Generating Structured Opinion Sets in Unstructured Financial Texts

Lindsay Willmore linzwill@stanford
Diego Represas diegorep@stanford
David Dindi ddindi@stanford

Abstract

Analysts and researchers in the finance industry browse through hundreds of reports on a daily basis to extract quantitative data they can utilize to construct good investment models. The time consumed manually finding and compiling relevant data for these models represents a sizable proportion of each analyst's working time and could be greatly aided by full or semi automation. To this end, we are developing an information retrieval strategy using Neural Network models for Natural Language Processing (NLP) that can automatically find, categorize and aggregate in-line data-points that deal with a certain topic. In this paper, we explore the use of Tree-LSTMs and clustering models to rate the similarity between data-rich sentences. Finally, we introduce a hybrid, 2-armed bandit-like augmented Tree-LSTM method that outperformed all others at our task.

1 Introduction

As of today, financial analysts and researchers spend a considerable amount of time browsing through hundreds of publications each day in order to retrieve data points they can utilize to construct their investment models. Although most of the data points commonly used by finance professionals are structured and indexed, there are some, still, whose publication environment and niche audience have made them unattractive options for companies wishing to structure data. Consequently, it has become the job of financial analysts to manually scavenge for this data and present it in a structured format.

A number of these yet unstructured data points present themselves in the form of in-line numbers within natural language. For example, a research report might have a sentence saying: "we predict that the EPS of XYZ equity will increase 7% over the month of September." A complete information retrieval task would be to identify what index the sentence is talking about, the equity it concerns, the value of the index and the value's time frame. This task could then be repeated for every in-line number across every publication available. However an easier intermediary step could be be to group together all the sentences that contain an in-line value and have similar semantic content. The grouped and similarity-ranked sentences might not be as desirable as a fully structured table, but they would still aid the financial researcher tremendously.

We explore the performance of "traditional" similarity ranking models in this task and compare it to a recently developed (Tai et. al. 2015) Neural Network model defined as a Tree Long Short-Term Memory (Tree-LSTM). The Tree-LSTM is a Recurrent Neural Network designed to best perform with data characterized by tree-structured network topologies, such as Natural Language. Tree-LSTMs have been shown to outperform all existing systems in predicting the semantic relatedness of two sentences in the SemEval data set. The "traditional" models we compare the Tree-LSTM's performance to consist of non-neural network models often used to measure semantic similarity in industry, such as Bag Of Words (BOW) models and sequential Tree-LSTM models.

2 Models and Background

Given our task of assessing semantic similarity between sentences, we explored the use of four models of increasing sophistication.

2.1 Word Vector Representations

Continuous Bag of Words CBOW is a computationally efficient algorithm introduce by Mikolov et al. (2013) for learning continuous word representations. Provided with a local window of words (the, child, over, the, wall) CBOW predicts the center word (climbed) such that the probability P (center word $\|$ context words) is maximized.

Continuous Skip-gram The continuous skip gram is an analogous algorithm to CBOW, also introduced by Mikolov et.al. The main difference between the two models is that Skip-gram performs the reverse prediction of CBOW. Given a center word (climbed), the Skip-gram model predicts the surrounding context words (the, child, over, the, wall) such that the probability P (context words $\|$ center word) is maximized.

Global Vectors for Word Representations GloVe is a framework introduced by Pennington et al. (2014) that efficiently employs both global matrix factorization methods and local context window methods. The appeal of the GloVe framework lies in its capacity not only to capture syntactic and semantic relationships, but also to spatially encode these relationships in a meaningful manner. GloVe generates continuous word vector representations using word-word co-occurrence statistics over the entire corpus.

