Learning to Exploit Structured Resources for Lexical Inference
Supplementary Material

Vered Shwartz† **Omer Levy**† **Ido Dagan**† **Jacob Goldberger**§
† Computer Science Department, Bar-Ilan University
§ Faculty of Engineering, Bar-Ilan University
vered1986@gmail.com {omerlevy,dagan}@cs.biu.ac.il
jacob.goldberger@biu.ac.il

Weighted Edge Model (Section 4.2.1)

A typical neural network approach is to assign a *weight* w_i to each edge type e_i, where more indicative edge types should have higher values of w_i. We thus represent each path p as a bag of edges $\vec{\phi}$, a vector with an entry for each edge type denoting its frequency in p. To that end, we present a neural network that learns $\vec{w}$ by propagating the pair-level supervision through their paths to the edge types.

Classification We model the indicativeness of a path ($\hat{p}$) using logistic regression:[1]

$$\hat{p} \triangleq \sigma(\vec{w} \cdot \vec{\phi}) \quad (1)$$

We denote the label of a term-pair (x, y) as z. Since each term-pair is represented by multiple paths, we use the same assumption as the binary model: if there is at least one indicative path between x and y, then $x\mathcal{R}y$. We model the probability of a pair being positive as the score of its most indicative path, by max-pooling:[2]

$$\hat{z} \triangleq P(z = 1) = \max\{\hat{p} \,|p \in paths(x, y)\} \quad (2)$$

A term-pair is classified as positive if $\hat{z} \geq 0.5$.

Training We learn $\vec{w}$ by optimizing F_β (subject to L_2 regularization), rather than the standard accuracy-oriented likelihood objective. Therefore, we define our global objective as follows, similar to Jansche's (2005) derivation of F_β-optimized logistic regression:

$$L = \frac{TP}{\alpha(TP+FP)+(1-\alpha)(TP+FN)} = \frac{\sum_i \hat{z}_i z_i}{\alpha \sum_i \hat{z}_i + (1-\alpha)\sum_i z_i} \quad (3)$$

where TP, FP, FN are the numbers of true-positives, false-positives, and false-negatives respectively; z_i is the label of term-pair i in the training set; and $\alpha = 1/(1+\beta^2)$. We use back-propagation to derive the update rule and apply gradient ascent to learn $\vec{w}$.

Algorithm 2 Weighted Model Update Rule

1: **function** CALCULATEUPDATE($\vec{w}, \Phi, \vec{z}$)
2: **for** $z_i \in \vec{z}$ **do**
3: $best_i = \arg\max_{j \in paths(i)} \vec{w} \cdot \Phi_j$
4: $\vec{\phi}_i = \Phi_{best_i}$
5: $\hat{z}_i = \sigma(\vec{w} \cdot \vec{\phi}_i)$
6: $d_i = \frac{\partial \hat{z}_i}{\partial \vec{w}} = \hat{z}_i \cdot (1 - \hat{z}_i)\vec{\phi}_i$
7: $h = \alpha \sum_i \hat{z}_i + (1-\alpha)\sum_i z_i$
8: $\Delta\vec{w} = \sum_i z_i d_i / h - \alpha \sum_i z_i \hat{z}_i \cdot \sum_i d_i / h^2$
9: **return** $\Delta\vec{w}$

Updating requires the paths' features, given in the matrix Φ. Max-pooling selects the most indicative path for each term-pair z_i using the current weights, and uses their features to calculate the update $\Delta\vec{w}$ according to the F_β objective.

Algorithm 2 explains how the update $\Delta\vec{w}$ is calculated, given the current weight vector $\vec{w}$, path features Φ, and pair labels $\vec{z}$. Computing the derivation of the objective function F_β (line 8) requires the probability of each term-pair $\hat{z}_i$ (line 5) and its partial derivative by $\vec{w}$, d_i (line 6). To compute these, we select the most indicative path for each term-pair (line 3). This is where our algorithm differs from Jansche's (2005) derivation; while Jansche considers all instances (paths, in our case), we consider only the highest-scoring ones within each term-pair.

Following equation (2), the model assumes that a positive pair has at least one indicative path. This assumption does not model the frequent case where a positive pair has only non-indicative paths, due to lack of resource coverage. During training, the model is forced to se-

[1] As in standard logistic regression, we add a bias feature.

[2] We also experimented with a more traditional model that uses logistic regression (sigmoid over sum) instead of max-pooling. This model did not perform as well as the max-pooling model, and is therefore omitted.

lect a non-indicative path for each of these (positive) pairs, thus increasing the weights of non-indicative edges erroneously. We thus add an empty default path to each pair's representation; selecting the default prevents this erroneous update. The default path is represented by a zero-vector plus a bias feature.

Example Whitelist (Section 6.2)

Wikidata	
label_to_id$_{start}$	
altlabel$_{end}$	
id_to_label$_{end}$	
label_to_id	
p106c	(occupation)
p112c	(founder)
p136c	(genre)
p144c	(based on)
p156c	(followed by)
p21c	(sex or gender)
p22c	(father)
p279c	(subclass of)
p31c	(instance of)
p361c	(part of)
p364c	(original language of work)
p37c	(official language)
p413c	(position played on team)
p417c	(patron saint)
p50c	(author)
p527c	(has part)
p54c	(member of sports team)
p674c	(characters)
p6c	(head of government)
p735c	(given name)
p800c	(notable works)
Yago	
label_to_id$_{start}$	
id_to_label$_{end}$	
actedIn$_{end}$	
created	
hasGender	
influences	
isMarriedTo	
label_to_id	
rdfs:subClassOf	
dealsWith	
imports	
isAffiliatedTo	
isCitizenOf	
skos:prefLabel	
WordNet	
word_to_sense$_{start}$	
sense_to_word$_{end}$	
holonym	
instance_hyponym	
hypernym	

Table 1: An example whitelist for `proper2015`.

Table 1 displays a whitelist that performed exceptionally well on `proper2015`, with 97% precision and 29% recall. DBPedia edge types were not chosen by the model, probably because they were redundant on the training set, given Wikidata and Yago.

References

Martin Jansche. 2005. Maximum expected F-measure training of logistic regression models. In *EMNLP*.