Detecting the Presence of Punning

Simon Fraser University

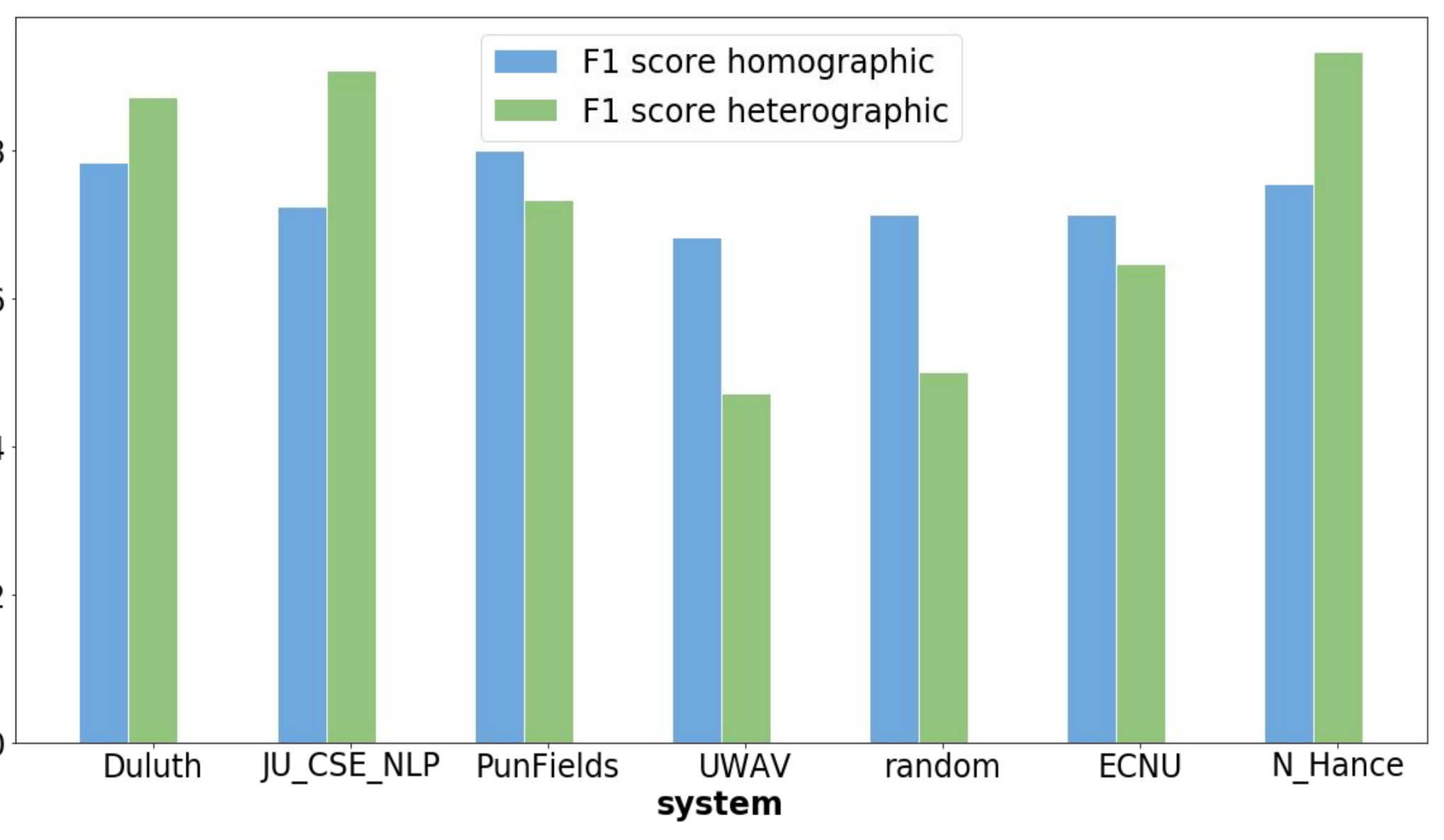
Kinnan Kwok, Rulai Hu

OVERVIEW

Detecting the presence of a pun is a non-trivial task for computers because it often depends on context and the ambiguity of phrases. We attempt to reproduce a result in the SemEval-2017 task 7 challenge, as well as improve one of the pun detection algorithms that participated in the competition.

Secondly, we aim to achieve similar or better results compared to systems participating in SemEval-2017 using a recurrent neural network.

STATE OF THE ART



Graph of F1-scores for various state of the art systems measured separately for homographic and heterographic datasets.