2.2 LSTM

The Long Short-Term Memory model is currently a popular choice for sequence-based tasks in NLP. LSTM gates are applied at each activation unit to preserve long-range memory across our input sequence. The mathematical steps that make up the LSTM unit are shown in Figure 1. Five stages precede the final calculation of each hidden unit in network. First, the current word and previous hidden state are linearly combined and non-linearly transformed to create the **New memory cell**. Two gates, the **Input gate** and **Forget gate** are then calculated: the Input gate governs how important the current word is and the Forget gate indicates how much of the previous state must be remembered or forgotten. Finally the **Final memory cell** is generated by filtering the New memory cell from the current state through the Input gate and the New memory cell from the previous state through the Forget gate. Then a final **Output/Exposure Gate** is generated through which the final memory cell is passed to determine which information to preserve in that unit's hidden state.

$$
\begin{aligned}
i^{(t)} &= \sigma(W^{(i)}x^{(t)} + U^{(i)}h^{(t-1)}) && \text{(Input gate)} \\
f^{(t)} &= \sigma(W^{(f)}x^{(t)} + U^{(f)}h^{(t-1)}) && \text{(Forget gate)} \\
o^{(t)} &= \sigma(W^{(o)}x^{(t)} + U^{(o)}h^{(t-1)}) && \text{(Output/Exposure gate)} \\
\tilde{c}^{(t)} &= \tanh(W^{(c)}x^{(t)} + U^{(c)}h^{(t-1)}) && \text{(New memory cell)} \\
c^{(t)} &= f^{(t)} \circ \tilde{c}^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} && \text{(Final memory cell)} \\
h^{(t)} &= o^{(t)} \circ \tanh(c^{(t)})
\end{aligned}
$$

Figure 1: Mathematical equations describing the LSTM unit.

This activation unit is traditionally applied to recurrent neural networks (RNNs) and its variants which include bi-directional and multi-layer RNNs. We follow Tai et al. (2015) in evaluating the performance of these models in addition to the state of the art Dependency Tree-LSTM described below.

2.3 Dependency Tree-LSTM

While RNNs are strictly linear in the connectivity of their hidden units, Tree-LSTM nodes are connected in tree structures with a root node and subsequent branches of an arbitrary number. Dependency Tree-LSTMs derive their tree structure from the dependency parse of the sentence or sequence of words being evaluated. See Figure 2 for an example of a dependency tree. These trees are useful for capturing the syntactic structure of a sentence, structure that is helpful for determining semantic meaning but cannot be captured by sequential models.

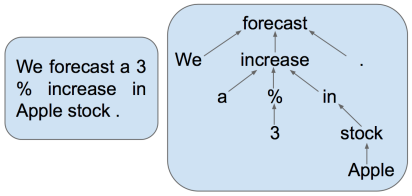

Figure 2: Dependency parse of an example sentence.

As each hidden layer is now connected to each of its child layers (as opposed to the single previous hidden layer in an RNN), the LSTM equations show in Figure 1 must be slightly altered. The amended equations shown below (Figure 3) are defined as the Child-Sum Tree-LSTM model. The changes from our original LSTM unit include summing the child hidden layers, calculating the forget gate for each child hidden state, and defining the Final memory cell as including the sum of each child's forget-gate-modified New memory cell.

$$
\begin{aligned}
h^{(t-1)} &= \sum_{k \in C(j)} h_k && \text{(sum the child states)} \\
f_k &= \sigma(W^{(f)} x^{(t)} + U^{(f)} h_k) && \text{(forget gate for each child)} \\
c^{(t)} &= i^{(t)} \circ \tilde{c}^{(t)} + \sum_{k \in C(j)} f_k \circ \tilde{c}_k && \text{(New memory cell has sum of child states)}
\end{aligned}
$$

Figure 3: Updated equations for Child-Sum Tree-LSTM model

The Dependency Tree-LSTM is used to create a vector representation of each sentence (root node hidden state), which, like the average CBOW vectors, can be compared pairwise to find semantic similarity. Kai et al. (2015) found that the optimal semantic similarity metric was a combination of distance and angle between sentence representations. We use the same in our Tree-LSTM approach and experiments below.

2.4 2-Armed Contextual Bandit

Whilst the Tree-LSTM is a powerful model that is able to capture indirect semantic relationships, it cannot reach an optimal performance if training data is insufficient or syntactically different than the data it will perform predictions in. The Tree-LSTM model may also under-perform other "traditional" models when the topology and structure of the data it analyzes is extremely simple.

