Identifying Aspects and Values In Review Text

Kevin Clark

March 25, 2013

1 Problem Description

Web user's are increasingly sharing their opinions about products on the Web in the form of consumer reviews. Due to the typically large volume of reviews for a product, manually looking through them can be infeasible. Sentiment analysis systems, however, can extract and aggregate reviewer opinions expressed over several reviews. The results of these systems are often useful in downstream tasks, like summarization or recommendation. A simple form of sentiment analysis is predicting the overall sentiment (positive or negative) of a single review. Other systems can find much finer grained sentiment information by finding specific features for the item being reviewed (aspects) and the reviewer's opinions towards that aspect (values). For example, RevMiner [1] extracts (aspect, value) pairs from review text where the aspect and value are single words. However, more complex opinion relations require multiple words to be expressed. For example, we would like to extract (chocolate cake, very tasty icing) from "I tried the chocolate cake and it had very tasty icing." In order to build a system capable of this, I consider opinion mining as a tagging problem, where words are labeled as referring to an aspect, referring to a value, or neither. For example, "I tried the chocolate cake and it had very tasty icing" would be annotated "I$_n$ tried$_n$ the$_n$ chocolate$_a$ cake$_a$ and$_n$ it$_n$ had$_n$ very$_v$ tasty$_v$ icing$_v$" where the label a indicates an aspect word, v indicates a value word, and n indicates a word not referring to an aspect or value.

2 Data Set

I used a data set and annotations provided by Sauper and Barzilay [2] for their work on a similar task. The data consists of snippets selected from Yelp reviews by a previous system [3] that is trained to select snippers containing short descriptions of user sentiment towards some aspect of a restaurant. For this experiment, only snippets labeled by the system as referring to *food* were chosen.

2.1 Annotation

A portion of the phrases in the data set were given per-word annotations (each word was labeled as an aspect, value, or neither) provided by Mechanical Turk. In order to filter out annotators who did produce reliable labels, only annotators who were successful in producing mostly correct labels for a set of "difficult" phrases were allowed access to the phrases. Each phrase was presented to three annotators, and the majority label is taken for each word. In total, this produced a set of 732 phrases containing 6026 words.

2.2 Preprocessing

The set of phrases were split into 150 test phrases, 150 validation phrases, and 432 train phrases. To provide features for the models, the Stanford POS tagger and Stanford Parser were run over the full set of phrases.

3 Rule-Base Models

3.1 Labeling by Part-of-Speech

The part-of-speech of a word is fairly correlated with its label. For example, adjectives tend to be values because values are descriptive and nouns tend to be aspects since aspects are usually concrete objects. For the first baseline rule-based system, TAGS-SMALL, we assign each noun and aspect label and each adjective a value label.

For the second rule-based system, TAGS-LARGE, a larger set of tags were marked as values. In particular, numeral, adverbs, and verb participles were labeled as values in addition to adjectives.

3.2 Tree Expansions

Sequences of aspects or values tend to occur together under phrasal nodes of the tree. For example "pork with apple glaze" contains a sequence of aspect-identifying words all falling under a NP. To capture this idea, the tree expansion system allows aspects and values to propagate across the words falling under the same phrase in a tree. The procedure is as follows:

1. Label words based on their POS tag as in the first rule-based model.
2. For each aspect, find the largest noun phrase containing it such that every word under that phrase is unlabeled or labeled as an aspect. Label all words under the phrase as aspect words.
3. For each value, find the largest noun phrase, adjective phrase, or adverb phrase containing it such that every word under that phrase is unlabeled or labeled as a value. Label all words in the phrase as value words.
4. Remove all labels added in the previous steps for words that are punctuation marks, determiners, or conjunctions.

3.3 Comparison

Four rule-based models were all run on the validation set: TAGS-SMALL, TAGS-LARGE, and TREE-SMALL and TREE-SMALL, which adds the tree expansion procedure to the results of the TAGS-SMALL and TAGS-LARGE systems.