Algorithms for pun detection were comparatively reviewed in SemEval-2017. Systems such as Fermi, UWAV, ECNU and JU_CSE_NLP took a supervised approach.

ELiRF-UPV and ECNU leverages the WordNet synset database when determining whether a word has more than one sense in the context of a sentence (ie. it is a pun). Duluth also uses sense labels for pun detection. Of all the unsupervised algorithms, N-Hance performs the best relying on a minimal set of information.

A PUN IS...

For the purpose of pun detection in NLP, puns can be characterized as homographic or heterographic. The former is when a word is spelled the same but has two or more senses. The latter is when the word(s) are homophonic or *almost* a homophone with respect to another word.

Want to hear a joke about paper?
Nevermind, it's tearable.

An example of a heterographic pun on the word "terrible".

I used to be a banker but I lost interest.

An example of a homographic pun on the meaning of "interest".

A TOY EXAMPLE USING PMI

The shovel was a groundbreaking invention.

1. Ignoring stopwords, enumerate all the *forward pairs* in the sentence.
2. Score these pairs by PMI or one of its variants.

Word pair	PMI
shovel, groundbreaking	19.77
groundbreaking, invention	16.73
shovel, invention	10.73

3. Compare the difference between the max score and runner up given a median IQR = 2.33.
4. If the difference (19.77 - 16.73) is greater than the the median IQR, then classify this sentence as containing a pun.

TWO APPROACHES

Pointwise Mutual Information (N-Hance)

For our baseline, we chose N-Hance because it reportedly has very good overall precision, recall and accuracy across the board.

More importantly, the algorithm is fast, simple and unsupervised, and works on both homographic and heterographic puns. This makes it fast and easy to see if our improvements are working.

The definition of PMI is:

$$pmi(w1, w2) = log_2 \frac{p(w1, w2)}{p(w1)p(w2)}$$

We can compute the probability of bigrams and unigrams in a sentence with respect to either a frequency distribution or a smoothed distribution. Since N-Hance does not use smoothing, but rather combines their training and test set, we chose to augment their method with Jelinek-Mercer smoothing.

As for our lambdas, we chose values that maximize the difference between the highest PMI and the runner up for the pun dataset.

$$p_{JM}(w_i|w_{i-1}) = \lambda_1\, p(w_i|w_{i-1}) \;+\; (1-\lambda_1)\, p_{JM}(w_i)$$
$$p_{JM}(w) = \lambda_2\, p(w) \;+\; (1-\lambda_2)\, V^{-1}$$

After computing the frequency distribution, we compute the interquartile range (IQR) of the PMI scores for every sentence in the training set, then we compare the highest PMI scoring pair with the the median of the IQRs.

