A Supplementary Materials

A.1 TypeNet Construction

Freebase type: musical_chord
Example entities: psalms_chord, power_chord harmonic_seventh_chord
chord.n.01: a straight line connecting two points on a curve
chord.n.02: a combination of three or more notes that blend harmoniously when sounded together
musical.n.01: a play or film whose action and dialogue is interspersed with singing and dancing

Table 9: Example given to TypeNet annotators. Here, the Freebase type to be linked is musical_chord. This type is annotated in Freebase belonging to the entities psalms_chord, harmonic_seventh_chord, and power_chord. Below the list of example entities are candidate WordNet synsets obtained by substring matching between the Freebase type and all WordNet synsets. The correctly aligned synset is chord.n.02 shown in bold.

A.2 Model Implementation Details

For all of our experiments, we use pretrained 300 dimensional word vectors from Pennington et al. (2014). These embeddings are fixed during training. The type vectors and entity vectors are all 300 dimensional vectors initialized using Glorot initialization (Glorot and Bengio, 2010). The number of negative links for hierarchical training $n \in \{16, 32, 64, 128, 256\}$.

For regularization, we use dropout (Srivastava et al., 2014) with p $\in \{0.5, 0.75, 0.8\}$ on the sentence encoder output and L2 regularize all learned parameters with $\lambda \in \{$1e-5, 5e-5, 1e-4$\}$. All our parameters are optimized using Adam (Kingma and Ba, 2014) with a learning rate of 0.001. We tune our hyper-parameters via grid search and early stopping on the development set.

A.3 FIGER Implementation Details

To train our models, we use the mention typing loss function defined in Section-5. For models with structure training, we additionally add in the hierarchical loss, along with a weight that is obtained by tuning on the dev set. We follow the same inference time procedure as Shimaoka et al. (2017) For each mention, we first assign the type with the largest probability according to the logits, and then assign additional types based on the condition that their corresponding probability be greater than 0.5.

A.4 Wikipedia Data and Implementation Details

At train time, each training example randomly samples an entity bag of 10 mentions. At test time we classify bags of 20 mentions of an entity. The dataset contains a total of 344,246 entities mapped to the 1081 Freebase types from TypeNet. We consider all sentences in Wikipedia between 10 and 50 tokens long. Tokenization and sentence splitting was performed using NLTK (Loper and Bird, 2002). From these sentences, we considered all entities annotated with a cross-link in Wikipedia that we could link to Freebase and assign types in TypeNet. We then split the data by entities into a 90-5-5 train, dev, test split.

A.5 UMLS Implementation details

We pre-process each string by lowercasing and removing stop words. We consider ngrams from size 1 to 5 and keep the top 100,000 features and the final vectors are L2 normalized. For each mention, In our experiments we consider the top 100 most similar entities as the candidate set.

A.5.1 Candidate Generation Details

Each mention and each canonical entity string in UMLS are mapped to TFIDF character ngram vectors. We pre-process each string by lowercasing and removing stop words. We consider ngrams from size 1 to 5 and keep the top 100,000 features and the final vectors are L2 normalized. For each mention, we calculate the cosine similarity, csim, between the mention string and each canonical entity string. In our experiments we consider the top 100 most similar entities as the candidate set.