	Aspect			Value		
	Precision	Recall	F_1	Precision	Recall	F_1
TAG-SMALL	**80.69**	84.25	**82.43**	84.43	43.03	57.01
TAG-LARGE	**80.69**	84.25	**82.43**	86.07	66.83	75.24
TREE-SMALL	74.65	**87.85**	80.71	**87.59**	56.01	68.33
TREE-LARGE	75.79	86.46	80.77	86.14	**68.75**	**76.47**

Tree expansion improves recall by labeling more potential aspects and values, but this often comes at the expense of precision.

4 Markov Models

4.1 Hidden Markov Model

Hidden Markov Models (HMMs) are commonly applied to tagging tasks. For tasks like POS tagging, HMMs usually find the most likely sequence of states over a sequence of words. However, due to the small amount of training data, words proved to be too fine-grained for this task: most words occur only a few times in the dataset, leading to inaccurate probability estimates. Instead, the HMM finds the most likely sequence of labels $l_1, ..., l_n$ over a sequence of part-of-speech tags $t_1, ..., t_n$. Although it would have been possible to mediate this problem through smoothing, I decided to build this model without looking at words and add lexical information in the Maximum Entropy Markov Model (see the next section).

The transition score is conditioned on both the previous labels and the previous tags. The HMM assigns a score of label l_i as

$$\begin{aligned} score(l_i) &= \log p(l_i|t_{i-1}, ..., t_{i-d_t}, l_{i-1}, ..., l_{i-d_l})p(t_i|l_i) \\ &= \log p(l_i|t_{i-1}, ..., t_{i-d_t}, l_{i-1}, ..., l_{i-d_l})p(l_i|t_i)p(t_i)/p(l_i) \\ &\propto p(l_i|t_{i-1}, ..., t_{i-d_t}, l_{i-1}, ..., l_{i-d_l})p(l_i|t_i)/p(l_i) \end{aligned}$$

where d_t is the *tag depth*, the number of previous tags the transition score is conditioned on and d_l is the *label depth*, the number of previous labels the transition score is conditioned on. The probabilities are estimated with with maximum likelihood estimates:

$$p_{ML}(l_i|t_{i-1}, ..., t_{i-d_t}, l_{i-1}, ..., l_{i-d_l}) = \frac{c(l_i, t_{i-1}, ..., t_{i-d_t}, l_{i-1}, ..., l_{i-d_l})}{c(t_{i-1}, ..., t_{i-d_t}, l_{i-1}, ..., l_{i-d_l})}$$

$$p_{ML}(l_i|t_i) = \frac{c(l_i, t_i)}{c(t_i)} \quad p_{ML}(l_i) = \frac{c(l_i)}{c(l)}$$

The score for a sequence of tags $t_1, ..., t_n$ with labels $l_1, ..., l_n$ is given by:

$$q(t_1, ..., t_n, l_1, ..., l_n) = p(STOP|l_n) \prod_{i=1}^{n} score(l_i)$$

Inference is done by finding the most likely sequence under the model, which is computed using the Viterbi algorithm.

$$\arg\max_{l_1,...,l_n} q(t_1, ..., t_n, l_1, ..., l_n)$$

There are sequences of tags and labels that do not occur in the training data, so they get a probability of zero according to the ML estimate. To deal with this, I applied additive smoothing to the estimate.

To find the best values for d_l and d_t, I searched over possible values and picked the one with the best score on the validation set.

	Aspect			Value		
	Precision	Recall	F$_1$	Precision	Recall	F$_1$
$d_t = 0, d_l = 0$	80.16	84.81	82.42	86.32	68.27	76.24
$d_t = 0, d_l = 1$	80.35	76.80	78.53	78.06	73.56	75.74
$d_t = 0, d_l = 2$	84.04	77.07	80.40	78.10	**78.85**	78.47
$d_t = 1, d_l = 0$	80.60	**89.50**	84.82	**88.66**	71.39	79.09
$d_t = 1, d_l = 1$	**82.22**	88.12	**85.07**	87.43	75.24	**80.88**
$d_t = 1, d_l = 2$	81.82	87.02	84.34	85.21	74.76	79.64
$d_t = 2, d_l = 0$	81.22	88.40	84.66	86.61	73.08	79.27
$d_t = 2, d_l = 1$	81.20	85.91	83.49	82.43	73.32	77.61
$d_t = 2, d_l = 2$	81.10	85.36	83.18	81.67	72.84	77.00

The setting producing the highest F$_1$ score for both aspect and value identification was $d_t = 1, d_l = 1$.