Unlike movie reviews and other natural language data sets, where language can be highly subjective and context-based, the financial texts we examine tend to have a more direct structure and are often repetitive in nature. For example, a sentence where an analyst predicts the stock price for equity ABC in month Y will change very little from one publication to another or from one month to another. Consequently, the semantically similar sentence categories we examined were evidently split between non-subjective, syntactically similar sentences and sentences with varying syntax and structure. In this case, it would be ideal to have a system where the predictive engine can choose between models according to the structure of the sentences.

To this end, we developed a simple, 2-Armed Contextual Bandit algorithm for sentence similarity prediction. A *Multi-Armed Bandit* (Robbins 1952) can be defined as a set of real distributions $B = \{R_1, \ldots, R_K\}$, each distribution being associated with the rewards delivered by one of the $K \in \mathbb{Z}$ levers. Let $\mu_1, \ldots, \mu_K$ be the mean values associated with these reward distributions. The regret ρ after T rounds is then defined as the expected difference between the reward sum associated with an optimal strategy and the sum of the collected rewards: $\rho = T\mu^* - \sum_{t=1}^{T} \widehat{r}_t$, where μ^* is the maximal reward mean, $\mu^* = \max_k\{\mu_k\}$, and $\widehat{r}_t$ is the reward at time t (Mohri et al., 2005). The objective of the multi-armed bandit game is to, on each iteration, choose the lever $k \in K$ that will minimize regret.

In our case, the multi-armed bandit is just a two armed bandit that chooses between two levers: the Tree-LSTM model and the CBOW averaging technique. We expanded this bandit to be a *Contextual Bandit*, meaning that the distributions $B = \{R_1, \ldots, R_K\}$ are defined according to the context of a sentence. As of this writing, our "context" was simply the Jaccard distance between two sentences, albeit this context can be expanded into a vector of many features. We defined the maximal reward as the truth similarity value of our examples and the collected rewards as the sum of the similarity scores returned by the lever chosen on each turn. In short, our bandit works by choosing between the Tree-LSTM and the CBOW averaging technique according to the Jaccard Distance of the sentences it will try to predict the similarity of.

3 Approach and Experiments

3.1 Data and Pre-processing

Data In total we collected and cleaned 2800 investment banking reports containing 47K phrases and 200K unique tokens. Out of this cleaned set, 1088 sentence pairs were selected for manual scoring on the basis of structured semantic similarity. That data was split into 3 partitions: 628 training, 102 validation, and 358 testing. The validation and training partitions were combined into one large validation set where training was done on external data sources.

Ground Truth In order to evaluate the various methods against a standard baseline, a manually determined similarity was assigned to each of the 1088 sentence pairs mentioned above. The similarity was rated on a scale of 0 to 5 in increments of 0.5. Similarity was determined on the basis of comparing 5 categories, which included SUBJECT (company or financial entity), FIELD (financial indicator, data descriptor, e.g. 'price target', 'eps'), FIELD ATTRIBUTE (judgment made by source, e.g. 'forecast','estimate','model'), ACTION (increase, decrease, grow), and DATE (time frame: 'yoy','quarterly','may ####').

Pre-processing With the use of Stanford tokenizer, part-of-Speech tagger, and dependency parser, our data was prepared for evaluation on the various LSTM models presented in Kai et al. (2015).

Unless otherwise specified, the 703 sentences in our larger validation partition participated in the following experiments. Evaluative metrics included Pearson's correlation, Spearman's correlation, and mean squared error between ground truth and predicted similarity scores (all normalized to [0,1]).

3.2 Tuning Word Representations

We assessed the performance of the different frameworks outlined in section 2.1 (Table 1). For all three frameworks, we set the length of the word vectors to 300 and maintained a context window size of 5 words. Furthermore we applied a learning rate of 0.025, and trained on our full corpus for 10 epochs. Finally, we generated sentence representations by averaging the constituent word vectors. We found cosine similarity to be the optimal measure of similarity for the CBOW and Skip-gram models. For the GloVe model, we found a simple transformation of euclidean distance to be the optimal metric. Lastly, we optimized both CBOW and GloVe word vector representations. We found that a context window size of 9, and a word vector length of 400 were optimal for CBOW. Similarly, a context window size of 8, and word vector length of 600 were optimal for GloVe.

