CS 229 Final Project
Sentiment Causation Extraction

Desmond C. Ong (dco@stanford) & Wen Hao Lui (whlui@stanford)

I. Introduction

In state of the art sentiment analysis, text is analyzed for a single unidimensional *sentiment* or *opinion* score. This unidimensional sentiment, however, cannot capture the nuances of emotions: for example, two texts that respectively convey *anger* and *sadness* will both have a negative sentiment associated with them, while carrying very different connotations. Thus, we require a tool that allows more sophisticated analysis of emotional text.

This need for more sophisticated *relation extraction* is relevant outside of sentiment analysis as well. For example, causal relation extraction has important applications for building question and answer systems. Relation extraction remains an incredibly difficult problem to solve, but offers many promises such as constructing abstract knowledge representations or constructing narrative sequences from newspapers [1].

In this work, we propose building a causal relation extraction tool specifically for extracting the cause of emotions. Although we focus on causal extraction of emotions, the tool is generalizable to causal extraction in other domains.

II. Prior research

In recent years, there has been an exponentially increasing number of papers in sentiment analysis, most of which use simple bag-of-words models and a few that have made extensions by using bigrams [2] and deep learning over annotated sentiment of parse trees [3]. These unidimensional sentiment scores do not give insight into the different nuances of similarly valenced emotions (e.g. within negative emotions {*anger, sadness, disgust, disappointment, anxiety*}, or within positive emotions {*happy, excited, content*}).

A next class of research involving emotion analysis in text has focused on identifying a set of basic emotions in text. These have mainly focused on training a classifier (using text that has been annotated into one of these emotional categories), and has been used to classify newspapers [4] and blogs [5].

Several papers go a step further: [6] reported an emotion extraction system with a parser that accounts for auxiliary verbs, referent, tense, conditionals, etc, while [7] proposed a fuzzy semantic typing method to analyze affect, and the system generates an affect profile based on analyzed words.

Our goal in this paper is not to identify the emotion in text (although that is a component of our model that can be improved upon using previous research), but, given an already identified emotion in the text, identify the potential cause of the emotion. That aspect of our work is more similar to work in causal relation extraction, such as using annotated causal-pair examples to extract linguistic features.

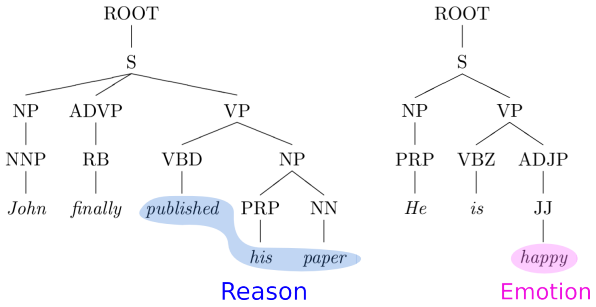

Fig. 1. Example of a sample problem. Given the two parse trees in this paragraph and the emotion "happy", we want to be able to identify the correct reason "published his paper".

III. Model

A. Statement of research objective

The goal is to build a system that, given a *paragraph* of text (consisting of ≥ 1 sentence) and an emotion word in the paragraph, identify the cause of the emotion. Candidate phrases from a dependency parse of the paragraph are considered, and the most likely phrase is chosen as the cause. None of the causes can also be chosen, if they all do not seem likely.

B. Corpus

The corpus we used is from the Experience Project (EP), which is a social networking site where users join groups based on shared experiences, e.g. "I am a military wife", or "I am a cancer survivor", or even "I am lonely". Users can, among other things, post stories and comment on others. The portion of the corpus we used for this work consists of short paragraphs of free-form text from Feb, 2008 through June, 2010, and consists of approximately 140,000 short paragraphs.