4.2 Maximum Entropy Markov Model

Maximum Entropy Markov Models (MEMMs) work similarly to HMMs, except the scoring is learned with a discrete log-linear model:

$$score(l_i) = \frac{\exp(w \cdot \Phi(l_i, ..., l_{i-d_l}, t_1, ..., t_n, x_1, ..., x_n, i))}{\sum_l \exp(w \cdot \Phi(l, ..., l_{i-d_l}, t_1, ..., t_n, x_1, ..., x_n, i))}$$

In addition to taking previous d_t tags and d_l labels like the Hidden Markov Model, three additional features were added.

- NEXT-TAG: the tag of the word following the current one.
- WORD: the current word
- PHRASE-LENGTH: the number of words in the current phrase.

As with the HMM, the best values of d_l and d_t were determined through a search on the validation set.

	Aspect			Value		
	Precision	Recall	F$_1$	Precision	Recall	F$_1$
$d_t = 1, d_l = 1$	83.86	**87.57**	85.68	**88.06**	76.20	81.70
$d_t = 1, d_l = 2$	86.50	86.74	86.62	87.83	79.81	83.63
$d_t = 1, d_l = 3$	**87.96**	86.74	**87.34**	87.69	**82.21**	**84.86**
$d_t = 2, d_l = 1$	85.92	84.25	85.08	84.82	77.88	81.20
$d_t = 2, d_l = 2$	86.55	85.36	85.95	86.13	79.09	82.46
$d_t = 2, d_l = 3$	87.85	85.91	86.87	86.60	80.77	83.58
$d_t = 3, d_l = 1$	86.30	87.02	86.66	86.39	79.33	82.71
$d_t = 3, d_l = 2$	86.11	85.64	85.87	86.05	80.05	82.94
$d_t = 3, d_l = 3$	86.55	85.36	85.95	85.68	80.53	83.02

The best setting for both aspect and label identification was $d_t = 1, d_l = 3$. To evaluate the importance of the additional features, an ablation study was performed on the validation set. The NEXT-TAG and WORD features both significantly improved performance, while PHRASE-LENGTH gave a smaller boost to the F$_1$ score.

	Aspect			Value		
	Precision	Recall	F$_1$	Precision	Recall	F$_1$
All Features ($d_t = 1, d_l = 3$)	**87.96**	**86.74**	**87.34**	**87.69**	**82.21**	**84.86**
NEXT-TAG deleted	85.92	82.60	84.23	83.12	79.33	81.18
WORD deleted	86.11	85.64	85.87	86.52	77.16	81.58
PHRASE-LENGTH deleted	87.68	86.46	87.07	87.40	81.73	84.47

4.3 Adding Tree Features

Three features from the parse trees produced by the Stanford Parser were added to the MEMM model.

- PARENT: The phrase type of the parent of the current pre terminal.
- ANCESTOR-NP, ANCESTOR-ADJP, Whether an ancestor of the current pre terminal is one of these phrases. Following the observations for the Tree Expansion model, falling under one of these phrases can be indicative of the label.

I tried adding other features such as siblings, grandparent, and other ancestor phrases, but they deceased performance on the validation set. To evaluate the importance of the additional features that were added, an ablation study was performed on the validation set. Overall, tree-based features did little to improve the score of the MEMM. This could be explained in part by errors in the parse trees.

	Aspect			Value		
	Precision	Recall	F$_1$	Precision	Recall	F$_1$
All Features ($d_t = 1, d_l = 3$)	**87.19**	**88.40**	**87.79**	**88.65**	80.77	84.53
PARENT deleted	86.92	88.12	87.52	88.45	**81.01**	**84.57**
ANCESTOR-NP deleted	87.12	87.85	87.48	88.68	81.01	84.67
ANCESTOR-ADVP deleted	87.16	88.12	87.64	88.65	80.77	84.53

5 Results

5.1 Final Evaluation

The best rule-based model, HMM, and MEMM (with and without parse tree features), were all run over the test set.

