Perturbation

Rik Bose

July 28, 2020

Contents

1 Perturbation Analysis

We test the robustness of the three parse-level similarity metrics – Trips-BLEU, SemBLEU, and Smatch – by generating random graphs and observing the change in similarity score over a sequence of perturbations. A perturbation represents a minimal unit of change between two parses:

- `NLABEL`: A changed node label
- `ELABEL`: A changed edge label
- `EADD`: An added edge
- `NADD`: An added node, implicitly with an added edge.

Labels are drawn from an *element space* defined by a closed set of labels and a similarity metric. We perform two types of label selection:

- `random`: Labels are selected uniformly randomly from the space.

- `modify(reference, threshold)`: We provide a reference label and threshold, θ, to select a label that is similar to the reference label.

Figure 1 demonstrates the possible labels drawn using each method in a simple label space. Note that =modify(reference, 1)

Element Space	selection	options
{1,2,3,4,5}	random	{1,2,3,4,5}
{1,2,3,4,5}	modify(3, 0.2)	{2,3,4}

Table 1: A simple *element space* consisting of 5 integers and normalized difference as similarity metric. Using `modify(3, 0.2)`, we select a random label from the set `{2,3,4}` instead.

1.1 Experimental Setup

A random graph generator is a tuple $\mathbf{Gr}(N, E, P)$ consisting of two element spaces, N for nodes and E for edges, and a probability distribution, P, over possible perturbations. We start the process by generating a random directed tree with k-nodes. We then draw n random perturbations from P and apply them to the graph. In order to apply a label-change operation (`NLABEL` or `ELABEL`) we first select a random source node or edge from the graph and use the `modify` operation to select a new label. The addition operations (`NADD` and `EADD`) are performed by selecting random source and target nodes and generating a random edge label from E. For `NADD`, the target is a a random node is generated from N instead.

The `modify` operation is essentially irrelevant to SemBLEU and Smatch since they both use exact-match to determine if two labels are are the same. A variant of the `NADD` operation, `CADD`, selects the label for the generated node using the `modify` operation applied to another randomly selected node from the graph. The purpose of `CADD` is to introduce more potential errors for TripsBLEU.

We use 3 different distributions of perturbations:

- `uniform`: All four basic perturbation operations are equally likely
- `relabel`: Use only the `NLABEL` and `ELABEL` operations. The structure of the graph does not change.
- `adding`: Use only `NADD` (or `CADD`) and `EADD` operations

1.2 Results

For each distribution, we generate 50 unique random graphs and apply 25 perturbations to each. We compute (1) `max-jump` - the largest single change in similarity from a perturbation and (2) `cut` the number of perturbations before a resulting graph's score against the original by more than a cutoff, ζ. That is, $cut_{\zeta} = min_i |sim(p_{i-1}, p_0) - sim(p_i, p_0)| > \zeta$

$\theta = 0.8$	$\zeta = 0.2$			
metric	P	max-jump	cut_{ζ}	# Cuts
SemBLEU		0.193	4.58	31
TripsBLEU	uniform	0.109	4.0	3
SMatch†		0.137	4.5	59
SemBLEU		0.268	0	0
TripsBLEU	relabel	0.159	0	0
SMatch†		0.163	6.17	68
SemBLEU		0.10	1.01	99
TripsBLEU	adding	0.10	0.10	99
SMatch†		0.119	0.19	100
SemBLEU		0.097	1	100
TripsBLEU	cadd	0.097	1	100
SMatch†		0.119	1.28	100

Table 2: This table shows results from the perturbation analysis

1.3 Discussion

Unlike TripsBLEU and SemBLEU, Smatch is unable to handle situations where a triple is duplicated. Since this is a corner case and the AMR specification does not allow for duplicated triples, we discard any duplicates.

2 Sentence-level Experiment

[?] performs a sentence level experiment to test the ability of structural similarity metrics to recreate human judgements. For each of 200 pairs of AMR parses, 3 human annotations are acquired to determine which of each pair of parses is most correct. The sentence-level experiment uses a structural similarity metric to compare each candidate parse against its respective gold parse. The metric is deemed correct if agrees with the most common human annotation.

2.1 Naive AMR label similarity

In order to adapt TripsBLEU's benefits to AMR, we use a simple similarity metric. Leveraging the naming convention of Propbank types we use Jaro-Winkler string similarity ([?]) to compare node labels. Jaro-Winkler similarity (extending Jaro similarity) was originally devised to identify misspellings in surnames. Observing that misspellings in surnames tend to occur towards the end of short strings, Jaro-Winkler places more weight on the longest prefix match of two strings. Hence, diffeerent senses of the same word will be scored as more similar than senses of different words. We apply a threshold of 0.8 to the similarity metric.

2.2 Results

Table 3 shows the results of the sentence level experiments. We observe a small improvement using a flexible matching for Propbank labels. This can be attributed to retaining partial edge-matches in situations where incorrect types were assigned but the argument structure was still correct. For example, if a node is assigned `make-01` where `make-02` is expected, SemBLEU would give a score of 0 to all ngrams containing `make-01`. Instead, TripsBLEU retains a partial match, preferring the correct type of `make-02` but still rewarding `make-01`.

This is similar in nature to the way Smatch's underlying alignment allows it to score structural similarity through a maximal alignment even when the node labels don't match.

metric	score
smatch	0.765
sembleu (n=3)	0.815
TripsBLEU (n=3)	0.830

Table 3: This table shows the results of the Sentence level experiments over 200 parse pairs

2.3 Future Work

There is much more work to be done in determining semantic similarity between Propbank types. In particular, the Jaro-Winkler similarity only really measures whether a node is attached to the correct parse-level predicate. It is, for example, unable to determine whether `make-01` is more similar to `make-02` or `make-03`. Future similarity metrics should take into account the argument structure and the expected types of elements filling the arguments.