Interactive Visualizations for Deep Learning

Jason Chuang and Richard Socher

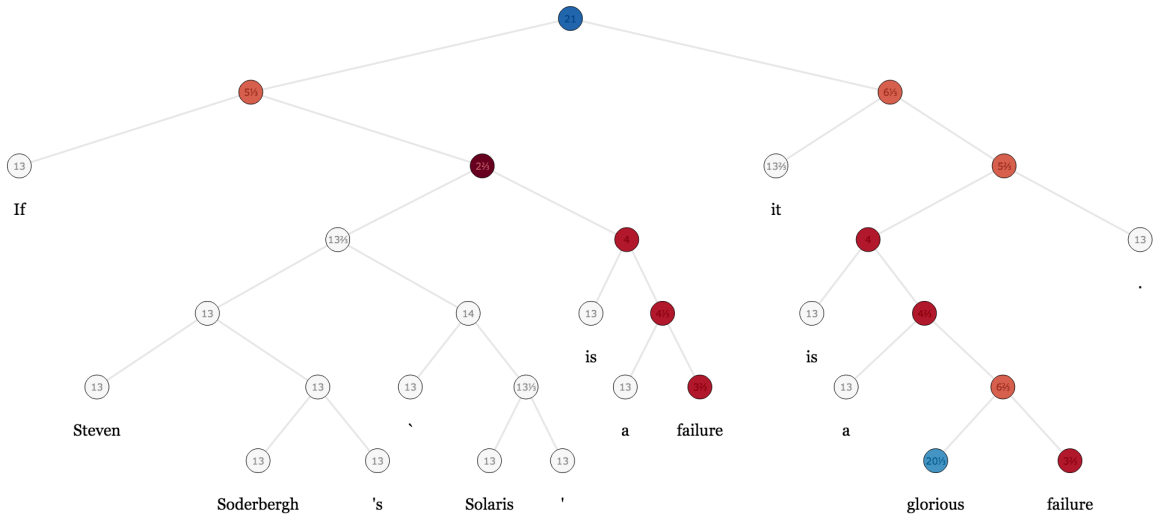

Fig. 1. This SentimentTree visualization shows how word-level sentiments build up over multi-word phrases to produce the overall sentiment associated with a sentence. Leaf nodes represent individual words such as `Steven`, `Solaris`, and `failure`. Non-terminal nodes represent phrases such as `glorious failure`. The root node represents the full sentence: `If Steven Soderbergh's 'Solaris' is a failure it is a glorious failure`. Colors encode sentiment ratings given by human subjects, where blue is positive and red is negative. Human raters find the majority of words in this sentence to be neutral with the exceptions of `failure` being negative and `glorious` being positive. Sentiment compositionality comes from combining words or phrases to form longer text, but the resulting ratings can take many forms. The phrase `glorious failure` is rated as negatively as the word `failure` itself. However, while the two main clauses of this sentence, `Steven Soderbergh's 'Solaris' is a failure` and `it is a glorious failure` are both negatively rated, their concatenation suggests an overall positive view about the film Solaris. Our algorithm produces the current state-of-the-art accuracy on sentence-level sentiment predictions.

1 Introduction

We describe how interaction visualizations contribute to the development and deployment of *Deeply Moving: Deep Learning for Sentiment Analysis*, a state-of-the-art classifier for sentence-level sentiment predictions. During model development, we create a *SentimentTree* visualization to facilitate explorations of user-labeled data as well as studying the model's characteristics. Designed to work at two zoom levels, the visualization is suitable for inspecting both individual data instances as well as a corpus of 10,000 data points. Beyond initial internal use, the visualization is also deployed on the web to collect user corrections and enable continued refinement of the algorithm.

We apply additional visual analysis techniques to aid feature engineering during model design. While our primary analysis task is akin to data exploratory over a high dimensional space, we find that existing multi-dimensional visualization techniques do not offer sufficient explanatory power to support our task. We examine analysis needs associated with feature engineering for supervised machine learning, the gap in available tools, and the potential for future research in both visual analysis and machine learning.

