this space; see Figure 2 for an illustration. Relationships between labels are captured by their distances in the embedded space: co-occurring labels tend to be closer, whereas mutually exclusive labels are further apart.

Formally, we are interested in learning the mapping functions:

$$f(\mathbf{x}) : \mathbb{R}^D \rightarrow \mathbb{R}^H$$
$$\forall t \in \{1, 2, \ldots, T\}, g(\mathbf{y}_t) : \{0,1\}^T \rightarrow \mathbb{R}^H$$

In this work, we parameterize them as linear functions $f(\mathbf{x}, \mathbf{A}) = \mathbf{A}\mathbf{x}$ and $g(\mathbf{y}_t, \mathbf{B}) = \mathbf{B}\mathbf{y}_t$, where $\mathbf{A} \in \mathbb{R}^{H \times D}$ and $\mathbf{B} \in \mathbb{R}^{H \times T}$ are parameters.

The score of a label t (represented as a one-hot label vector $\mathbf{y}_t$) and a feature vector $\mathbf{x}$ is the dot product between their embeddings:

$$s(\mathbf{x}, \mathbf{y}_t; \mathbf{A}, \mathbf{B}) = f(\mathbf{x}, \mathbf{A}) \cdot g(\mathbf{y}_t, \mathbf{B}) = \mathbf{A}\mathbf{x} \cdot \mathbf{B}\mathbf{y}_t$$

For brevity, we denote this score by $s(\mathbf{x}, \mathbf{y}_t)$. Note that the total number of parameters is $(D+T) \times H$, which is typically less than the number of parameters in standard classification models that use regular conjunctions of input features with label classes (e.g., logistic regression) when $H < T$.

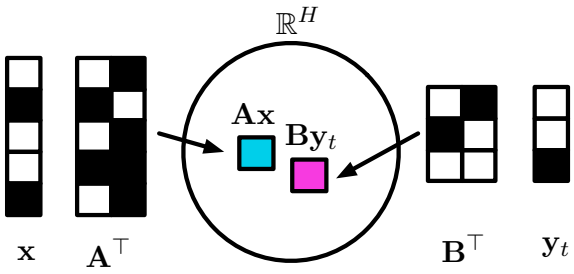

Figure 2: An illustration of the standard WSABIE model. $\mathbf{x}$ is the feature vector extracted from a mention, and $\mathbf{y}_t$ is its label. Here, black cells indicate non-zero and white cells indicate zero values. The parameters are matrices $\mathbf{A}$ and $\mathbf{B}$ which are used to map the feature vector $\mathbf{x}$ and the label vector $\mathbf{y}_t$ into an embedding space.

Learning Since we expect the training data to contain some extraneous labels, we use a ranking loss to encourage the model to place positive labels above negative labels without competing with each other. Let $\mathcal{Y}$ denote the set of positive labels for a mention, and let $\bar{\mathcal{Y}}$ denote its complement. Intuitively, we try to rank labels in $\mathcal{Y}$ higher than labels in $\bar{\mathcal{Y}}$. Specifically, we use the weighted approximate pairwise (WARP) loss of Weston et al. (2011). For a mention $\{\mathbf{x}, \mathbf{y}\}$, the WARP loss is:

$$\sum_{t \in \mathcal{Y}} \sum_{\bar{t} \in \bar{\mathcal{Y}}} \mathcal{R}(\text{rank}(\mathbf{x}, \mathbf{y}_t)) \max(1 - s(\mathbf{x}, \mathbf{y}_t) + s(\mathbf{x}, \mathbf{y}_{\bar{t}}), 0)$$

where $\text{rank}(\mathbf{x}, \mathbf{y}_t)$ is the margin-infused rank of label t: $\text{rank}(\mathbf{x}, \mathbf{y}_t) = \sum_{\bar{t} \in \bar{\mathcal{Y}}} \mathbb{I}(1 + s(\mathbf{x}, \mathbf{y}_{\bar{t}}) > s(\mathbf{x}, \mathbf{y}_t))$, $\mathcal{R}(\text{rank}(\mathbf{x}, \mathbf{y}_t))$ is a function that transforms this rank into a weight. In this work, since

each mention can have multiple positive labels, we choose to optimize precision at k by setting $\mathcal{R}(k) = \sum_{i=1}^{k} \frac{1}{i}$. Favoring precision over recall in fine grained entity type classification makes sense because if we are not certain about a particular fine grained label for a mention, we should use its ancestor label in the hierarchy.