C. Generation of labeled dataset

From the 140k paragraphs, we selected a subset of 6k paragraphs, and applied simple filters to discard the paragraphs with no identified emotions and to weed out potentially offensive material to get a set of 1,903 paragraphs. Of these, we had workers on Amazon Mechanical Turk label 1,267 paragraphs, while we hand-labeled the remaining 636 paragraphs to build intuition for later feature engineering.

For each paragraph, an emotion word was identified using a simple word-matching lexicon, which took into account

synonymous words using *synsets* (or synonym sets) constructed using thesauri – e.g. "joyful" and "happy" would fall under the same synset. The most frequently occurring emotion-synset in the paragraph was chosen. Human labelers were asked to "identify the cause of the given emotion" by copying the cause phrase into a text box, or by indicating if no cause was found. Two raters provided ratings for each paragraph. If there was disagreement, we took the longest common substring between the two labels.

Next, we used the Stanford Dependency Parser to parse all the paragraphs. Using the dependency parse trees, subtrees that contain more than 1 word (to avoid terminal nodes/unigrams) were identified as candidate phrases. The explanations labeled by human raters were tagged to the smallest subtree that contains the explanation. This subtree was then marked as a positive example for the explanation of the emotion in the paragraph. All other subtrees were marked as negative examples. In some paragraphs, there was no cause found, and so all the subtrees were marked as negative. Thus, our final labeled dataset consisted of a set of *paragraphs* (≥ 1 sentence, and thus ≥ 1 parse trees), a list of *candidate phrases* for each paragraph (all subtrees with ≥ 1 word within the paragraph), an *emotion* word within the paragraph. The $(phrase, paragraph, emotion)$ input tuple is labeled with a simple 1/0 indicator to indicate if the candidate phrase is a positive or negative example.

D. Linguistic Features

We designed 18 features in total, spread over the following categories:

1) Bag of Words: Given the dominant emotion in the paragraph, we create $(word, emotion)$ indicator features for each word in the phrase.

2) Part of Speech Tags: We included indicator features that coded for whether the candidate was a Noun Phrase (NP), a Verb Phrase (VP), an Adjective phrase (ADJP), an Adverb Phrase (ADVP), a Quantifier Phrase (QP), or a Prepositional Phrase (PP). In addition, we had a feature that coded for wh-clauses (who-what-when-where-why).

3) Proximity features: We coded several features that accounted for semantic cues in proximity. We added features that indicated if the candidate contained causal conjunctions like "because", "so", "that". Some other examples:

- Indicator feature for whether the candidate's left sibling contains "because" at the end. E.g. if the candidate phrase is [I fell down], then the feature fires if the sentence contains "...because [I fell down]...".
- Indicator feature for whether the candidate's left sibling contains "I am (emotion) that", for example, "I am sad that [I fell down]".

4) Distance features:

- Indicator feature for whether the candidate is in the same sentence as the emotion
- Indicator feature for whether the candidate appears before the emotion in the paragraph
- Indicator feature for whether the candidate appears after the emotion in the paragraph (The only case where both candidateBeforeEmotion and candidateAfterEmotion return 0 is when the emotion is within the candidate phrase).
- 3 distance metrics that code the absolute distance between the emotion word and the start/end (whichever is closer) of the candidate phrase, in units of characters, words and sentences. We scale these features appropriately to keep the maximum value below 1.

E. Learning

We break our learning problem into two phases. We first train a linear SVM classifier (using LIBLINEAR [8]) using a simple bag of words model. Each input's feature vector for this step is purely based on the bag-of-word indicators described earlier. The classifier then infers a score on the candidate phrase and the score is fed as a single feature into the second phase. The sparsity and large cardinality of the (bag of words) feature set makes a linear kernel well-suited for the learning in this phase.