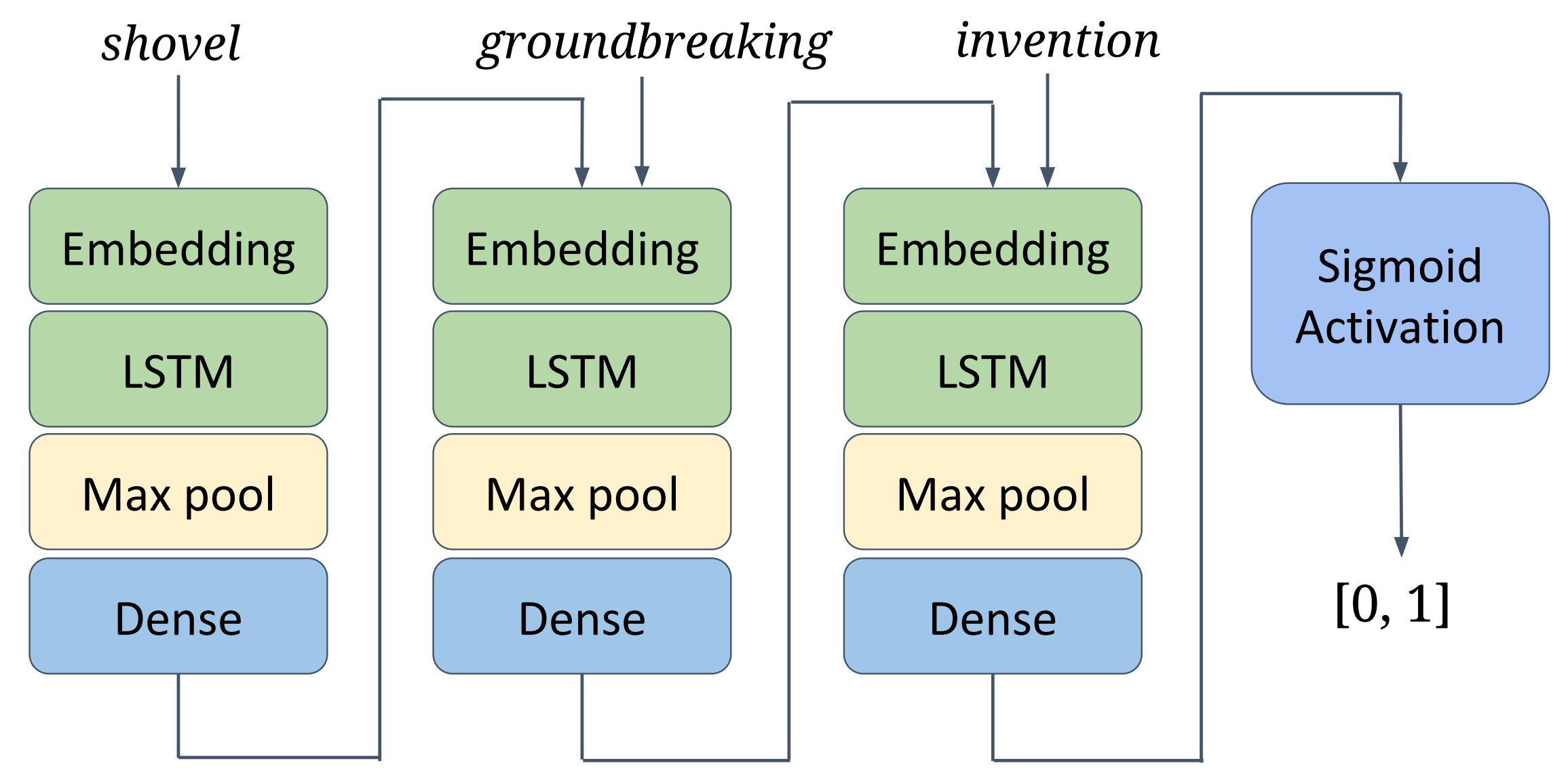
Window size

For generating the collocations, we chose an arbitrary window size of 20. We found that most sentences containing puns are 20 words or less, so this will catch most of the word pairs in our pun dataset.

VS

Recurrent Neural Networks with LSTM units

Another approach is to do binary classification using a neural network. A corpus of 913 puns and 1000 non-pun sentences was used to train a RNN. Sentences were vectorized using Keras' tokenizer module.



- **Embedding layer** learns weights that represent word relationships.
- **LSTM layer** learns words from previous time steps and decides whether they are retained or forgotten.
- **Max Pooling layer** reduces overfitting by sampling the max values from a quadrant in the tokenized text.
- **Dense layer** is a sigmoid activation layer that outputs a probability that represents whether the input is a pun or not.

By feeding the corpus with correct labels into the network, the system is able to adjust and learn weights using back propagation through time.

Upon training completion, a new sequence of words is fed into the network and a prediction is obtained. Values closer to 1 mean that the sentence is highly likely to be a pun, and values closer to zero means the sentence is not a pun.

RESULTS

Our implementation of N-Hance with smoothing and the modified scoring function did not perform as well as the N-Hance model in the paper. This could be a factor from using a different dataset. Here are the first few rows of their example:

Word pair	PMI - N-Hance paper	PMI - Our attempt
toaster, toasted	11.6896	12.9301
threw, toasted	7.9549	8.6081
threw, toaster	7.8618	8.9301
inventor, toasted	7.4851	9.7206
party, inventor	-1.1516	2.9724

The spread of PMI values is not as dramatic in our implementation as compared with the paper. The word pair (toaster, toasted) is barely an outlier which means the classification is less sure.

The F1-score from the RNN is higher than from N-Hance, however the scores may not be comparable, because the datasets we used were different. We can compare the results to our modified model of N-Hance and can see that the improvements are significant.