In order to learn the parameters with this WARP loss, we use stochastic (sub)gradient descent.

Inference During inference, we consider the top-k predicted labels, where k is the maximum depth of the label hierarchy, and greedily remove labels that are not consistent with other labels (i.e., not on the same path of the tree). For example, if the (ordered) top-k labels are `person`, `artist`, and `location`, we output only `person` and `artist` as the predicted labels. We use a threshold δ such that $\hat{y}_t = 1$ if $s(\mathbf{x}, \mathbf{y}_t) > \delta$ and $\hat{y}_t = 0$ otherwise.

Kernel extension We extend the WSABIE model to include a weighting function between each feature and label, similar in spirit to Weston et al. (2014). Recall that the WSABIE scoring function is: $s(\mathbf{x}, \mathbf{y}_t) = \mathbf{A}\mathbf{x} \cdot \mathbf{B}\mathbf{y}_t = \sum_d (\mathbf{A}_d x_d)^\top \mathbf{B}_t$, where $\mathbf{A}_d$ and $\mathbf{B}_t$ denote the column vectors of $\mathbf{A}$ and $\mathbf{B}$. We can weight each (feature, label) pair by a kernel function prior to computing the embedding:

$$s(\mathbf{x}, \mathbf{y}_t) = \sum_d K_{d,t} (\mathbf{A}_d x_d)^\top \mathbf{B}_t,$$

where $\mathbf{K} \in \mathbb{R}^{D \times T}$ is the kernel matrix. We use a N-nearest neighbor kernel[2] and set $K_{d,t} = 1$ if $\mathbf{A}_d$ is one of N-nearest neighbors of the label vector $\mathbf{B}_t$, and $K_{d,t} = 0$ otherwise. In all our experiments, we set $N = 200$.

To incorporate the kernel weighting function, we only need to make minor modifications to the learning procedure. At every iteration, we first compute the similarity between each feature embedding and each label embedding. For each label t, we then set the kernel values for the N most similar features to 1, and the rest to 0 (update $\mathbf{K}$). We can then follow the learning algorithm for the standard WSABIE model described above. At inference time, we fix $\mathbf{K}$ so this extension is only slightly slower than the standard model.

[2]We explored various kernels in preliminary experiments and found that the nearest neighbor kernel performs the best.

The nearest-neighbor kernel introduces nonlinearities to the embedding model. It implicitly plays the role of a label-dependent feature selector, learning which features can interact with which labels and turns off potentially noisy features that are not in the relevant label's neighborhood.

3 Experiments

Setup and Baselines We evaluate our methods on two publicly available datasets that are manually annotated with gold labels for fine grained entity type classification: GFT (Google Fine Types; Gillick et al., 2014) and FIGER (Ling and Weld, 2012). On the GFT dataset, we compare with state-of-the-art baselines from Gillick et al. (2014): flat logistic regression (FLAT), an extension of multiclass logistic regression for multilabel classification problems; and multiple independent binary logistic regression (BINARY), one per label $t \in \{1, 2, \ldots, T\}$. On the FIGER dataset, we compare with a state-of-the-art baseline from Ling and Weld (2012).

We denote the standard embedding method by WSABIE and its extension by K-WSABIE. We fix our embedding size to $H = 50$. We report micro-averaged precision, recall, and F1-score for each of the competing methods (this is called *Loose Micro* by Ling and Weld). When development data is available, we use it to tune δ by optimizing F1-score.

Training data Because we have no manually annotated data, we create training data using the technique described in Gillick et al. (2014). A set of 133,000 news documents are automatically annotated by a parser, a mention chunker, and an entity resolver that assigns Freebase types to entites, which we map to fine grained labels. This approach results in approximately 3 million training examples which we use to train all the models evaluated below. The only difference between models trained for different tasks is the mapping from Freebase types. See Gillick et al. (2014) for details.

