Summary of "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank" paper

RNTN.md

Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank

Introduction

- The paper introduces:
  - Sentiment Sentiment Treebank - A dataset containing 215,154 phrases with fine-grained sentiment labels (5 classes).
  - Recursive Neural Tensor Network - Model to learn these fine-grained sentiment labels.
- Link to the paper

Sentiment Sentiment Treebank

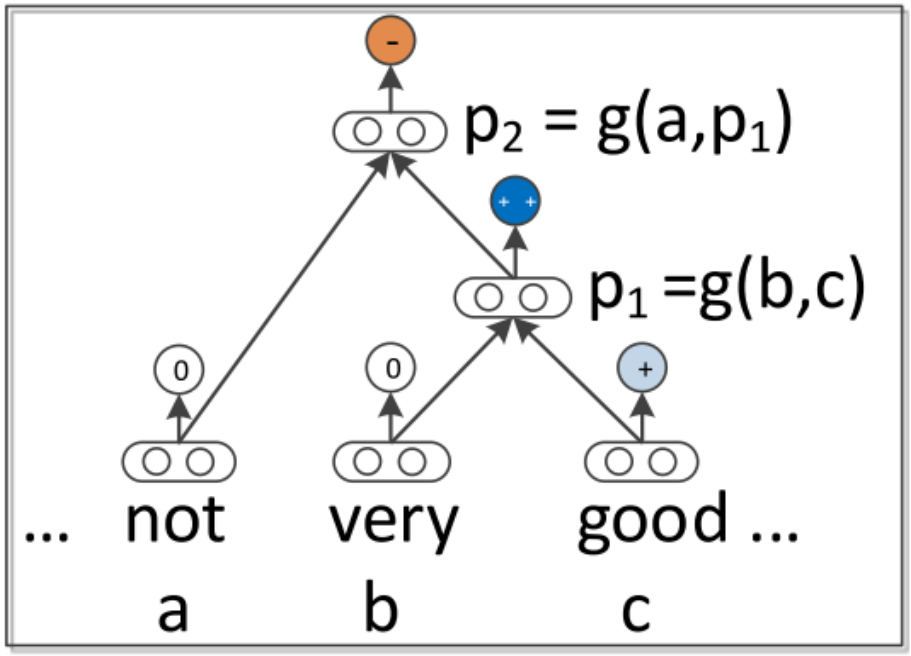
- Corpus of 11,855 sentences with fully labelled parse trees.
- Can be used to analyse the compositional effects of sentiment in language.
- Start with movie reviews dataset, normalise and parse sentences, and label using crowdsourcing on Amazon Mechanical Turk.

Observations

- Majority of shorter phrases are neutral while longer phrases have stronger sentiments.
- A 5-class classification is sufficient to capture the variability of sentiments.

Recursive Neural Models

- Parse a given n-gram into a binary tree and represent each word (corresponding to leaves in the tree) using a *d*-dimensional vector.
- Compute parent vectors using a bottom-up approach using different composition functions.
- To start with, word vectors are initialized randomly from a uniform distribution.
- For classification task, use the compositions word vectors as input for the softmax.
- Different models differ in terms of how word vectors are combined together as shown in the figure.



RNN: Recursive Neural Network

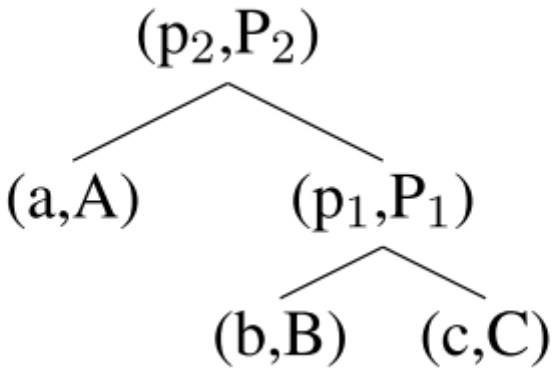
- Uses the equation shown in the figure

$$p_1 = f\left(W\begin{bmatrix} b \\ c \end{bmatrix}\right), p_2 = f\left(W\begin{bmatrix} a \\ p_1 \end{bmatrix}\right),$$

- *f = tanh*
- *W* is the weight matrix to be learnt.
- Con
  - Input vectors interact only implicitly, via the nonlinear *tanh* function.

MV-RNN: Matrix-Vector RNN

- Represent every word and phrase as both a vector and a matrix.
- Matrix for each word is initialized as identity matrix plus a small Gaussian noise.
- For the parse tree as



the equation used is

$$p_1 = f\left(W\begin{bmatrix} Cb \\ Bc \end{bmatrix}\right), P_1 = f\left(W_M\begin{bmatrix} B \\ C \end{bmatrix}\right),$$

- W and W_M are both learnt.
- Con
  - Number of parameters depend on the size of vocabulary and could be very large.

RNTN: Recursive Neural Tensor Network

- Equations

$$p_1 = f\left(\begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]}\begin{bmatrix} b \\ c \end{bmatrix} + W\begin{bmatrix} b \\ c \end{bmatrix}\right),$$

$$p_2 = f\left(\begin{bmatrix} a \\ p_1 \end{bmatrix}^T V^{[1:d]}\begin{bmatrix} a \\ p_1 \end{bmatrix} + W\begin{bmatrix} a \\ p_1 \end{bmatrix}\right).$$

- V is the tensor that defines multiple bilinear forms.

- Each slice of the tensor *V* can be interpreted as capturing a specific type of composition.

Observations

- Models compared with
    - Naive Bayes - NB
    - SVMs
    - Naive Bayes with bag of bigram features - biNB
    - Average neural word vectors (ignoring word order) - vecAvg
- Task
    - Fine-grained Sentiment For All Phrases
        - RNTN > MV-RNN > RNN > other models.
    - Full Sentence Binary Sentiment
        - RNTN pushes the state of the art on short phrases to 85.4%
    - Contrastive Conjunction
        - Sentences of the form *X but Y*
        - RNTN > MV-RNN > RNN > SVM
    - RNTN outperforms other models in special cases like where a positive sentence is negated or where a negative sentence is negated to make it less negative (not positive though). This suggests that RNTN could capture the effect of negative words in both positive and negative sentiment sentences.

Notes

- The optimal word vector size reported by the paper was between 25 and 35 and these word vectors were trained as part of sentiment tagging process. It would be interesting to see how are these results affected by using word vectors from say Glove which may or may not be fine tuned for the sentiment labelling.