- *Jason Chuang is with Computer Science and Engineering at the University of Washington. E-mail: jcchuang@cs.washington.edu.*
- *Richard Socher is with the Computer Science Department at Stanford University. E-mail: socherr@stanford.edu.*

Manuscript received 31 Mar. 2014; accepted 1 Aug. 2014; date of publication xx xxx 2014; date of current version xx xxx 2014.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

2 Sentiment Analysis

Most approaches in sentiment analysis are based on bag-of-words representations [3] treating a sentence as frequency counts of its constituent words — or heuristics [6] that extract known aspects about specific domains such as restaurant reviews being comprised of ratings on food, ambience, and service. The sentiment associated with a sentence is then computed by summing or voting on word- or aspect-level sentiments. However, such approaches ignore the linguistic structure of a sentence, and can fail to capture opinions expressed in texts with more complex constructions, as shown in Figure 1.

2.1 Background Information

Our work builds on several areas of natural language processing. First, prior research in semantic vector spaces [8] finds that multiple interpretations of a word can be captured through word vector presentations that account for a word's context of use, though they do not work well for longer phrases. Second, Mitchell et al. [1] shows that compositionality in the word vector space — such as additions and multiplications of word vectors — can produce semantically meaningful categories of words. Third, Zettlemoyer et al. [9] shows that logical forms can be applied to sentiment compositionality. Finally, Pollack [4] demonstrates that compositionality can be incorporated into neural networks.