Metric	N-Hance	Modified N-Hance	RNN
Accuracy	0.7364	0.5266	0.8761
Precision	0.7553	0.8321	0.9627
Recall	0.9334	0.5228	0.8165
F1-score	0.8350	0.6422	0.8836

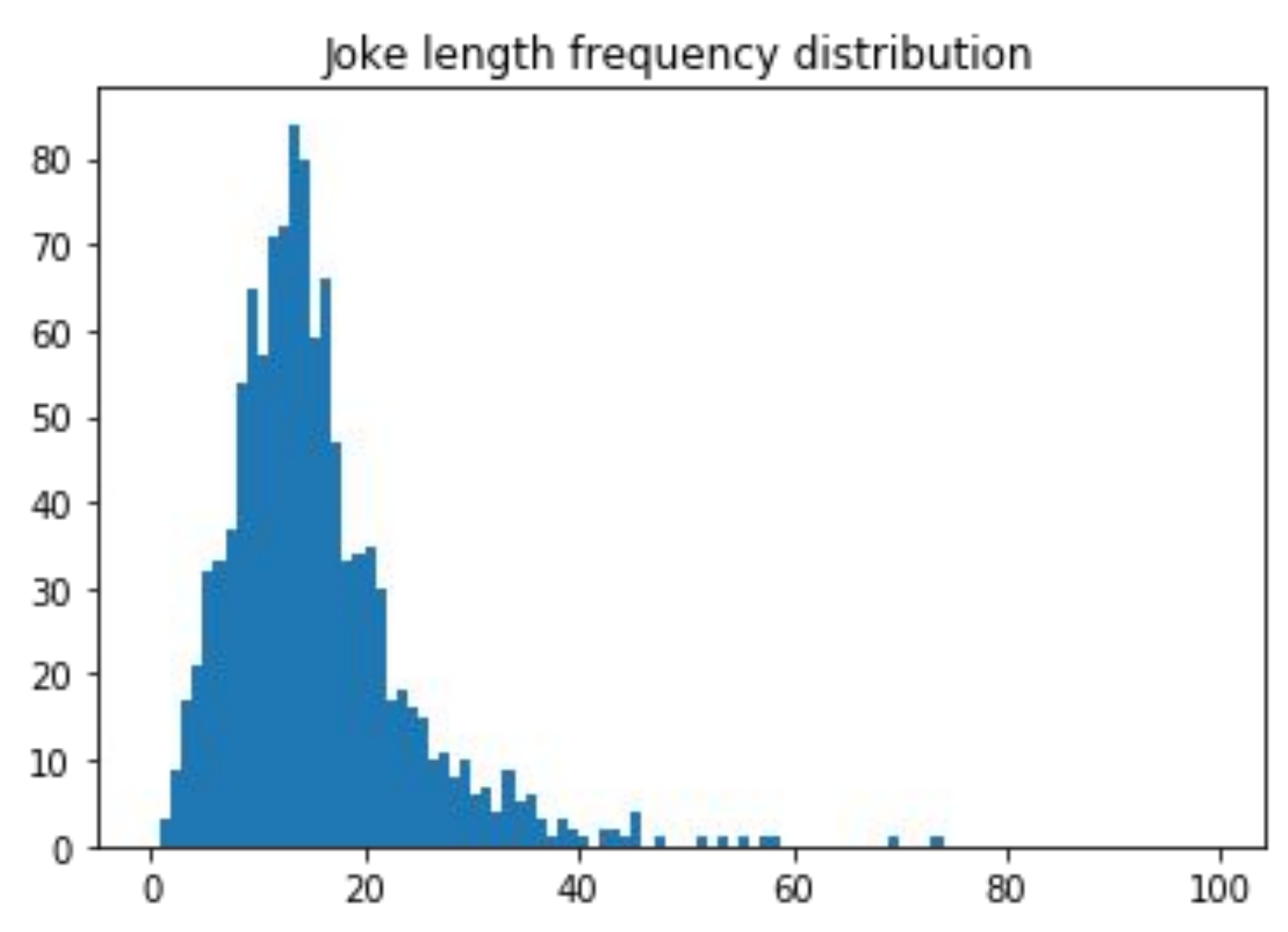
LIMITATIONS OF THE MODEL

N-Hance (PMI + Smoothing)

The performance of N-Hance assumes that for most puns there exists a word with high semantic or phonetic association with the punned word (eg. banker, interest).

We found in practice that the median IQR, typically in the range of 2 to 4 is too large to be a reasonable threshold. For example, a threshold value of 0.5 yields much better performance. More sophisticated outlier detection to catch pairs at the borderline should yield better results.

Due to finite window size, the model is unable to capture puns that have long distances between the pairs.



Recurrent Neural Network

The learned weights from the model are highly dependent on the dataset. Since we had a small dataset to work with, our results may have fitted well to this particular distribution of puns that may not represent puns in the real world.

Some Examples of Input	Prediction	Analysis
Why do people litter? Because they don't take the litter signs litterally.	0.999	Good
The embedding layer learns weights that represent word relationships.	0.004	Good
I ordered 2000 lbs of chinese soup. It was won ton.	0.022	Poor
The key to a good mailman joke is in its delivery.	0.457	Poor