

Detecting emotional scenes using Semantic Analysis on Subtitles

Chetan Kalyan, Min Young Kim
Stanford University

{ ckalyan, minykim } @ stanford.edu

Abstract

In this project, our aim was to detect and clip emotional scenes from movies, using natural language processing techniques on their subtitles. Our approach involved the design of a naïve counting classifier, to which we gradually added semantic knowledge. With each increment, we analyzed the change in performance and made suitable modifications to our classifier. In our second approach, we trained a maximum entropy classifier on some hand-tagged subtitles and compared the performance of the MaxEnt classifier with different types of data and pre-learned emotional words. Using these classifiers, we then created an interactive tool for searching and watching 2-3 minute clips of movies. Subsequent sections of this paper describe our approaches in more detail.

1. Introduction

Video editing has always been a tedious and repetitive task for humans. It has evaded automation due to the lack of computer-perceivable data. In this project we aim to make some progress in video editing, particularly movies, by using NLP techniques on their subtitles.

The emotional content of a dialogue is determined both by the specific words being spoken and by the semantic context of the scene. It was our belief that we would be able to detect emotional content in groups of sentences by collating the words and the context in which they occur. (by context, we refer to the tone of the dialogue)

Movie subtitles are an ideal source for detecting emotion in bodies of text, since movies have significant emotional content in them and the tone of the dialogues are quite evident from the text itself.

Due to the lack of a tagged data corpus for emotion, we implemented a rather different approach for training data, by classifying words occurring in sentences rather than the sentences themselves. This allowed us to build our model with a much cleaner data set, where each entry had a clear indication of emotion.

Subtitles were analyzed with 2 approaches : one was a naïve counting classifier which used word frequencies and some semantic constructs to predict emoting sentences. Another was a maximum entropy classifier which analyzed unigrams within sentences, with emoting words as input features.

We found that with sufficient semantic constructs, the naïve counting classifier outperformed the maxent classifier in prediction. However, the maxent classifier seemed to perform better in cases where the emotion was not completely evident from the words. We describe both these classifiers and the datasets we used in subsequent sections.

2. Data

We had to acquire data from publicly available sources, and this limited us to movies released in the recent past and whose subtitles have been released to the public by their production studios (or by enterprising private translators). We chose the following 8 movies for our study: Gladiator, Love Actually, Remember the Titans, the X-men trilogy, Transformers and Troy. These movies were chosen mainly for their simpler English (we did not want to deal with Shakespearean English words) and for the high emotional content in them.

We hand-tagged groups of sentences to act as a gold standard. On average, each movie had about 1000 sentences, so we hand-tagged about 8000 sentences to act as our data set. While performing hand-tagging, we took care to tag the scene based on the emotional content of the scene and not on the words alone. This allowed us to have high confidence in our accuracy measurements.

We chose to tag groups of sentences rather than single ones since we observed that the emotional content of a group of sentences was higher than that of a single one. We chose to group in sizes of 20 sentences because we wanted to collect scenes, and assuming that 1 dialogue covers 1 sentence and each sentence takes an average of 10 seconds to be spoken, 20 sentences is about 3 minutes of the film, which can be readily called a scene/clip.

We identified a set of emotions which could be detected easily and which could comprehensively represent the range of human emotions. We initially used the 9 emotions identified in ancient Indian drama and literature, the Navarasas [1] . However, we later found that only a few emotions among the 9 were widely spread and hence we consolidated our list into the following 5 emotions:

Anger, Happiness, Love, Sadness and Tension.

In order to find the emotion of a group of sentences, we decided to use the words occurring in them as a guide. We then created an exhaustive list of emotional words (about 1200 words) using the FrameNet[2] frame tags. This gave us a very clean training data set with very little

chance for ambiguities, as against having sentences denoting some emotion and the extraneous words also being given weightage.

The FrameNet data has a very basic part of speech tagging, in which the word can be any one of verb, noun, adjective or preposition. Since we wanted to use these parts of speech, we initially worked with the Stanford Part of Speech Tagger [3], which satisfied our need for a reliable and fast tagger. Due to the close resemblance between our data and that of the Penn Treebank, we trained the tagger with the Penn Treebank before tagging our sentences. We found this to be a perfectly good fit for tagging and saw no errors in the words being tagged.