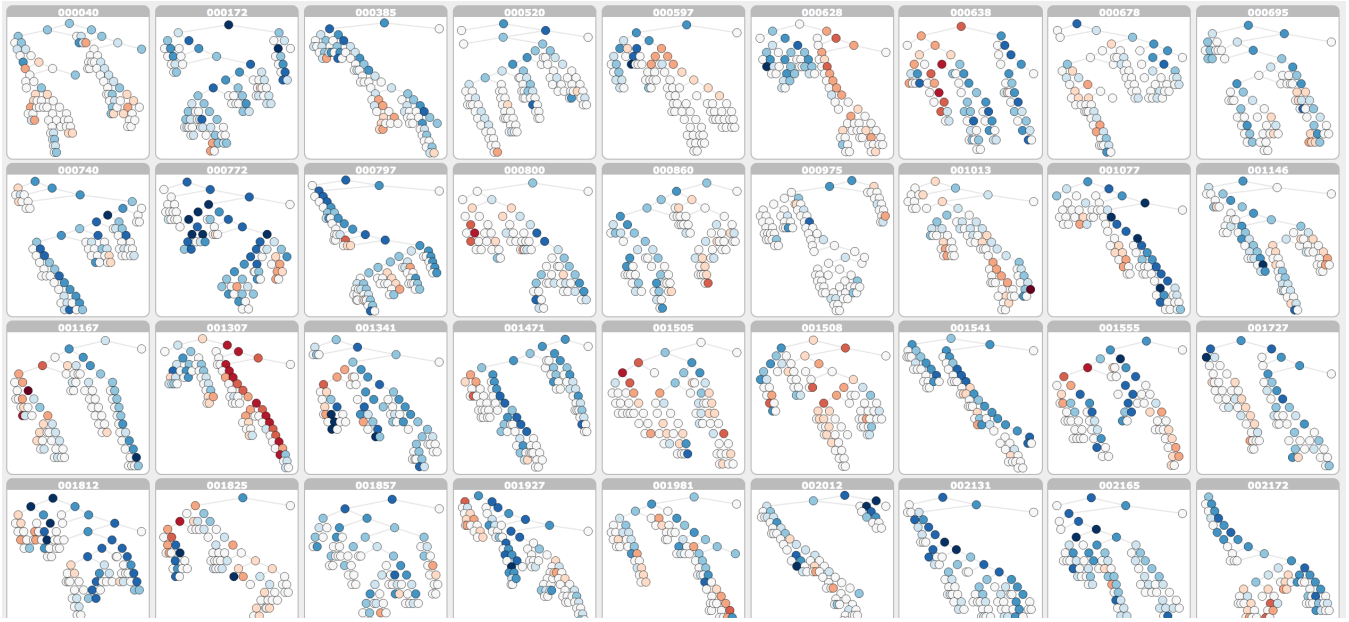

Fig. 2. The simplified representation allows for an overview of the annotated corpus and more rapid browsing.

2.2 Sentiment Prediction with Deep Learning

Our work on recursive neural tensor network [7], a class of deep learning algorithms, produces predictive models that take a sentence as input and estimate the overall sentiment expressed in the sentence.

Through a bottom-up process, our models first estimate word-level sentiments and represent them as feature vectors; they then compositionally combine the sentiments and features vectors for multi-word phrases through a parse tree, until reaching the full sentence. The compositional functions are learned by training the models on a large corpus of annotated sentences.

We demonstrate our model on a dataset of 11,855 sentences extracted from movie reviews on Rotten Tomatoes [2]; the sentences contain a total of 215,154 phrases within their parse trees. We used Amazon Mechanical Turk to obtain three human sentiment ratings for each sentence and phrase.

3 SentimentTree Visualization

We devise the *SentimentTree* visualization for inspecting the sentiments associated with a sentence, as they build up from words to phrases along a linguistic parse tree. We utilize the visualization for several purposes: exploring the annotated corpus, examining model predictions, identifying patterns of misclassifications, and supporting experimental designs. Feedback from domain experts highlight the need for multiple levels of zoom and effective data specification. Though initially designed for internal use, the visualization is now deployed publicly to support online learning.

3.1 Visual Design

Two early design decisions pertain to the orientation of the tree and the placement of text labels, both of which were largely determined by domain-specific considerations.

Leaf nodes in a parse tree represent words. Words can have varying number of characters, with corresponding text labels that vary greatly in length. During prototyping, we find that by laying out the parse tree horizontally (with the root of the tree on the left side and leave nodes on the right), we can allocate a fixed height to each leaf node, and generate nicely-formatted trees while allowing text labels to extend unobscured to the right. However, such designs were met with fierce resistance from collaborators in natural language processing, where the convention is to always render parse trees vertically.

Our final design, as shown in Figure 1, applies a vertical layout and determines node widths in proportion to its word length. Though it may unintentionally emphasize longer words, domain experts find the layout more intuitive. Lay users also find reading across the screen to be more natural than down the screen, as for vertical layout.

3.2 Usages of the Visualization

The visualization allows us to explore the annotated corpus, so that we can build up an intuition on how people perceive the sentiment of a passage of text. For example, we find that many of the words and phrases are labeled as neutral. Stronger sentiments tend to build up in longer phrases; the majority of the shorter phrases are rated as neutral. Such findings suggest limitations to previous approaches that try to predict sentence-level sentiment by voting on word-level sentiments without additional linguistic structure information.

The SentimentTree visualization also allows us to track the performance of the model along all stages of its modeling pipeline. While our ultimate goal is to ensure accurate sentence-level predictions, these additional details allow us to build up a profile of the model performance for shorter phrases and words. By comparing such profiles to observed human ratings, we find insignificant differences when we reduce our algorithm from a 25-class prediction problem (estimating sentiment with 25 levels of granularity) down to a 5-class prediction problem (estimating only five levels of sentiments: *positive*, *somewhat positive*, *neutral*, *somewhat negative*, and *negative*).

Furthermore, we superimpose model output on top of the user-supplied sentiment ratings in the visualization, to identify sources of misclassification errors. The visualization also contributes to our experiments on contrastive conjunctions and negations. Supporting these latter two tasks, however, requires additional visualization and interaction support.