We formulate this as an SVM with ℓ_1 regularization for the learned weights w:

$$\begin{aligned} \min_{w} \quad & \frac{1}{2}||w||^2 + C\sum_i \xi_i \\ \text{s.t.} \quad & y_i(w^T\phi(x_i)) \geq 1-\xi_i \quad \forall i \\ & \xi_i \geq 0 \quad \forall i \end{aligned}$$

Where $\phi(x_i)$ is the feature vector comprised of bag-of-word indicators for x_i, with a corresponding y_i label where $+1$ is a positive example and -1 is a negative example. C is an adjustable hyper-parameter which encodes the trade-off between error margin and minimizing the weight. C is set to 1 in our experiments. The last factor ξ_i encodes the slack for each misclassified example.

In the second step, we train a Radial Basis Function (RBF) kernel on the linguistic features (including the bag-of-words score) of the input $(phrase, paragraph, emotion)$ tuple. We have 18 linguistic features for this phase, which is a much smaller and more manageable set. This allows us to use the RBF kernel, which runs much slower than a linear kernel but has an infinite dimensional kernel feature space and is well-suited for exploiting non-linear relationships between the various features.

We present the RBF SVM objective in the dual form:

$$\begin{aligned} & K(x,z) = exp(-\gamma\|x-z\|^2) \\ \max_{\alpha} \quad & \sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j K(x_i,x_j) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C \quad \forall i \\ & \sum_i \alpha_i y_i = 0 \quad \forall i \end{aligned}$$

Of the 1903 labeled examples, we set aside 203 for testing and trained our model on the remaining 1700 examples. We chose to include more paragraphs for our training because we plan to scale up the training set using semi-supervised learning later, and wanted to reduce the amount of bias present in the seed training set.

1) Inference: Given a list of $(phrase, paragraph, emotion)$ inputs, where *paragraph* and *emotion* remain constant, the SVM classifier returns a list of scores corresponding to how confident it thinks each input is a positive example. We take the input phrase with the highest score and choose it as the output cause phrase.

2) "No Cause Found" Scenario: We define a score margin as the difference in scores between the chosen cause phrase and the second-best cause phrase. We initially formulated the "no cause found" situation as a simple threshold problem, where the inference algorithm would return an empty string if the score margin during inference was too low. However, there was no easy way to estimate a good threshold value that works across various examples. We thus decided to fold this component into our learning algorithm as well, by adding an empty string as one of the candidate phrases for each training example. The empty string would then activate its own unique indicator feature, allowing the SVM to effectively train on it as well.

F. Scoring Metric

We initially used a simple accuracy metric to keep track of the algorithm's performance, but some preliminary analysis revealed that the number of false negative cases was disproportionately higher than false positives. This is primarily because the number of positive examples is much less than the number of negative examples (for each paragraph, there are many subtrees, but there is only at most one correct cause). We use a BLEU score as a more useful metric instead. We also included a confusion matrix in our analysis to provide another perspective for understanding our model's performance.

1) BLEU: The F1 unigram BLEU scoring mechanism takes into account both precision and recall. Only unigrams are considered because higher n-grams are typically used to measure the fluency of the output, which is not a concern for our inference method. We are only considering subtrees in the parse trees extracted from the paragraph as possible outputs, and since all subtrees can be converted back to a original phrase or sentence in the input paragraph, we do not anticipate problems with fluency. A guess phrase that perfectly matches the correct label phrase will receive a BLEU score of 1.

As an example, take the guess phrase "the lottery winner" against the correct phrase "he won the lottery". The unigram precision is $p = 0.33$ and the unigram recall is $r = 0.5$, leading to an F1 unigram BLEU score of $2pr/(p+r) = 0.4$.

IV. Results

We experimented with different formulations of the SVM, and found that the best-performing configuration was the classification SVM using a radial basis function kernel. As seen in Table I, it significantly outperforms the other classifiers with a BLEU score of 0.342. The RBF kernel is superior to the linear kernel because it can map non-linear relations between features, which is useful in our small feature space (excluding the bag-of-words). The classification formulation (C-SVC) also performs better than the regression formulation (ε-SVR), primarily because of our formulation of the objective as a binary classification problem. The data input thus lends itself well to classification, while regression on it will require large slack margins that can skew the result.