While tagging gave us the part of speech, we recognized that words in the sentences need to be stemmed to improve accuracy. We hence implemented the Porter Stemmer algorithm [4] to stem words occurring in various tenses and forms. These stemmed words were then compared with the lexicon whose words were also stemmed.

Finally, we evaluated our outputs by calculating precision scores for our data. We observed that recall is not very important for our experiments, since classification scores do not reflect the completeness of data. All scores reported in this paper are hence precision scores.

3. Techniques(Classifiers)

3.1. Naïve emotion count classifier

The concept behind a naive counting classifier was to verify if non-semantic information could still be used for accurate classification. In its most basic form, for each group of 20 sentences, the classifier finds all emotion words and assigns an emotion based on the most frequently occurring emotion.

We created a lexicon of emotional words, with each word allowed to contain only one emotion. However, a word could have different meanings/emotions depending on its usage in a sentence. A case in point is the word *order*- As a noun, it does not signify any emotion, but as a verb it can convey tension. It is hence imperative to know the part of speech of each word, which is why we used the Part of Speech tagger. Once populated, this lexicon was used as a lookup table by the classifier and each occurrence of an emoting word was added to the bag of words for that group of sentences. Thus, we created a sophisticated bag of words model as a base classifier.

As expected, this classifier performs very poorly. We saw scores as low as 39% for some sets of data with this classifier. Without any real semantic knowledge of the group, it is difficult to determine what constitutes a genuine emotion and what doesn't. For this reason, we made the following improvements:

1. We reasoned that we can improve accuracy of the classifier by reducing the number of classes we use. We thus reduced the number of emotions from 9 to 5, namely angry, happy, love, tense and sad. A lack of any of these emotions in a group would be classified as a non-emotional sentence. Just this change improved our accuracy to about 42%, and this could be explained by the fewer number of options the classifier now had during classification.

2. We inferred that just a single occurrence of an emoting word is not strong enough to classify the group with that emotion. We hence introduced a sensitivity measure to the classifier, allowing a group to be tagged only if an emotion occurs more than a pre-set number of times. This gave us a lot of flexibility to tune our parameters, and this improved accuracy by 5-8%, depending on the dataset.
3. Inferring the semantic meaning of a sentence turned out to be surprisingly simple. We observed that the tone of a dialogue is heavily dependent on the way it is written. So, a sentence like “*let us run there*” does not convey the urgency that “*let us run there!!*” does. We built a punctuation detecting module which changed the weight of the corresponding emotion depending on the way the sentence was constructed. This also gained us a significant improvement in classification accuracy, ranging between 8 - 15%.
4. Negations: we expected this to be a bottleneck in our classification, but since we had a large group of sentences, we noticed that even if one sentence had a negated emotion, subsequent sentences conveyed the same meaning in a more direct way. So, even though we lost some accuracy with negations like “not good”, it was buffered by subsequent emotional words in the group.
5. Dependencies within sentences: We used the Stanford Parser to find all dependencies within a sentence and increase the weight of an emotion based on the dependency. So, a phrase such as “*this is so nice!*” would have a higher happiness quotient due to the dependency between the preposition *so* and the adjective *nice*. However, since our bag of words method takes a group of about 10-15 emoting words in each clip, we found that there was no significant improvement in classification with dependencies. We also reasoned that this may be due to the lack of distant dependencies in colloquial English. Since film dialogues are short (less than 6 words on average) and the weight of the emotion was already augmented by the punctuation detecting module, we saw slight improvements in only a few data sets with dependency working.

With all these augments in place, we still found that a single set of parameters were not achieving uniformly high classification rates. We found that for some movies such as Remember the Titans, we had to be very sensitive to each emoting word, since dialogues were longer and the language more involved. For films such as Gladiator, we found that a lower sensitivity resulted in much higher classification accuracies. From this, we found that the genre of the movie is a distinguishing feature for setting the sensitivity parameter. Once we grouped the movies into “high emotion” (Gladiator, Troy etc.) and “low emotion”(Remember the Titans, X-Men 1, X-Men2 etc.), we found that we could use similar sensitivity parameters for each group and still maintain reasonable classification accuracy.

