Bake Off Results: NLI

Stephanie Wang

The task

Problem: Word-level natural language inference.

Training examples are pairs of words ***(wL,wR), y*** with y a relation in

- **synonym**: very roughly identical meanings; symmetric
- **hyponym**: e.g., *puppy* is a hyponym of *dog*
- **hypernym**: e.g., *dog* is a hypernym of *puppy*
- **antonym**: semantically opposed within a domain; symmetric

The goal: achieve the highest average F1 score on **word_disjoint**.

Weighted F1 was the wrong metric!

Chris's regret: allowing the bake-off to default to weighted F1 as the core metric.
Why? Clever modeling choices can lead to huge gains on the smallest classes, but these are hardly reflected in the final score if we micro-average.
Consider this hypothetical report where **antonym** goes from **0** to **0.50**:

	precision	recall	f1-score	support
antonym	0.00	0.00	**0.00**	150
hypernym	0.54	0.43	0.48	1594
hyponym	0.22	0.01	0.03	275
synonym	0.59	0.77	0.67	2229
avg / total	0.52	0.57	**0.53**	4248

	precision	recall	f1-score	support
antonym	0.45	0.55	**0.50**	150
hypernym	0.54	0.43	0.48	1594
hyponym	0.22	0.01	0.03	275
synonym	0.59	0.77	0.67	2229
avg / total	0.52	0.57	**0.55**	4248

Hardly any change to the final score! Unfair!

Weighted F1 was the wrong metric!

Suppose we had instead used **macro-averaging**. Then:

Baseline: 0.3

Hypothetical result with antonym at 0.50: 0.42

1st Place

kvchen, el168

F1: 0.62

“Used **200d GloVE vectors**, **inverted relationships between synonyms/antonyms**. Tried multiple approaches to augmenting dataset (inverting hypernyms/hyponyms, using all combinations of synonyms within a synonym "set", creating DAG for hyper/hyponyms for longer chained relationships) but these all lowered the F1 score.”

2nd Place

Jayadev,keshav2

F1: 0.6

"To represent words, we used word vectors described in Mrksic et al. 2016 (Counter-fitting Word Vectors to Linguistic Constraints), where they **apply antonymy and synonymy constraints on top of pretrained word vectors** (Source: https://github.com/nmrksic/counter-fitting).

To combine inputs, we **concatenated the sum and difference of their vector representations**. Given that the word vectors we used were expected to have synonyms clustered together, we thought that this would help distinguish synonyms/antonyms in different dimensions.

The model we used was a simple **neural network** with **one hidden layer of size 1000** (using sklearn's MLPClassifier).

One interesting feature of these word embeddings was the **large increase in F1 score for antonyms**, from ~0% (baseline) or ~10-15% (300d GloVe embeddings) to ~40%. "

3rd Place

Cholden

F1: 0.58

“I kept the majority of the baseline method, changing the **hidden dimensionality variable to 350**, and **max iterations to 650**. Additionally, I implemented combinations to **include commutative and transitive properties** as discussed in the Notes section. Otherwise, word representation remained the same.”

Vector combinations

- Concatenating
- Adding
- Subtracting
- Elementwise-multiplying
- Averaging

Data Augmentation

- “adding hyponyms of the form [[A,C], 'hyponym'] when we found the training corpus to have [[A,B], 'hyponym'] and [[B,C], 'hyponym']”
- “using commutativity of the synonym and antonym relations and by adding (x, y), 'hypernym' if (y, x), 'hyponym' is in the data (and vice versa)”
- “If A is a hyponym to B, and B is a hyponym to C, then A is a hyponym to C”
- “creating DAG for hyper/hyponyms for longer chained relationships”