Kernel	Training Set	BLEU Score
Linear	S	0.118
Radial Basis Function (ε-SVR)	S	0.165
Radial Basis Function (C-SVC)	S	0.342
Radial Basis Function (C-SVC)	S+U	0.427

TABLE I. **BLEU results** for various training configurations. "S" (Seed) = 1700 initially labeled training examples, "U" = 3142 unlabeled training examples (relevant for the semi-supervised approach)

	Predicted Positive	Predicted Negative	Total
Actual Positive	121	58	179
Actual Negative	1826	6779	8605
Total	1947	6837	8784

TABLE II. **Confusion matrix** for the SVM model trained only on the labeled set.

We next look at the confusion matrix in Table II and calculate some basic statistics:

$$Precision = TP/(TP+FP) = 6.2\%$$
$$Recall = TP/(TP+FN) = 67.6\%$$
$$Specificity = TN/(TN+FP) = 78.8\%$$
$$Accuracy = (TP+TN)/(TP+TN+FP+FN) = 78.6\%$$

Where TP = True Positive, FP = False Positive, TN = True Negative, FN = False Negative.

We see that although we have a reasonable performance in classifying both true positive and true negative examples, the precision statistic is remarkably low simply because there are a lot fewer positive examples than negative examples. Recall is significantly higher than precision, although they are both measuring the correctly classified positive examples.

The accuracy statistic is also misleading for our case. If we have a classifier that labels all of the test data as negative, the accuracy will be $8605/(8605+179) = 98.0\%$ even though it is not making any useful predictions. This misalignment is the primary motivation for choosing the F1 unigram BLEU as our metric for performance, since it is not affected by data where one label greatly outnumbers another. We also compensated for the relative low numbers of positive examples by making their weights 50 times that of the negative examples in our learning step.

V. Semi-supervised Learning

We performed training on an iteratively larger subset of the corpus in order to make use of the large amount of unlabeled data. Given that the initial labeled set of paragraphs is of size m, we first train and obtain the weights on that labeled set. We then perform labeling on another disjoint and unlabeled set of size $0.5m$ and obtain the labels for them, keeping the most confident $0.1m$ predictions based on

labeled set ← seed set ;
while *unlabeled set is not empty* **do**
    take 0.5×(size of labeled set) examples from unlabeled set ;
    classify taken examples ;
    pick top 20% with highest score margin, add to labeled set ;
    return remainder to unlabeled set ;
end

Algorithm 1: Semi-supervised Learning

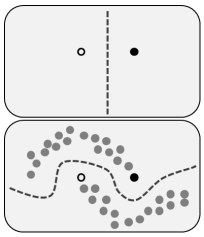

Fig. 2. Illustration of a semi-supervised learning approach. The black and white circles indicate labeled examples, while the gray circles are unlabeled. Using only labeled examples for learning may result in a decision boundary that does not fit the underlying structure of the data. By inferring labels on unlabeled data to grow the training set, we can find a better decision boundary. Image sourced from `http://en.wikipedia.org/wiki/Semi-supervised_learning`

the score separation of the correct pair from the second-best pair. We then perform the optimization of weights again on the combined $1.1m$ examples, repeating this process until we have trained on a sufficiently large set or reached convergence. See Algorithm 1 for the pseudocode.

The main strength of this approach is that it allows us to make use of the unlabeled set, which is about 100 times larger than our labeled set. Although we are using our own inferred labels for further training, which risks magnifying errors and biases in our original training set, in practice this approach has proven to work well as long as we can identify a useful heuristic for the metric we are measuring against (in this case, the score difference of the top phrase from the second-best one). Figure 2 shows how adding more unlabeled examples (paired with their inferred labels) can considerably help with forming a better decision boundary

A. Semi-supervised results

Looking at the results in Figure 3, we see a clear upward trend in both the accuracy and the BLEU score. The gains in accuracy are due to the reduction of false positives, as seen in Table III. The BLEU score plateaus at around 0.42, which is significantly higher than that of the original 0.342. We believe that good directions for future work would be to find more discriminative features and start with a larger and more diverse seed set to help to raise the upper limit on the semi-supervised approach.