Table 1 lists the features we use—the same set as used by Gillick et al. (2014), and very similar to those used by Ling and Weld. String features are randomly hashed to a value in 0 to 999,999, which simplifies feature extraction and adds some additional regularization (Ganchev and Dredze, 2008).

Feature	Description	Example
Head	The syntactic head of the mention phrase	"Obama"
Non-head	Each non-head word in the mention phrase	"Barack", "H."
Cluster	Word cluster id for the head word	"59"
Characters	Each character trigram in the mention head	":ob", "oba", "bam", "ama", "ma:"
Shape	The word shape of the words in the mention phrase	"Aa A. Aa"
Role	Dependency label on the mention head	"subj"
Context	Words before and after the mention phrase	"B:who", "A:first"
Parent	The head's lexical parent in the dependency tree	"picked"
Topic	The most likely topic label for the document	"politics"

Table 1: List of features used in our experiments, similar to features in Gillick et al. (2014). Features are extracted from each mention. The example mention in context is ... *who Barack H. Obama first picked* ....

	GFT Dev	GFT Test	FIGER
Total mentions	6,380	11,324	778
at Level 1	3,934	7,975	568
at Level 2	2,215	2,994	210
at Level 3	251	335	–

Table 2: Mention counts in our datasets.

GFT evaluation There are $T = 86$ fine grained labels in the GFT dataset, as listed in Figure 1. The four top-level labels are: `person`, `location`, `organization`, and `other`; the remaining labels are subtypes of these labels. The maximum depth of a label is 3. We split the dataset into a development set (for tuning hyperparameters) and test set (see Table 2).

The overall experimental results are shown in Table 3. Embedding methods performed well. Both WSABIE and K-WSABIE outperformed the baselines by substantial margins in F1-score, though the advantage of the kernel version over the linear version is only marginally significant.

To visualize the learned embeddings, we project label embeddings down to two dimensions using PCA in Figure 3. Since there are only 4 top-level labels here, the fine grained labels are color-coded according to their top-level labels for readability. We can see that related labels are clustered together, and the four major clusters correspond to to the top-level labels. We note that these first two components only capture 14% of the total variance of the full 50-dimensional space.

Method	P	R	F1
FLAT	79.22	60.18	68.40
BINARY	80.05	62.20	70.01
WSABIE	**80.58**	66.20	72.68
K-WSABIE	80.11	**67.01**	**72.98**

Table 3: Precision (P), Recall (R), and F1-score on the GFT test dataset for four competing models. The improvements for WSABIE and K-WSABIE over both baselines are statistically significant ($p < 0.01$).

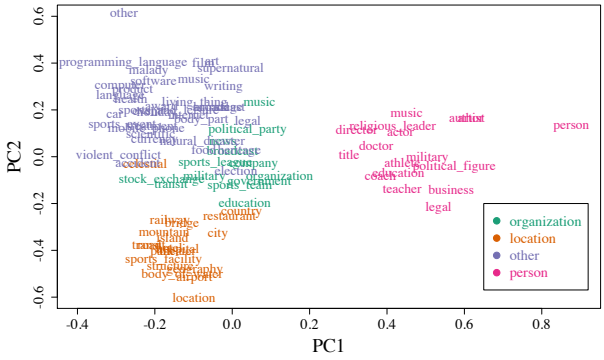

Figure 3: Two-dimensional projections of label embeddings for GFT dataset. See text for details.

FIGER evaluation Our second evaluation dataset is FIGER from Ling and Weld (2012). In this dataset, there are $T = 112$ labels organized in a two-level hierarchy; however, only 102 appear in our training data (see Figure 1, taken from their paper, for the complete set of labels). The training labels include 37 top-level labels (e.g., `person`, `location`, `product`, `art`, etc.) and 75 second-level labels (e.g., `actor`, `city`, `engine`, etc.) The FIGER dataset is much smaller than the GFT dataset (see Table 2).

Our experimental results are shown in Table 4. Again, K-WSABIE performed the best, followed by the standard WSABIE model. Both of these methods significantly outperformed Ling and Weld's best result.