Word Vectors	Pearson's r	Spearman's ρ	MSE
GloVe	0.5027	0.4578	0.9579
Skip-gram	0.5596	0.5104	0.3941
CBOW	0.7961	0.5960	0.0416

Table 1: Word Vector Results

3.3 Tuning LSTM Hyperparameters

LSTM, bi-LSTM, and Tree-LSTM models were trained on the default hyperparameters settings of 150 dimensions in the LSTM memory layer, 1 layer for the LSTM and bi-LSTM, and training time of 10 epochs on the Sentences Involving Compositional Knowledge (SICK) dataset (Marelli et al., 2014) using Common Crawl GloVe word vectors. A comparison of these baseline models revealed that bi-LSTM and Tree-LSTM outperformed the standard LSTM model with Pearson's r of 0.573, 0.562, and 0.522 respectively. Then over the two better models, we optimized epoch training time. Figure 4 shows the performance of the Tree-LSTM model across varying lengths of training, with the optimum being reached at 7 epochs with an r of 0.616. The Bi-LSTM model (not shown) peaked at 13 epochs, with a slightly lower r of 0.602.

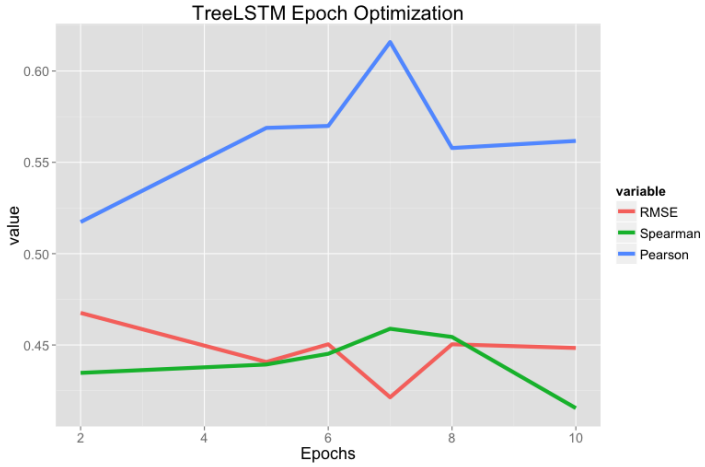

Figure 4: Tree-LSTM trained on the SICK data set, evaluated on the financial texts data. Pearson correlation value on the y, number of epochs on the x.

As the Tree-LSTM model was the highest performing when trained on the SICK data set, we evaluated the performance of this model when trained on our own data and learned CBOW vectors. Training on our smaller 628-word data split, we achieved meager results, as shown in Table 2.

4 Results and Discussion

4.1 Word Vectors Selection

Table 1 demonstrates that performance is strongly dependent on the word representation framework chosen. We observe that CBOW derived sentence representations perform better than their Skip-gram and GloVe counterparts. According to Mikolov et al. (2013), Skip-gram is most effective at capturing the meaning of rare words whereas CBOW is most effective at capturing the meaning of frequent words. Financial texts exhibit a low degree of linguistic redundancy as a result of the use of finance specific terminology and expressions. For instance, "price-target (projected price level of a share at a future date) will most often occur in context similar to the following,"We maintain a price-target of $4.50 for Q1.' Consequently, the semantic meaning of a sentence is strongly dependent

Method	Pearson's r	Spearman's ρ	MSE
GloVe Averaging	0.5656	0.4552	0.3316
GloVe Paragraph Vector	0.5791	0.4281	0.3250
CBOW Clustering	0.7671	0.5380	0.0975
CBOW Averaging	0.8313	0.6071	0.0330
LSTM (Kai, 2015)	0.5216	0.4051	0.2037
Bi-directional LSTM (Kai, 2015)	0.6014	0.4307	0.2065
Dependency - our CBOW word vectors	0.5065	0.2750	0.0781
Dependency Tree-LSTM - common crawl word vectors (Kai, 2015)	0.6158	0.4589	0.1776
TreeLSTM-CBOW Contextual Bandit	**0.8572**	**0.6086**	**0.0301**

Table 2: Final Results

on the finance-specific terminology that it contains. These financial terms occur frequently in our corpus. Our results also demonstrate that the GloVe model generally performed worse than shallow window based methods. We recognize that the GloVe model would have benefited from training on a larger corpus.