One interesting feature in the learning curve is the significant drop in the BLEU score for the first iteration. This is probably an artifact of the algorithm that we are using, which chooses the top 20% inferred labels with the highest score margin. Since the score margin has a lower limit of 0,

	Predicted Positive	Predicted Negative	Total
Actual Positive	100	79	179
Actual Negative	795	7810	8605
Total	895	7889	8784

TABLE III. **Confusion matrix** for the SVM model trained on 1700 labeled and 3142 unlabeled training examples. The number of false positives is greatly reduced compared to training only on the labeled set.

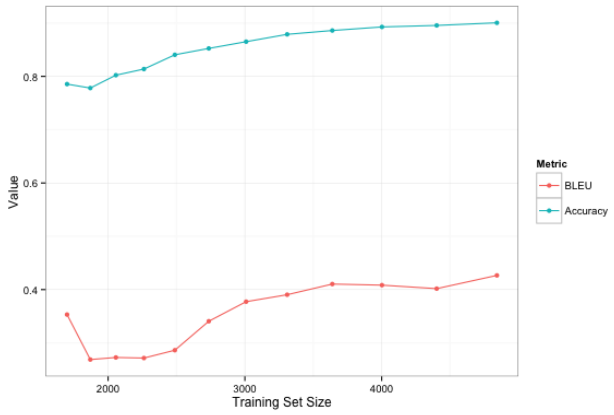

Fig. 3. The learning curve for the semi-supervised approach we are taking. We see significant gains in both the BLEU score and the accuracy as the size of the training set increases with the number of iterations.

positive examples tend to have much higher score margins and are selected disproportionately.

Examining the confusion matrix for the first iteration, we see that our classifier increases the amount of true positives labeled, but reduces the number of true negatives identified. Since the negative examples outweigh the positives by about 50 times in our test set, the BLEU score understandably decreases. However, the positive examples added are still valid, and therefore when we add more training examples in subsequent iterations we tend to increase the BLEU score as well.

VI. Error Analysis

The largest source of remaining errors is incorrectly predicting the presence/absence of causal phrases. Even though we encoded this as a feature, the classifier still cannot differentiate between the two cases sufficiently. Abstracting this away as a separate problem, such that our input data only consists of paragraphs with an identified causal phrase, would significantly improve performance in identifying cause phrases.

However, the problem of identifying if there exists a cause phrase in a paragraph in itself is not trivial, and using a feature-based classifier for this simplified problem may not yield significantly improved performance over our current implementation.

Another common error that our tool makes is that the guesses it makes often contains the emotion itself, e.g. “that your life makes you happy” when the gold label is “your life”. More often that not, the candidate phrase that contains the cause usually does not contain the emotion, and although

we have features that account for that, perhaps there needs to be additional features that code for dependency structures more. In this case, the correct answer is actually contained within the guess, but one level down.

We also noticed a drop in performance when the labeled cause is in a different sentence from the identified sentiment. This is because we lose quite a bit of structural information in the features due to the dependency parser operating on a sentence level instead of a paragraph level. One interesting extension would be to try and encode some kind of dependency relationship between sentences as well, instead of just words within a sentence. For example, establishing temporal relations between sentences can help the classifier focus on sentences with events that occur before the events corresponding to the given sentiment. We can also include some basic co-reference resolution, to help establish links between sentences.

Lastly, the cleanliness of the input also posed some challenges. Because we were taking data from a live website with user postings, there were often misspellings and punctuational acrobatics that challenged the Stanford dependency parser and led to unpredictable features for those examples. For example, some users do not leave spaces between their sentences, which causes the dependency parser to interpret the two sentences as an usually long one. This led features like those based on part-of-speech tags to misfire. Given more time, we would have written more code to clean the data and do some sanity checks before sending it out for labeling.