Thus, the naïve emotion classifier detected emotion reliably most of the time, with an average accuracy of 62%, and a highest score of 80% when all augments were added. Detailed results are reported in the results section.

Results

```
[It, 's, been, arranged, ., We, who, are, about, to, die, salute, you, ., We, 're, with, you, ., Maximus, !, Pull, !, Pull, !, Pull, !, Loose, !, Loose, !, -, Gut, him, !, -, Kill, !, Kill, !, Kill, !, Kill, !, Kill, !, Kill, !, Kill, !, Kill, !, Maximus, the, Merciful, !, Forward, ., guards, !]
group: [4.0, 4.0, 4.0, 4.0, 4.0, 4.0, 4.0, 4.0, 4.0, 4.0, 4.0, 4.0, 4.0, 4.0, 4.0, 4.0, 4.0, 4.0]
emotion for above block = tense
```

[Sample results with Gladiator.srt]

Movie	Basic Classifier (%)	With enhanced features (%)
Gladiator	52	80
Troy	46	62
Love Actually	26	42
X-Men 1	38	60
X-Men 2	39	60
Transformers	42	70
Remember the Titans cd 1	44	66
Remember the Titans cd 2	42	64

* all percentages rounded off to nearest integer

[Accuracies with several movie subtitles]

From the table we see that the worst performance is seen for the comedy movie Love Actually. The classifier failed miserably for this movie since the entire movie is composed of indirect references to emotion, which are immediately evident to a human, but impossible for a machine. We realized that we still have some way to go with our classifier before we can begin to classify such “intelligent” movies, but the other results look promising.

3.2. MaxEnt classifier

We used MaxEnt classifier to label groups of sentences appearing in subtitles. Since we had implemented a basic MaxEnt classifier for programming assignment 2, we utilized the existing model and improved it to be suitable for our purposes. In general, MaxEnt classifier is used for sequence labeling-it classifies data with possible labels. MaxEnt classifier produces a probability distribution over the possible labels by using its feature functions and objective function.

As described in section 2, we have a list of emotions and each emotion contains words directly related to the emotion. We used these words as feature lists. So, all six categories of emotion have features with words representing the emotion. Since the same word can have different meaning when it appears in different location in grammatical order, we added words to the feature list depending on their usage. These features are used when training the MaxEnt Markov Model. Then, we classified each sentence into one of the emotion categories. However, since the sentences appearing in subtitles tend to be relatively short, the emoting word has an important role in classification and it is possible to misclassify the sentence if the word was meant differently than we expected. Thus, to make the classifier more reliable, we grouped sentences by chunking them into groups of 20. We cropped the subtitles by 20 time frames so that every group has around 20~25 sentences. With accumulated probabilities of those sentences, the model outputs an emotion for the group.

Also, since we consider unigrams only, the classifier may not detect a negated emotion, which can result in misclassification. One way that we found it useful for reducing this problem is to use Stanford parser[5]. Before the testing step, we parse all sentences and tag

every word with its grammatical property. Then, when we test those sentences, we find words which have the most powerful emotion and look for other words having negative meaning such as ‘not’, ‘never’, ‘doesn’t’, and so on. We created a list of these words and if we found them in the sentence, we checked if there is a direct relation to the emoting word. If there is one, we negate the classification. For example, if the sentence was initially classified as happy, we change it to sad. The best example of this problem is shown below:

S: not too good not too good
 Emotion of current sentence: happy
 MaxProb: [1.0 : 0.5098828054661357]
 sum_probCounter: [1.0 : 0.18236]
 ...
Duration: 1493.0 to 1569.0
Emotion: happy Gold: sad

The example above is a part of classification process and we can easily see that the sentence **S** is classified as a happy sentence because it contains ‘good’, which is a *happy* word. However, this sentence should actually be classified as a *sad* sentence. The classifier failed to classify this sentence because it only considers unigrams. Thus, to classify **S** appropriately, first we should know if the word ‘good’ is used as a verb, and then look for RB-‘not’ in this case-and if it exists, we have to classify with the opposite emotion. For some reason, this method does not work for every sentence having negative words, and is hence part of our future work.