3.3 Two Levels of Zoom: Overview and Details

While rendering a single tree is already useful, domain experts quickly find themselves wanting (and eventually needing) to browse the full corpus of over 10,000 sentences more efficiently.

However, as each sentence in the dataset generates a different parse tree, these trees have no correspondences among their structures to allow for aggregation. By carefully adjusting the visual aesthetics, we find that we can shrink each tree by up to a couple orders of magnitude in area while preserving their overall shape. We develop a simplified representation by removing the text labels and adjusting the relative

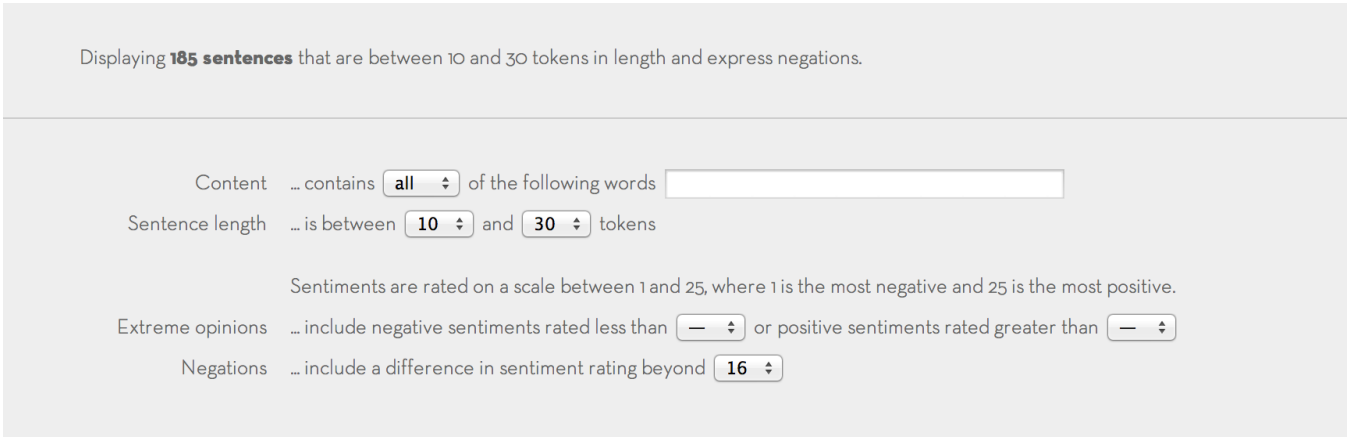

Fig. 3. User interface for specifying common search queries over the collection of parse trees representing the annotated corpus.

node size and edge length. A simplified view is designed to animate smoothly to a detailed view, and vice versa. By providing two levels of zoom, we are able to display up to hundreds of trees in a single monitor (Figure 2) and support more rapid scanning.

3.4 Interactions to Support Data Specification

Another issue of working with complex internal model structure is the lack of a natural means to search or filter data instances.

Let's consider Shneiderman's visual information-seeking mantra [5] that recommends *overview first, zoom and filter, then details-on-demand*. With the simplified and detailed representations and the capability to display text labels and superimpose model output, our tool now supports every operation specified in the mantra, except for filtering. However, unlike more traditional numerical data types where filtering operations can be defined via equality tests or over a range of numbers, filtering is ill-defined for our dataset of trees, each with a unique structure.

We define the filter operation as search over all subtrees within the corpus of parse trees. As our visualization is implemented in javascript, we leverage the functional nature of the language, and permit users to pass in functions that can be recursively applied to all nodes belonging to a tree, as a means to select data instances of interest. Such operations occur frequently during model design. For example, when a researcher identifies a misclassified data instance, she may wish to investigate whether certain linguistic structures are causing problems for the model. She may wish to find all other data instances exhibiting similar linguistic structures, such as "*sentences containing a contrastive conjunction (e.g., but) followed by a negation word more than 10 words away*", in order to determine whether the problem is systematic.

