Hedge Detection using a Rewards and Penalties Approach

Ken Stahl[1], Samira Shaikh[1], Tomek Strzalkowski[1,2]

[1]State University of New York - University at Albany
[2]Polish Academy of Sciences

Abstract

Semantic and syntactic features found in text can be used in combination to statistically predict linguistic devices such as hedges in online chat. Some features are better indicators than others, and there are cases when multiple features need to be considered together to be useful. Once the features are identified, it becomes an optimization problem to find the best division of data.

We have devised a genetic algorithm approach towards detecting hedges in online multi-party chat discourse. A system was created using rewards and penalties for matching features in tokenized text, so optimizing the reward and penalty amounts are the main challenge. Genetic algorithms, a subset of Evolutionary Algorithms, are great for optimization; as they are massively parallel directed searches, and therefore suited to finding the best ratio of integer rewards and penalties. "Evolutionary algorithms (EAs) utilize principles of natural selection and are robust adaptive search schemes suitable for searching nonlinear, discontinuous, and high-dimensional spaces. This class of algorithms is being increasingly applied to obtain optimal or near-optimal solutions to many complex real-world optimization problems" (Bonissone, et. al. 2006)

We show results using 10-fold cross validation as commonly used in traditional machine learning. The best performance without further fine tuning is 79% in classifying whether an utterance in chat contains a hedge or not.

1 Introduction

A hedge is a mitigating device used to qualify or lessen the impact of an utterance. Writers reduce the "degree of liability" or responsibility that they might face in expressing referential information (Crismore & Van Kopple, 1988). In situation where a person is attempting to convince another of an assertion, especially if the other is of higher status (or a stranger) or if the message is counter-attitudinal, it is seen as polite to give the person the feeling that the assertion is not a command. Hedges used in this way provide the listener the opportunity to reject or disagree with the assertion (Durik, Britt, Reynolds & Storey, 2008).

As indicators, hedges are thought to help determine a person's gender, age, and leadership qualities (Martey et. al. 2010). An example of a hedge:
"It might rain later."

As opposed to:
"I will rain later."
Might, in the example above, is a hedge conveying an element of uncertainty.

Hedges are thought to be indicators used in predicting real life characteristics of online game players. When tasked with finding hedges, we originally tried using a hedge lexicon and doing string matches. However, the performance was unacceptable. Machine learning worked fairly well in specific corpora such as Wikipedia and biological datasets (Chen, Eugenio, 2010), but in a chat context, grammar and spelling have a more unpredictable nature. Using rewards and penalties, it is possible to create a system related to machine learning based on feature matching while having many alternate solutions and being able to fine tune weights and feature evaluations.

The features to identify hedges are based on tokens created by the Stanford Part of Speech tagger. Each token has a word or punctuation value as well as a part of speech tag; which are used as features. In addition the previous two tokens as well as the subsequent two tokens (if they exist) provide features. This combination of features take into account repetitions of exact use, grammatical context, and rough position in a sentence.

2 Comparison with Related Works

Various methods of machine learning have been applied to the problem of identifying hedges in text. Many of these systems perform between 0.6 and 0.8 (f-score), such as the Lucene and Maximum Entropy Model based hedge detection system (Chen, Eugenio, 2010) and RelHunter approach (Fernandes, Crestana, Milidiú, 2011). These systems depend on generating features such as relations and hedge cues and utilize very complicated systems.

We were interested in achieving good performance, but with a system that can adapt its evaluation strategy on the fly as well as being able to export the best solution as a tagger to be used in other projects. In addition, using a collection of individuals provides a variety of solutions which may not be optimal for the current data but potentially transferable to other data sets. Trends in the solutions also show how feature combinations work together as indicators in hedge identification.

3 Corpora

Training and testing were done using two annotated corpora. One corpus consists of 47 chat sessions in Second Life with 38807 lines (Small et. al. 2011). The other consists of 5 chat sessions in World of Warcraft with 2041 lines. The Second Life corpus had 1786 lines containing 1 or more hedge. The World of Warcraft corpus had 128 lines containing 1 or more hedges. Both corpora had similar density of hedges at approximately 5% of all utterances.

The annotators underwent training until they reached an 80% agreement (Krippendorff's alpha) among each other and then the corpora were split into sessions and annotated by one of three annotators each. The annotation was a count of hedges in the utterance as well as beginning and end word positions of the hedge phrase.

4 Representation

Text is tokenized using Stanford's Part of Speech Tagger. Each token consists of its text value and part of speech. In order to identify a token as being inside a hedge, these two values are checked for whether they appear in hedges that were identified by a human annotator before. If a match is made, then a reward is added to a point total for that token. If no match is made, then a penalty is removed from the token's total.

In addition to a token's part of speech and text value, the previous two tokens as well as the subsequent two tokens are considered. The evaluation of surrounding tokens as context adds or subtracts from the token's total points. After the total is added up for a token, it is compared with a threshold which determines its classification as a hedge.