	Aspect			Value		
	Precision	Recall	F$_1$	Precision	Recall	F$_1$
TAG-LARGE	82.49	84.50	83.48	82.18	71.14	76.26
HMM ($d_t = 1, d_l = 1$)	84.38	**90.27**	87.22	**87.17**	75.71	81.04
MEMM ($d_t = 1, d_l = 3$)	91.22	88.45	89.81	85.04	**87.71**	86.36
MEMM-TREE ($d_t = 1, d_l = 3$)	**91.30**	89.36	**90.32**	85.96	87.43	**86.69**

The MEMM-TREE model significantly outperforms the rule-based baseline on both aspect and value identification.

5.2 Error Analysis

The following are common errors made by the final MEMM-TREE system

Mis-identifying conjunctions: The word "and" is difficult to label because it can sometimes be part of an aspect, but usually isn't. For example, in "soup and salad" or "creamy and delicious", "and" separates two aspect or two value words and is labeled as being neither an aspect or value. However, in "cookies and cream", "and" is part of a three-word aspect.
Target: Cookies$_a$ and$_a$ cream$_a$ was$_n$ the$_n$ best$_v$
Predicted: Cookies$_a$ and$_n$ cream$_a$ was$_n$ the$_n$ best$_v$
One possibility of fixing this would be to identify collocations using a statistical test like pointwise mutual information over a large set of unlabeled reviews.

Review snippets with unusual/bad grammar: Reviews on yelp are sometimes written with informal or improper English. This causes errors in the POS Tagger and Parser, making the sentences hard to label. For example,
Target: Food$_a$ $=_n$ 3$_v$ stars$_v$
Predicted: Food$_a$ $=_a$ 3$_a$ stars$_a$
This could partially be dealt with by providing a specific training set for the POS tagger and Parser where the language is similar to the reviews.

Value descriptions as noun phrases: Typically, values are expressed as adjective or adverb phrases. However, for words like "highlight", the value is a noun. This causes errors like below:
Target: The$_n$ highlight$_v$ was$_n$ the$_n$ half$_a$ roasted$_a$ chicken$_a$
Predicted: The$_n$ highlight$_a$ was$_n$ the$_n$ half$_v$ roasted$_v$ chicken$_v$
This would also be challenging to deal with. Providing more data would help to some degree, because the MEMM could then possibly see examples of "highlight" and learn it is almost always a value.

6 Conclusion

A Maximum Entropy Markov Model proved effective for the task of providing per-word labeling to review snippets. Unfortunately, inter-annotator agreement was not available on the dataset was not available, but it is likely that there is not much improvement left in the system before subjectivity prevents further gains. The hand-build tree rules did not significantly improve the performance of TAG-LARGE and tree features did not significantly improve the MEMM. However, intuitively aspects and values marked in a sentence should be highly dependent on the sentence's grammatical structure. Because of this, a statistical method informed by the parse trees could likely outperform the MEMM. Adding parse tree features to the MEMM does not capture the idea that a word will likely refer to an aspect if its siblings under a subtree of the complete parse also refer to aspects. This could possibly be addressed by treating the task as a tree annotation problem. A model could learn a grammar that produces the appropriate labels for terminals (a terminal would, for example, have label NNS-a, meaning a plural noun referred to as an aspect) given a tree provided by the Stanford Parser. This would have the disadvantage of being even more affected by errors produced by the parser, but it would address the previously mentioned issue.

References

[1] Jeff Huang, Oren Etzioni, Luke Zettlemoyer, Kevin Clark, and Christian Lee. Revminer: An extractive interface for navigating reviews on a smartphone. 2012.

[2] Christina Sauper and Regina Barzilay. Automatic aggregation by joint modeling of aspects and values. *Journal of Artificial Intelligence Research*, 46:89–127, 2013.

[3] Christina Sauper, Aria Haghighi, and Regina Barzilay. Incorporating content structure into text analysis applications. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 377–387. Association for Computational Linguistics, 2010.