VII. Limitations

Because the main goal of our project was relation extraction and not emotion identification, we used a very naive system to identify the emotions in text, namely, word-matching against a dictionary of emotion terms. We considered only unigram emotion words without context, and hence our tool identified examples like "*happy* family" and "*upset* stomach", where the emotion word does not convey a state of feeling a particular emotion, but just an adjective ("*happy* family") which sometimes has a semantically different meaning ("*upset* stomach"). It also identified examples where the emotions were part of phrases or idiomatic expressions, such as "*happy* birthday", "*sad* to say", "I'm *afraid* that", and "just as I *feared*". This was only a small subset (estimated to be 10%), and they were labeled as "No Cause Found" by human raters. We can probably greatly improve our system by improving our emotion identification tool with more contextual features.

A tricky case that we came up with is the case of conditionals. For example, the sentence "I am afraid of X", sometimes means that "X caused (me to feel) fear" (e.g. "spiders"), but might also means that the *possibility of X happening* causes fear (e.g. "death"). This also made us rethink our definition of causality and whether a conditional event "causes" an emotion (for the purposes of this paper, we said yes); this should be an important point to consider in future work on causal relation extraction.

VIII. Conclusion

In conclusion, we have developed a tool that performs well at identifying the causes of an identified emotion in a multi-sentence paragraph. We have also shown that, given only labels for a small fraction of the dataset, we are able to bootstrap our way using a semi-supervised learning algorithm that improves the prediction of the classifier tool. Future work would be to include other types of relations, such as identifying the agent who is experiencing the emotion, and in applicable cases, identifying the target of the emotion.

As mentioned, this work has many applications to more nuanced emotion analysis of text. In addition, the tools developed here can be generalized to other domains. For example, the emotion identification step of our algorithm can be generalized to event identification, and the causal relation extraction part can be generalized to other relations. The framework of this tool offers many promising applications for more powerful natural language processing.

IX. Acknowledgements

We thank Christopher Potts for sharing the corpus data with us; Noah Goodman, Dan Jurafsky, Christopher Manning, and Justine Kao for helpful discussion; and Emily Yeh for working on the emotion synset classification tool. We also want to thank Sebastian Schuster, Spence Green, Milind Ganjoo, and Bharath Bhat for useful advice.

X. Miscellaneous Notes

Wen Hao is taking CS224N and CS229, and is using the project for both of those classes as well.

Desmond is not taking CS221, but will be submitting the project for CS224N.

References

[1] Chambers, N., & Jurafsky, D. (2009, August). Unsupervised learning of narrative schemas and their participants. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2 (pp. 602-610). Association for Computational Linguistics.

[2] Wang, S., & Manning, C. D. (2012, July). Baselines and bigrams: Simple, good sentiment and topic classification. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2 (pp. 90-94). Association for Computational Linguistics.

[3] Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. EMNLP.

[4] Strapparava, C., & Mihalcea, R. (2008, March). Learning to identify emotions in text. In Proceedings of the 2008 ACM symposium on Applied computing (pp. 1556-1560). ACM.

[5] Aman, S., & Szpakowicz, S. (2007, January). Identifying expressions of emotion in text. In Text, Speech and Dialogue (pp. 196-205). Springer Berlin Heidelberg.

[6] Zhe, X., & Boucouvalas, A. C. (2002, July). Text-to-emotion engine for real time internet communication. In Proceedings of International Symposium on Communication Systems, Networks and DSPs (pp. 164-168).

[7] Subasic, P., & Huettner, A. (2001). Affect analysis of text using fuzzy semantic typing. Fuzzy Systems, IEEE Transactions on, 9(4), 483-496.

[8] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A Library for Large Linear Classification, Journal of Machine Learning Research 9(2008), 1871-1874