As a result of the classification, we output time duration with emotion classified for the duration and this will be used in the GUI application we built for streaming actual movies as the result of search command. We tested the MaxEnt classifier with several movie subtitles and the results are shown below.

Results

S: give me a kiss mmm ah Emotion: love
 [Max: 2.0]
 S: man you ve done a good job coach boon Emotion: happy
 [Max: 1.0]
 S: neighbor cheer Emotion: happy
 [Max: 1.0]
 S: i won t make it ani other wai Emotion: no_emotion
 [Max: 5.0]
 S: well i got new for you Emotion: happy
 [Max: 1.0]
 sum_probCounter: [Max: 1.0]
 ...
 duration: 1410.0 to 1451.0
 Emotion: happy Gold: happy
 ...
 ...
 S: you know if you could just keep your mouth shut Emotion: tense
 [Max: 4.0]

S: no i am talk about set a good exampl Emotion: happy
 [Max: 1.0]
 S: to your grave that s your busi Emotion: tense
 [Max: 4.0]
 S: it becom mine Emotion: tense
 [Max: 4.0]
 sum_probCounter: [Max: 4.0]
 ...
 duration: 294.0 to 331.0
 Emotion: tense Gold: tense

*Every word in sentences are stemmed before doing classification..

[Sample results with remember_the_titans.srt]

Movies	# of sentences	# of groups	Accuracy (%)
Remember the Titans cd1	1532	76	78
Remember the Titans cd2	1109	55	73
Gladiator cd1	1260	62	72
Love Actually	1829	91	48
Troy	1156	57	65
X-men 1	689	35	62
X-men 2	852	42	59
Transformers	1835	92	68

* # of groups corresponds to the number of gold labels in testing data.

[Accuracies with several movie subtitles]

There are some differences in accuracies among movies and it is because of characteristics of sentences appearing in subtitles. Some movies have longer sentences in conversation making classification easier, and other movies have lots of sarcasms resulting in lower accuracies. Also, we could see that action movies contains many tense sentences, movies about love story definitely have many happy and love sentences, and also movies related with sports like ‘Remember The Titans’ has also lots of tense and happy sentences.

4. GUI

4.1. Motivation

Building a GUI application for the results from our classifiers leads to our goal for this project – editing video by entering search queries. Initially, while we were discussing topics for our project, we realized that we wanted to do something which was practical and which could be used as a real-world application. So, we chose a topic which we can utilize for the application.

Sometimes, people wants to watch certain scenes rather than watching a whole movie, especially if they have already seen it and only want to find the memorable scenes. Those scenes can easily be remembered with emotions appeared in the scenes. With our list of

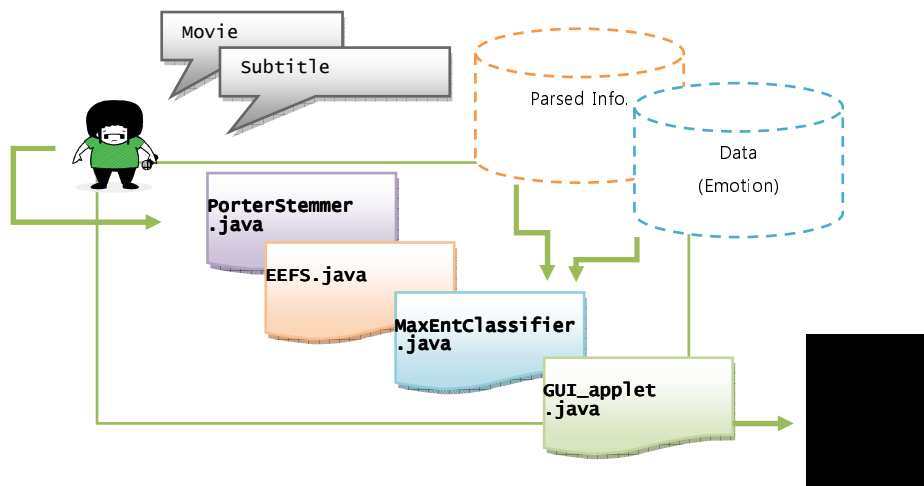
emotions and classifier, our application can search those scenes for them once they have subtitles of the movie.