The problem of finding appropriate rewards and penalties is then represented by twenty integers. These twenty integers represent a reward and penalty for each of ten features associated with a token. The token's word value as well as its part of speech, as well as the previous and subsequent tokens' word values and parts of speech are the features being examined. The optimization of the rewards and penalties is done using a genetic algorithm, with integer values for rewards and penalties being the genome of an individual in the population.

Example of an individual:
Rewards: 1 0 0 1 1 7 0 2 8 6
Penalties: 0 0 0 2 0 0 0 0 0 0

The first column represents a the token's word value; if there is a match then the reward value is added to the token's point sum. If it doesn't match, then the penalty is subtracted from the sum.

The second and third column represent the two previous tokens' word values. The fourth and fifth columns represent the subsequent tokens' word values.

The next five columns represent the same tokens in the same order as word values, except they are for part of speech.

5 Genetic Algorithm Fitness

Chat text annotated with hedges was used as the ground truth for evaluation. Using the rewards and penalties contained in the genome in an individual, all the tokens in the ground truth corpus are evaluated and then compared with the human annotation. Hits, false alarms, and misses are tallied, and then precision and recall were computed. The harmonic mean of the precision and recall was then computed (f-score) and used as the fitness of an individual. Using the f-score fitness value, the entire population was then sorted in descending order.

Various methods of comparing tokens with annotated hedge tokens were tried. Examples of these methods included lists of encountered attributes were made disregarding frequency, weighted lists containing frequency information, and finding the token with closest match were all used. Weighted lists and closest match worked the best, with weighted lists having a slightly higher recall by being tolerant of misspells and closest match having a slightly higher precision with the right threshold.

6 Genetic Algorithm Population

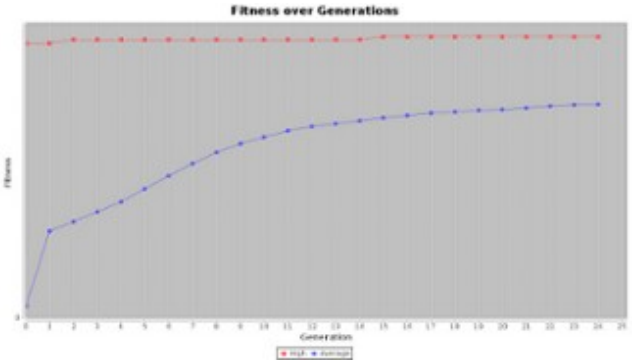
A population of 5000 individuals was used, with each individual containing a solution of 20 integers representing rewards and penalties. After the fitness was determined, the population was sorted and the top 1% formed an elite group to carry over to the next generation unmodified. The 25% right below the elite group used a uniform crossover between two random members of the population. Members to breed were chosen by roulette-wheel selection, with higher fitness increasing the chance to be chosen. The chance for a crossover to occur is 35%, meaning alleles are copied from one parent with a 35% chance to switch to the other parent after every allele copied. 35% was chosen because higher values are more disruptive and exploratory in nature while lower values are too conservative.

The lower 75% of the population was subjected to mutation as well. The chance and amount of mutation increased proportionally as lower ranks were reached. The mutation chance for every allele was 15% with the lowest scoring individual and maximum amount the allele was altered was 10.

The population was initialized by setting all individuals' alleles to 0 and then going through a round of mutation and crossover. This forms “generation 0”. Often, a good

solution is found quickly and minor improvements will occur every several generations.

Using these methods, it took approximately 20 generations for the population to converge, so 25 generations were used to confirm convergence. A graph was generated to show average fitness and the highest scoring individual's solution.



Graph showing the top fitness in the population (top) and the average fitness of the entire population (bottom). When both the top and average fitnesses converge on their own values, it is an indication that the top fitness is an optimal solution. The scale is from 0 to 0.65 here (y-axis) over 25 generations (x-axis)

7 Pruning and Weighting

Pruning

Certain hedges were removed from annotation due to uncertainty in intent. For example, "I think" was removed because annotators were unable to determine which usages were out of habit as opposed to hedge usage. "If" and modals (can/could, may/might, shall/should) were removed as well.

Emoticons were identified with a lexicon and removed as well. The placement of emoticons varied among the users and threw off the context.

Examples of pruning candidates:

Hedge	Match w/ hedge	Match w/o hedge
IMO	3	18
I think	497	17
seems	169	15
perhaps	103	13

Weighting

Creating a list for each feature containing values encountered in the training corpus has the problem of treating all entries equally. If a hedge token is found only once and is marked a hedge, then that is equivalent to another token being marked once while occurring many times. A weighting scheme was devised to take into account the frequency of the tokens as well as how often they appear as hedges. The weighting was done as follows:

frequency: percentage of all occurrences a token is marked as hedge
inverted utterance frequency (IUF):
log((utterance count) / (occurrence count))
weight:
1 + max(IUF) – IUF

When a match is made, the reward or penalty is multiplied by the weight for that feature.

