We consider the general problem of sentence similarity, which we treat as a textual classification problem. This has application in a number of fields: comment deanonymization, intent recognition, chat-bots, and web parsing.

**Problem**

• We assume that given two wine reviews, we return a prediction/probability of whether they’re about the same wine
• This is easier if we have all the labels for the possible wines, since we can predict for each label.
• Accuracy also varies based off whether the wine categories are included in the training (interpolating), or are not (extrapolating).

**Data**

Input data:
- Wine Review Dataset from Kaggle [1]
- >100,000 reviews of wines

Pre-processing:
- Restrict to the top 50 most common wine categories, each of which has over 200 reviews.
- We remove words that may make the task too easy—like label names.
- Somewhat balance the dataset.

Final dataset:
- 100,000 comments, which are fairly evenly distributed among the top.
- We sample random pairs, so interpolate.

**Introduction**

- **Word Embeddings**
  - We use word embeddings as our core feature extraction method.
  - We create 300-dimensional vectors representing each word.
  - 300 dimensions was chosen as this is industry-standard.
  - We try pre-trained GLoVe and word2vec vectors, and word2vec vectors trained our data with a CBOW model.

\[
\text{Word Vectors} = \begin{bmatrix} \text{dilute (1:100)} \\ \text{tropical (4:1082)} \\ \text{aromas (1:1021)} \\ \text{of (1:10)} \\ \text{banana (1:233)} \end{bmatrix}
\]

**Setting 1**

Problem definition:
- We can enumerate all the wine categories.
- We train a model to predict a single wine category, and then predict probability using its outputs.

Methodology:
- We use this neural network architecture, with cross-entropy loss.

\[
\text{Review 1:} \begin{bmatrix} \text{Word Embedding Layer} \\ \text{Optimal Dropout/ MaxPool/ Convolution Layers} \\ \text{LSTM Layer} \\ \text{Drop Out Layer} \end{bmatrix} \rightarrow \text{Probability for each category}
\]

\[
\text{Review 2:} \begin{bmatrix} \text{Word Embedding Layer} \\ \text{Optimal Dropout/ MaxPool/ Convolution Layers} \\ \text{LSTM Layer} \end{bmatrix} \rightarrow \text{Binary Prediction}
\]

- We also implement simple baselines, simply averaging word vectors (a “bag of words”) model to turn sentences into features. These include:
  - Support Vector Machines with RBF Kernel.
  - Multiple Logistic Regression Models.

\[
\text{Attention} = \text{tanh}(W_w \cdot h_t + b_w)
\]

**Setting 2**

Problem definition:
- We cannot enumerate all categories.
- We simply train a model to compare pairs of reviews—a type of “one-shot” learning.

Architecture:
- We use the following Siamese architecture, with a shared model for the branches.

\[
\text{Distance} = \frac{1}{| \text{positive pairs} |} \sum \text{distance}(u_i, u_j)
\]

**Results**

We look at F1 score of the binary classifier—directly comparable to multi-class accuracy for Setting 1)—random baseline is 10%

<table>
<thead>
<tr>
<th>Setting 1</th>
<th>Setting 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLoVe</td>
<td>Word2Vec</td>
</tr>
<tr>
<td>Linear</td>
<td>SVM</td>
</tr>
<tr>
<td>55.2%</td>
<td>88.8%</td>
</tr>
<tr>
<td>48.8%</td>
<td>73.0%</td>
</tr>
<tr>
<td>36.8%</td>
<td>66.4%</td>
</tr>
<tr>
<td>30.6%</td>
<td>70.0%</td>
</tr>
</tbody>
</table>

**Future Work**

- Much better GPUs are needed to train such a large model.
- Siamese Model fails to generalise in a fairly small number of epochs—it may have too many parameters.
- Change dataset to be more relevant to sentence similarity; we realised that a lot of reviews are not similar sentences.

**Discussion**

Significant success when we have all labels
- Addition of convolutional layers allows it to generalise better
- Accuracy significantly above baseline

One-shot learning is significantly harder
- Unbalanced dataset (many more dissimilar pairs than similar pairs)
- Model is very slow to train, due to large dataset and parameter count

**References**