Method	P	R	F1
Ling and Weld (2012)	–	–	69.30
WSABIE	81.85	63.75	71.68
K-WSABIE	**82.23**	**64.55**	**72.35**

Table 4: Precision (P), Recall (R), and F1-score on the FIGER dataset for three competing models. We took the F1 score from Ling and Weld's best result (no precision and recall numbers were reported). The improvements for WSABIE and K-WSABIE over the baseline are statistically significant ($p < 0.01$).

Feature learning We investigate whether having a large fine grained label space is helpful in learning a good representation for *feature vectors* (recall that WSABIE learns representations for both feature vectors and labels). We focus on the task of coarse type classification, where we want to classify a mention into one of the four top-level GFT labels. We fix the training mentions and learn WSABIE embeddings for feature vectors and labels by (1) training only on coarse labels and (2) training on all labels; we evaluate the models only on coarse labels. Training with all labels gives an improvement of about 2 points (F1 score) over training with just coarse labels, as shown in Table 5. This suggests that including additional subtype labels can help us learn better feature embeddings, even if we are not explicitly interested in the deeper labels.

Training labels	P	R	F1
Coarse labels only	82.41	77.87	80.07
All labels	85.18	79.28	82.12

Table 5: Comparison of two WSABIE models on coarse type classification for GFT. The first model only used coarse top-level labels, while the second model was trained on all 86 labels.

4 Discussion

Design of fine grained label hierarchy Results at different levels of the hierarchies in Table 6 show that it is more difficult to discriminate among deeper labels. However, it appears that the depth-2 FIGER types are easier to discriminate than the depth-2 (and depth-3) GFT labels. This may simply be an artifact of the very small FIGER dataset, but it suggests it may be worthwhile to flatten the `other` subtree ini GFT since many of its subtypes do not obviously share any information.

GFT	P	R	F1
LEVEL 1	85.22	80.55	82.82
LEVEL 2	56.02	37.14	44.67
LEVEL 3	65.12	7.89	14.07
FIGER	P	R	F1
LEVEL 1	82.82	70.42	76.12
LEVEL 2	68.28	47.14	55.77

Table 6: WSABIE model's Precision (P), Recall (R), and F1-score at each level of the label hierarchies for GFT (top) and FIGER (bottom).

5 Conclusion

We introduced embedding methods for fine grained entity type classifications that outperforms state-of-the-art methods on benchmark entity-classification datasets. We showed that these methods learned reasonable embeddings for fine-type labels which allowed information sharing across related labels.

Acknowledgements

We thank Andrew McCallum for helpful discussions and anonymous reviewers for feedback on an earlier draft of this paper.

References

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proc. of WSDM*.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program tasks, data, and evaluation. In *Proc. of LREC*.

Kuzman Ganchev and Mark Dredze. 2008. Small statistical models by random feature mixing. In *Proceedings of the ACL08 HLT Workshop on Mobile Language Processing*, pages 19–20.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. In *arXiv*.

Lynette Hirschman and Nancy Chinchor. 1997. MUC-7 named entity task definition. In *Proc. of MUC-7*.

Thomas Lin, Mausam, and Oren Etzioni. 2012. No noun phrase left behind: Detecting and typing unlinkable entities. In *Proc. of EMNLP-CoNLL*.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proc. of AAAI*.

Yanjun Qi, Sujatha Das G, Ronan Collobert, and Jason Weston. 2014. Deep learning for character-based information extraction. In *Proc. of ECIR*.

Marta Recasens, Marie-Catherine de Marneffe, and Christopher Potts. 2013. The life and death of discourse entities: Identifying singleton mentions. In *Proc. of NAACL*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: language-independent named entity recognition. In *Proc. of HLT-NAACL*.

Vivek Srikumar and Christopher D. Manning. 2014. Learning distributed representations for structured output prediction. In *Proc. of NIPS*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proc. of ACL*.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *Proc. of IJCAI*.

Jason Weston, Ron Weiss, and Hector Yee. 2014. Affinity weighted embedding. In *Proc. of ICML*.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2010. Collective cross-document relation extraction without labelled data. In *Proc. of EMNLP*.

Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. HYENA: Hierarchical type classification for entity names. In *Proc. of COLING*.