4.2. Environment

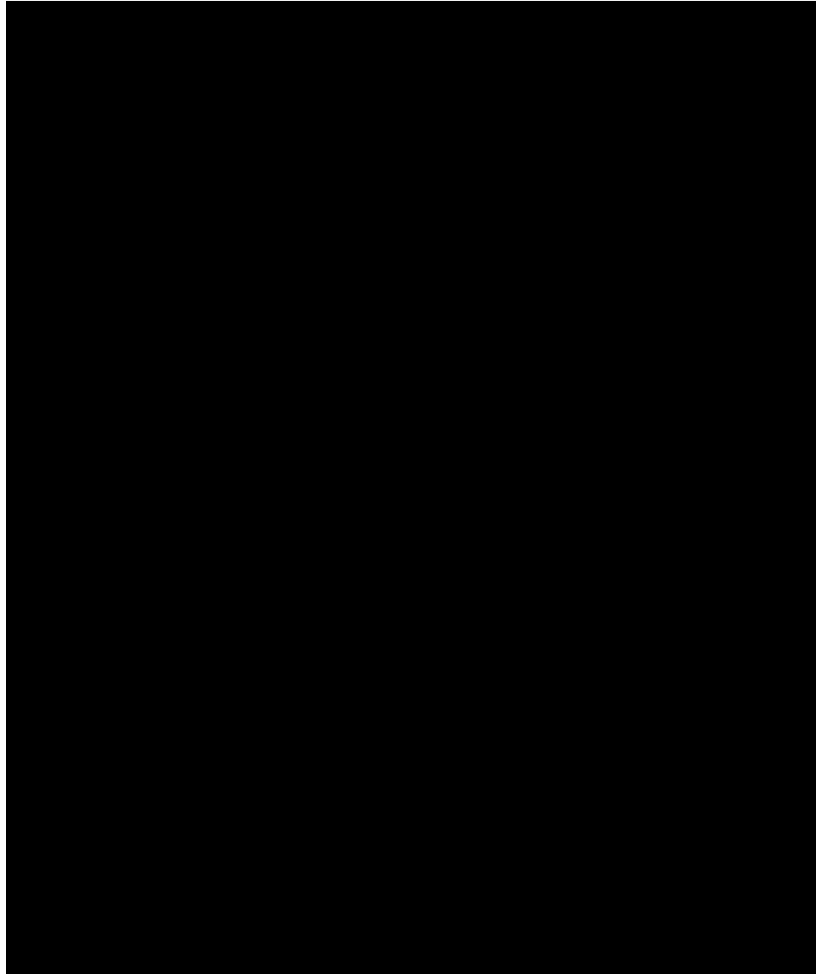
For implementation, we used AWT-GUI toolkit for Java, Swing components, and JMF-Java Media Framework-API(ver.2.1.1). Also, since JMF supports limited audio and video codecs and most of movie files are encoded with various codecs, we needed to convert our movie files into compatible ones other with appropriate codecs. Converting movie files and finding right codecs was a tedious and time-consuming process, but the results were worth the effort.

4.3. Description

The architecture and final version of the application are as follows:



[Architecture of the application]



[Final version of the application]

There are three major components in this application in detail.