4.2 Relative Performance of Models

Table 2 demonstrates that bag of word methods outperform all other single methods in measuring the similarity of financial phrases. Averaged CBOW word representations exhibit the highest correlations, and lowest mean squared errors. Applying Mikolov's Paragraph Vector to represent sentences using GloVe vectors, provides only a modest improvement in performance. This supports our earlier observation that the semantic meaning of a phrase is strongly dependent on the constituent finance-specific terminology, irrespective of word order (which Paragraph Vector attempts to resolve). We observe as well that the LSTM and Bi-directional LSTM models exhibit the poorest performance in capturing phrase similarity. We speculate that their lackluster performance is a consequence of our use of GloVe word-vectors that were trained on a non-finance dataset (Common Crawl). Furthermore we recognize that the fact that the models themselves were trained on a non-finance dataset (SICK dataset) limits the performance that we can achieve.

Tree-LSTM derived sentence representations exhibit intermediate performance. We recognize that the Tree-LSTM framework suffers from the same shortcomings as the LSTM and bidirectional LSTM frameworks. Moreover we found that applying our optimal CBOW word representations, and training on our dataset resulted in poorer performance. We speculate the poorer performance is a consequence of the fact that Tree-LSTMs utilize both euclidean and angular distances to as-

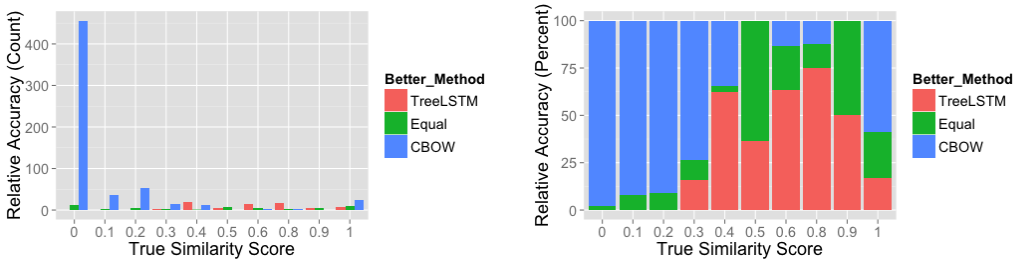

Figure 5: Comparison of CBOW and Tree-LSTM models on accurately predicting sentence similarity. Counts or percentages of sentences in each true similarity bucket are shown for which each model (or both equally) predicts a value closer to that true score.

sess phrase similarity. We found that our averaged CBOW vectors perform poorly when euclidean distance is applied as the distance metric. Lastly we recognize that the Tree-LSTM model would benefit from training on a larger finance-specific dataset.

We explored the differences between CBOW and Tree-LSTM further by examining the relative performance of each model across different regimes of the ground-truth sentence similarity labels (see Figure 5). We observe that CBOW-derived phrase representations outperform all other models at low similarity scores. Sentences that share few finance-specific terms are likely dissimilar. CBOW, which primarily relies on shared terms, better identifies dissimilar sentences (Table 3). At intermediate scores (0.4-0.9), the Tree-LSTM framework performs better than CBOW. This superior performance underscores that fact that tree structured models are more attune to subtleties shared between phrases that are non-obvious. In the second example sentences provided in Table 3 we observe that Tree-LSTMs are able to resolve that the two sentence are somewhat similar in their projection of some form of earnings over a fixed time period. Finally, from Figure 5, we see that at similarity scores of 1, CBOW outperforms Tree-LSTMs once again.

Sentences	**Truth**	**TreeLSTM**	**CBOW**
We maintain our overweight rating and establish a dec 2015 price target of $2.30. The company generated bpo book-to-bill of 10x well above the company average.	0	0.6	0
Expectations for the sequential change in real license revenue for the november quarter (+1.0%) are higher than the average seasonality experienced by oracle between 2013 and 2014 (+0.5%). Management forecasts that aig will earn a 0.6%+ roe in 2015 which is significantly higher than our model assumption of 0.3%.	0.4	0.6	0

Table 3: Word Vector Results

To address the striking unimodal distribution where the Tree-LSTM outperforms CBOW when the true similarity is neither high nor low, we developed the 2-armed bandit. We chose the Jaccard Distance between the two sentences as a proxy for lexical and syntactical similarity, and used this as the context for our bandit to make a decision on which model to trust. This approach outperformed the Tree-LSTM and CBOW individual models, which we consider to be a confirmation that, at least on the environment we studied, Tree-LSTMs work better than CBOW when sentences have moderate lexical differences and CBOW outperform Tree-LSTMs more extreme cases.