8 Cross Validation

The two corpora were tested separately and then together as one big corpus. The utterances were shuffled into a random ordering and then split into 10 sections. Each section was tested using the other 9 sections as training. The results of each population:

Second Life
1 - top: 0.857 avg: 0.619
2 - top: 0.749 avg: 0.494
3 - top: 0.818 avg: 0.651
4 - top: 0.8 avg: 0.554
5 - top: 0.9 avg: 0.710
6 - top: 0.774 avg: 0.517
7 - top: 0.631 avg: 0.446
8 - top: 0.533 avg: 0.381
9 - top: 0.736 avg: 0.607
10- top: 0.666 avg: 0.533
Avg = 0.746

World of Warcraft
1 - top: 0.801 avg: 0.675
2 - top: 0.772 avg: 0.653
3 - top: 0.814 avg: 0.671
4 - top: 0.740 avg: 0.632
5 - top: 0.782 avg: 0.667
6 - top: 0.729 avg: 0.646
7 - top: 0.809 avg: 0.681
8 - top: 0.822 avg: 0.707
9 - top: 0.783 avg: 0.668
10- top: 0.762 avg: 0.652
Avg = 0.790

Combined Corpora
1 - top: 0.709 avg: 0.598
2 - top: 0.784 avg: 0.661
3 - top: 0.778 avg: 0.645
4 - top: 0.809 avg: 0.693
5 - top: 0.785 avg: 0.648
6 - top: 0.808 avg: 0.689
7 - top: 0.750 avg: 0.642
8 - top: 0.689 avg: 0.587
9 - top: 0.766 avg: 0.664
10- top: 0.758 avg: 0.651
Avg = 0.761

9 Examples of top individuals

Each row of integers are the values of current token then 2 previous tokens then 2 subsequent tokens, then POS in the same order; the top score is the best performing f-score of an individual, the average score is the average f-score of the entire population.

Weighted lists, SL train, WoW 5 session test
Top Score: 0.806
Average Score: 0.615
Rewards: 8 2 0 2 0 0 0 2 7 0
Penalties: 0 0 0 0 0 0 10 0 0 1

Weighted lists, WoW 5 session train, SL test
Top Score: 0.734
Average Score: 0.458
Rewards: 1 0 1 1 1 10 0 0 6 4
Penalties: 13 0 0 0 1 0 5 0 0 0

10 Conclusion

Most of the highest scoring individuals appeared to have non-zero values in the same places, although the exact values vary somewhat. The exploratory nature of a genetic algorithm should result in variation wherever it doesn't detract from the fitness, so if a penalty is in the right place, it might not matter if it higher than it needs to be. Alternatively, consistent zero values indicate that feature actually hurts if included in the matching algorithm.

It appears the features that matter in determining whether a token is inside a hedge (when comparing tokens with weighted lists) are:
1) The token's word value matching rewarded
2) The token's word value mismatch penalized
3) The token's following word value matching rewarded
4) The token's following word POS matching rewarded
5) The token's following word POS mismatch penalized

Using the top individual, a decent tagger for hedges was developed. Further improvements have been made by creating a “hedge stop word” list to exclude words that aren't hedges often, and a list of hedge words can be introduced into the training corpus with varying weights. In further sessions of World of Warcraft, manipulating the training data resulted in a 90% agreement with annotators.

References

Crismore, A., & Vande Kopple, W. J. (1988). Readers‟ learning from prose: The effects of hedges. Written Communication, 5, 184-202.

Piero P. Bonissone, Fellow, IEEE, Raj Subbu, Senior Member, IEEE, Neil Eklund, Member, IEEE, and Thomas R. Kiehl 2006. Evolutionary Algorithms + Domain Knowledge = Real-World Evolutionary Computation: IEEE Transactions on Evolutionary Computation Vol. 10 No. 3

Durik, A.M., Britt, M.A., Reynolds, R., & Storey, J.K. (2008). The effects of hedges in persuasive arguments: A nuanced analysis of language. Journal of Language and Social Psychology, 27 (3), 217-234.

Lin Chen, and Barbara Di Eugenio 2010. A Lucene and Maximum Entropy Model Based Hedge Detection System: Association for Computational Linguistics.

Fernandes, Crestana, and Milidiú 2010. Hedge Detection using the RelHunter Approach: International Conference on Computational Natural Language Learning

Rosa Mikeal Martey, Jennifer Stromer-Galley, Mia Consalvo, Kelly Reene, Tomek Strzalkowski, Michelle Weihmann-Purcell, Jingsi Wu, Kevin Shiflett, Jaime Banks, Sharon Small and Michael Ferguson. (2011, forthcoming) Acting your age online: Identifying user age from avatar chat, appearance, and behavior. Submitted to *Journal of Computer-Mediated Communication*.

Small, Sharon, Jennifer Stromer-Galley and Tomek Strzalkowski (2011) Multi-Modal Annotation of Quest Games in Second Life. Proc. of ACL-2011, Portland.