Abstract

In this work we take advantage of both semantic networks provided by BabelNet and distributed word representations trained by skip-gram model with negative sampling in an effort to obtain vector representations of 8 emotional states. Subsequently, we have employed above-mentioned emotion vectors to identify emotional scenes in several contemporary movies spanning different genres and languages (Polish and English).

1 Introduction

With the rise of popularity of movie streaming and the explosion of available content it is increasingly important to find the movies which fit the diverse tastes of the viewers. In this work we introduce an unsupervised, automatic emotion annotation algorithm, which could be used for video information retrieval and recommendation systems. It is straightforward, to further extend our approach for additional task such as movie clustering.

In our work we combine hypernym/hyponym relations and synonyms of BabelNet and distributed word representation, which allows us to quickly annotate movie dialogues by reducing network communication overhead. This approach is dictated by the fact that BabelNet API calls are relatively expensive and are prohibitive in production systems.

The source code for this demo is available on GitHub.

2 Data Set

We based our analysis on OpenSubtitles2016 from the OPUS, a freely available, growing language resource of parallel corpora and related tools.

To obtain vector representation of words we trained skip-gram model on english and polish Wikipedia.

3 Algorithm Description

In this section we will briefly describe proposed approach.

3.1 Emotion Vectors

In an effort to obtain emotion vectors we have used BabelNet to construct Sense Trees with synset as nodes. Each Sense Tree represents one of the eight basic emotional states (’love’, ’happiness’, ’surprise’, ’emotionless’, ’sad’, ’disgust’, ’anger’ and ’fear’) as root and synsets of hyponyms as children. Depth of the Sense Tree has been decided to be equal to three. Subsequently, each node in the Tree has been assigned a vector representation of the synset’s lemma. Finally, we have calculated weighted average of word vectors in the Sense Tree with weights proportional to the distance from the root. Detailed description of Sense Tree and Emotion Vectors construction can be find in the algorithm listings respectively.

3.2 Sentence Vectors

For selected movies we have extracted both English and Polish subtitles and aligned them with the help of OpenSubtitles2016. Subsequently, for each line of dialogue we removed stop words and constructed sentence vectors by averaging word...
Function AddChildren(root, level)
input: A root Sense object root, A level a of recursion level

if level < maximum level then
    hs ← GetHyponyms(root) for h ∈ hs do
        if h.weight > threshold then
            root.addChild(h)
        end
    end

for ch ∈ root.children do
    AddChildren(child, level + 1)
end
end

Algorithm 1: Sense Tree Construction

vectors of the remaining words. Finally, we assigned the closest emotion vector to the sentence vector using cosine distance as a measure.

Function GeVector(tree, level)
input: A Sense Tree object tree
output: Vector representation of a tree

v ← WordToVec(tree.lemma)
w ← 1 if tree is a leaf, the inverse of the number of children otherwise.

if tree has children then
    for ch ∈ root.children do
        v ← v + w* GeVector(ch, level + 1)
    end
end

return v/level

Algorithm 2: Emotion Vector Construction

4 Results

Since subtitles with labeled emotions were not available, we decided to test our approach by plotting distribution of emotions for given periods of time for both languages and comparing them visually.

Plots were constructed in a following way: for a given movie, we grouped all dialogues in 10 sentence chunks and calculated distribution of emotion states for a given group. As a result, for each ANALYSED movie, we obtained 8 dimensional time series. Results are shown on figures 1 and 2. Courious reader can compareision of emotion for Cassablanca and Pulp Fiction in table 1.

<table>
<thead>
<tr>
<th></th>
<th>Casablanca</th>
<th>Pulp Fiction</th>
</tr>
</thead>
<tbody>
<tr>
<td>love</td>
<td>851</td>
<td>1210</td>
</tr>
<tr>
<td>sad</td>
<td>42</td>
<td>33</td>
</tr>
<tr>
<td>emotionless</td>
<td>694</td>
<td>121</td>
</tr>
<tr>
<td>disgust</td>
<td>33</td>
<td>17</td>
</tr>
<tr>
<td>anger</td>
<td>56</td>
<td>61</td>
</tr>
<tr>
<td>surprise</td>
<td>320</td>
<td>234</td>
</tr>
<tr>
<td>fear</td>
<td>241</td>
<td>168</td>
</tr>
<tr>
<td>happiness</td>
<td>68</td>
<td>50</td>
</tr>
</tbody>
</table>

sum 1744 1744 1817 1817

5 Conclusions and Further Work

Although, for some sentences algorithm is able to accurately match the appropriate emotion class, it seems that in general the results are not stable across languages. For each English movie we have evaluated our algorithm on, the most common emotion was love. In Polish language however the it was ‘emotionless’ that dominated the movie.

It would be interesting take the advantage of hyponym/hypernym relations for measuring the distance between a specific emotional word and the emotional concept with which it is associated (if it is).

References


Figure 1: Distribution of emotions over time in '12 Angry Men'

Figure 2: Distribution of emotions over time in 'Cassablanca'

Figure 3: Distribution of emotions over time in 'Pulp Fiction'