We subsequently develop a user interface for specifying common queries (Figure 3). Per best practices for search user interfaces, we also add a text box for general purpose text search. We include a descriptive phrase at the top, to help users understand the actions of their queries.

3.5 Deployment and Online Learning

While initially designed for internal use, the visualization has since been deployed on the web at `http://nlp.stanford.edu/sentiment`.

The visualization helps explain the action of our algorithm to lay users. We also utilize the visualization as an input tool, to elicit user feedback on any modeling errors and use such corrections to continuously improve the performance of our model.

4 Feature Engineering

We also apply visual analysis techniques to feature engineering, a subtask within model development where researchers turn unstructured text into linguistic features amenable to statistical analysis. Feature engineering allows researchers to leverage insights they have about a corpus, a model, or a specific domain, and operationalize them as features can then be fed into the predictive algorithm.

4.1 Improving Classifier Performance

Research in natural language processing has produced a large body of useful linguistic features that can be used as the initial feature set. In our experiences, rather than identifying *globally useful* features, feature engineering more often concerns (1) identifying additional features that complement the existing feature set to reduce misclassification errors, in a process known as boosting or (2) preventing the model from overfitting to specific features, which improves the apparent accuracy of the model on known data instances but causes problems when the model is applied to unseen data.

4.2 Multi-Dimensional Visualizations

We point out that both tasks involve exploring a high-dimensional space for recognizable patterns. One way to approach boosting is to identify regions of high-dimensional feature space with a high density of misclassified instances. Uncovering such regions may help model builders craft additional features that target recurring misclassifications. Similarly, one way to recognize overfitting is to identify regions of the solution space (where the space is the combination of parameters and features that define a model) where model prediction accuracy on test data consistently falls.

However, we find that existing multi-dimensional visualization techniques perform poorly despite what appears to be their intended use. Categorically, we find three general approaches to high dimensional visualization: (1) "Overview" through the projection or rescaling of the entire solution space. In these tools, users interact with all data dimensions that are mixed into two or three principal axes. The layout of the such visualizations, in theory, also account for interactions between all data dimensions. (2) "Independent axes" such as parallel coordinates. Users may interact with hundreds of dimensions independently, but see only a small number of interactions between select pairs of the data dimensions (3) "Pairwise interactions" such as scatter plot matrices that present a small number of dimensions and all pairwise interactions between the dimensions.

For both cases, we are typically examine dozens data dimensions of interest. Overviews don't generally provide sufficient details to inspect the dimensions of interest. Visualizations of independent axes help access the quality of features in isolation, but do not provide support to test combinations of the dimensions. While visualizations of pairwise interactions generally do not scale up sufficiently.

We extract relevant dimensions of the multidimensional space, and interactively explore them as multiple tables in Tableau.

5 Conclusions

In this paper, we reflect on our own experiences applying visual analysis to the development of a deep learning model for sentiment analysis. We highlight the various roles visualizations played throughout the initial data exploration, model design, and model deployment.

Acknowledgments

We thank Quoc Le, Andrej Karpathy, Choon Hui Teo, Jeffrey Pennington, Christopher D. Manning, and Jeffrey Heer for their input.

References

[1] J. Mitchell and M. Lapata. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429, 2010.

[2] B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, pages 115–124, 2005.

[3] B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2008.

[4] J. B. Pollack. Recursive distributed representations. *Artificial Intelligence*, 46, 1990.

[5] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *IEEE Symposium on Visual Languages*, pages 336–343, 1996.

[6] B. Snyder and R. Barzilay. Multiple aspect ranking using the Good Grief algorithm. In *HLT-NAACL*, 2007.

[7] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, 2013.

[8] P. D. Turney and P. Pantel. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188, 2010.

[9] L. Zettlemoyer and M. Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*, 2005.