4.3.1. Setting Panel

Setting		
Load a subtitle	<input type="text" value="224N_FINAL_PROJECT\Remember_the_Titans.EN_DH_1.srt"/>	<input type="button" value="Load sub..."/>
Load Movie	<input type="text" value="224N_FINAL_PROJECT\Remember.The.Titans.2000.CD1.avi"/>	<input type="button" value="Load movie.."/>
Type to search	<input type="text" value="tense"/>	<input type="button" value="Search"/>

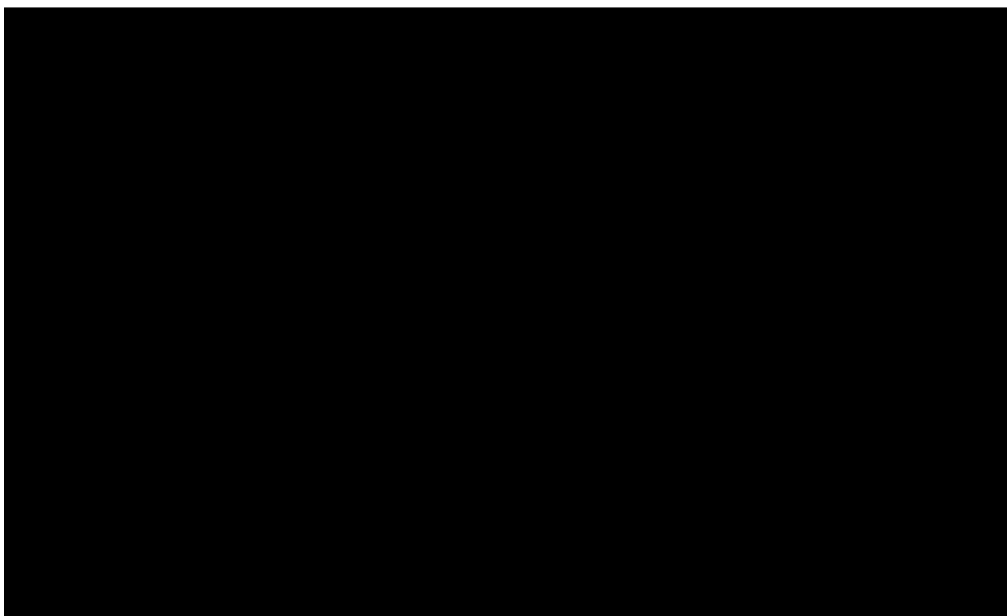
With this panel, users can load subtitles and movie files using 'file chooser'. Thus, it is not necessary to have subtitles and movies on the server. People can pick any movie which they want to extract certain scenes. After loading all files, the text field for search will be enabled and users can type their search query. Finally, when they click search, the result will be shown in below panels.

4.3.2. Debug Panel



This panel shows the result of each step of processing. Also, if there is an error captured by exception handler, the error message will also be printed in this panel so that users can easily notice why the program caused an error. This debug panel was also very useful during the implementation because of visibility.

4.3.3. Play Panel



Play panel is the main panel of the application. When an action occurs from search button in the Setting panel, it will return a list of time frames classified by the input emotion using the resources provided. If there are several scenes, those time frames will be listed in the box located in the left and the user can choose one of them to start playing the movie from that part. While the movie is playing, if the user wants to jump to another time frame, the user can do so by just clicking another time frame among the list and click the play button again, then the movie will be automatically played from that time frame.

5. Future Work

We would like to enhance the performance of our classifier with a richer semantic analysis engine, which can detect negations, dependencies and indirect references to sarcasm and humor. Our idea seems to promise a very wide array of applications, but our classifier will have to significantly improve before we can make this a commercial product.

6. References

1. Navarasas of Indian NatyaShastra: <http://www.ee.caltech.edu/~gowaikar/rand/navaras.html>
2. FrameNet: <http://framenet.icsi.berkeley.edu>
3. Stanford Part of Speech tagger: <http://nlp.stanford.edu/software/tagger.shtml>
4. Porter Stemmer: <http://tartarus.org/~martin/PorterStemmer/>
5. Stanford Parser: <http://nlp.stanford.edu/software/parser.shtml>
6. Java Media Framework: <http://java.sun.com/javase/technologies/desktop/media/jmf/>
7. Java Swing:
<http://java.sun.com/docs/books/tutorial/uiswing/examples/components/index.html#security>