5 Conclusion

We evaluated the performance of a number of vector-space models such as CBOW and GloVe, as well as a number of Neural Network LSTM models, in a sentence similarity prediction task where sentences were extracted from financial research publications. Predictions using CBOW averaging performed best among the simpler vector-space models, whereas the Tree-LSTM presented by Tai et al. (2015) performed best among the neural network models. However, a surprising finding was that the CBOW averaging technique outperformed all neural models despite its simplicity.

We suspect that higher performance of the CBOW model is partly due to the inherent differences between the data our LSTMs were trained on vs. the data it performed predictions in. Although the LSTMs were trained on a relatively large movie review data set (SICK), there are substantial syntactical, topological and lexicographical differences between movie reviews and financial publications. We attributed part of the neural models' under-performance to these differences and gained positive indication of this fact when we instead tried to train on our own financial data.

Nonetheless, we observed that the Tree-LSTM model despite its overall lower performance was clearly able to outperform CBOW when similarity of sentences was moderate; not very similar and not dissimilar either. We attributed this phenomenon to CBOW's inability to produce good similarity predictions on sentences that are lexically or syntactically dissimilar yet preserve the same meaning. We took advantage of this insight and developed a contextual bandit algorithm with the ability to choose which model to trust more based on a proxy measure for lexical/syntactical similarity. We used the Jaccard Distance between two tokenized sentences as this proxy measure. The resulting bandit model was able to outperform all other models we studied in all error and correlation measures.

6 Future Work

Our contextual bandit algorithm outperformed all other models but still suffers from considerable hindrances, mainly due to the weakness of our particular Tree-LSTM model. To address this issue, we believe that improving the size and reliability of our training data is imperative. A preliminary study of how our predictions improved with data set size leads us to estimate that at least 10,000 labeled sentence pairs, all from financial texts containing in-line numbers, will be required to obtain competitive measures. The granularity and subjectivity of the labeling process should also be addressed by averaging the similarity labels of more than one expert.

Short of improving the data, another potential source of improvement could be to introduce a more robust bandit algorithm. Our current implementation serves only as a proof of concept; it utilizes only a single context feature and has an elementary measure for regret and reward estimation. Given the modal distribution of success percentage we observed in figure 5, we believe that a contextual bandit using the LinUCB (Li et al., 2010) algorithm and a richer context vector could yield state-of-the-art results.

References

[1] Glove: Global Vectors for Word Representation, Jeffrey Pennington, Richard Socher and Christopher D. Manning, *Conference on Empirical Methods in Natural Language Processing* (EMNLP 2014)

[2] Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks, Kai Sheng Tai, Richard Socher, and Christopher D. Manning, *arXiv* preprint, 2015.

[3] Danqi Chen and Christopher D Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. *Proceedings of EMNLP* 2014

[4] Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In Proceedings of HLT-NAACL 2003, pp. 252-259.

[5] H. Robbins. Some Aspects of the Sequential Design of Experiments. In Bulletin of the American Mathematical Society, volume 55, pages 527535, 1952.

[6] Mohri, Mehryar and Vermorel, Joanns. Multi-armed Bandit Algorithms and Empirical Evaluation. Lecture Notes in Computer Science Volume 3720, 2005, pp 437-448.

[7] Tai, Kai Sheng, Richard Socher and Christopher D Manning. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. arXiv preprint arXiv: 1503.00075 2015

[8] Mikolov, Tomas and Quoc Le. Distributed Representations of Sentences and Documents. arXiv preprint arXiv:1405.4053

[9] Mikolov, Tomas, Kai Chen, Greg Corrado and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781

[10] Li, Lihong, Wei Chu, John Langford and Robert E. Schapire. A Contextual-Bandit Approach to Personalized News Article Recommendation. WWW 2010 Conference preprint. April 26-30 2010