Tweet-to-Scene Generation: WordsEye with Twitter

Mandy Korpusik, Professor Julia Hirschberg
Department of Computer Science, Columbia University
Distributed REU
August 1, 2012

I. Abstract

WordsEye is a text-to-scene generation program that creates a 3D scene from descriptive English text. For example, the sentence "The cat is on the chair" results in a scene with a cat sitting on a chair. Twitter provides a way to reach many potential WordsEye users, due to the increasing popularity of social media and the large number of Twitter users. For this reason, I worked on creating and uploading 3D WordsEye scenes from tweets. Generating descriptive sentences from tweets is an interesting challenge because tweets are limited to 140 characters and contain Internet slang, emoticons, misspellings, and Twitter lingo such as hashtags and retweets. For simplicity, I focused on filtering tweets for only four verbs (like, love, hate, and see), as well several different emotions such as happy or sad. I used part of speech tags to extract the subject and object of each verb/emotion, WordsEye attributes such as colors or textures, and whether or not the verb or emotion was negated (i.e. "do not like"). Using this information, I generated text that is compatible with WordsEye. For example, the tweet "Yay!" would become "The person is happy." The final result is a 3D WordsEye image uploaded to Twitpic and a tweet posted to the WordsEye Twitter account. I wrote several Python scripts to automate the entire process.

II. Introduction

WordsEye is a text-to-scene generation project started by Bob Coyne, which generates 3D scenes from English text. An example of a 3D scene, along with its corresponding text, is shown in Fig. 1. The user can choose a specific object from several options, as well as specify the texture, color, and size of objects; the placement of lights; and the position of the camera. However, there is the limitation that the text must be very specific and grammatically correct: you cannot use simply any text to generate a scene, or it may result in an error.

Twitter is a good way to reach millions of people because there are over 190 million Twitter users and over 65 million tweets every day (Jiang, Zhou, & Zhao, 2011), and the number of users is continually increasing as the popularity of social media grows. Communication is quick and easy. Users have instantaneous access to current events and to others' openly expressed opinions. For all these reasons, Twitter provides a way to reach many potential WordsEye users,

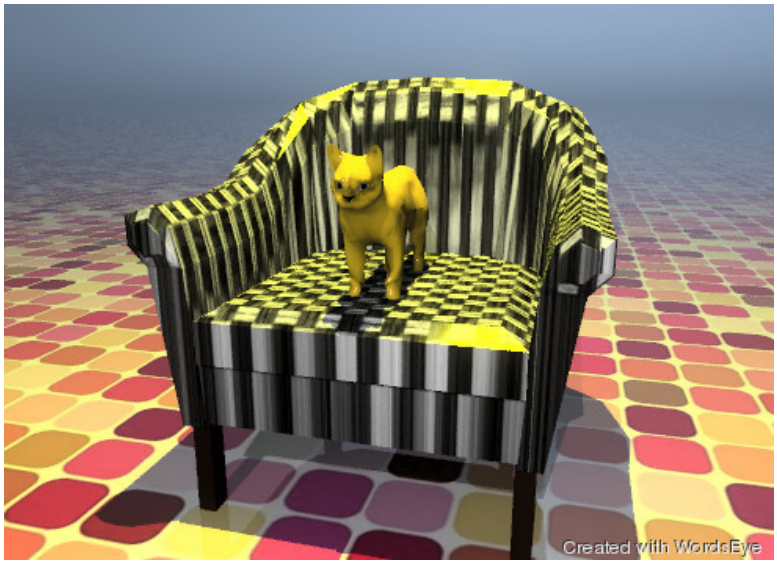

Fig. 1. *A sample WordsEye scene. The text used to generate this scene is: "the orange cat is on the chair. the chair is white. the ground has a design texture. the ground is shiny. the huge yellow illuminator is 7 feet above the cat."*

which is the motivation for combining WordsEye with Twitter.

The language used in Twitter is interesting for several reasons. First, tweets are very short. They are limited to only 140 characters. Tweets also contain unique Twitter lingo, such as hashtags, @usernames, and retweets. Tweets contain Internet slang, such as "lol" for "laugh out loud" or "omg" for "oh my God". They have emoticons, which are punctuation symbols arranged in such a way as to represent faces expressing particular emotions. Words are often misspelled, and tweets are rarely grammatically correct. Users often swear or use slang, and they can include links to websites. Finally, tweets are often not even in English.

My project was to filter tweets for those that contain simple verbs or emotions, extract the information necessary to generate a scene with WordsEye, and to post a tweet with the resulting WordsEye scene. Fig. 2 shows the overall process, starting with finding tweets on Twitter. My Python script processes and filters the tweet, and generates English text that is input into WordsEye. The 3D scene generated from WordsEye is finally uploaded to Twitpic.

III. Related Work

Twitter has recently become an extremely popular research topic. Researchers have worked on natural language processing (NLP), sentiment analysis, and topic extraction of tweets. Businesses and government are using Twitter as sources of people's opinions about various products and political issues.

Several researchers have specifically looked into how to process and normalize tweets. (Han & Baldwin, 2011) describe an algorithm for replacing misspelled (or ill-formed) words with the

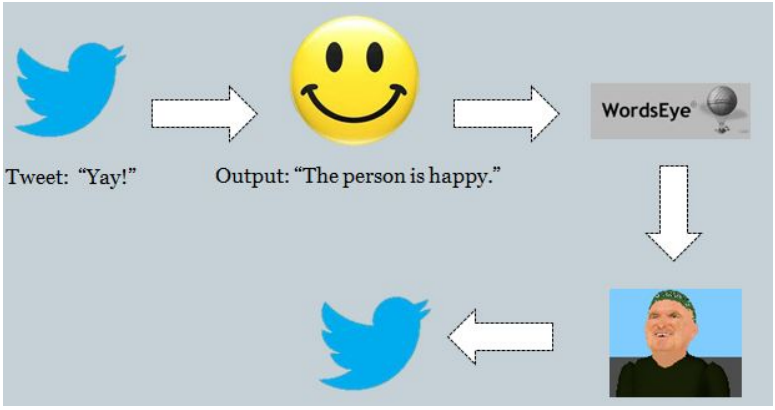

Fig. 2. *A diagram of the overall process for obtaining, processing, and filtering tweets, as well as using WordsEye to generate scenes and uploading scenes to Twitpic.*

correct English words, a process called lexical normalization. (Liu, Weng, Wang, & Liu, 2011) explain how to normalize text messages, which are similar to tweets because both are short in length and contain Internet acronyms and emoticons. Part of speech tagging for Twitter is described in detail by (Gimpel, et al., 2011).

Many researchers have also worked on opinion mining with Twitter, or sentiment classification of tweets, as done by (Jiang, Zhou, & Zhao, 2011). Although not specific to Twitter, (Kim & Hovy, 2006) worked on extracting opinions, opinion holders, and the topic of the opinions in online news media text. Similarly, (Das & Bandyopadhyay, 2010) extracted emotion topics from blog sentences, which is just a different form of online text. (Stoyanov & Cardie) describe the process of extracting the topics of opinions. Finally, (Gonzalez-Ibanez, Muresan, & Wacholder, 2011) worked on identifying sarcasm in tweets, which is even more challenging than identifying obvious emotions.

IV. Method

There are a couple options for obtaining publicly released tweets using the Python Twitter API. The first is simply getting 20 random tweets a minute. The other is getting 100 tweets a minute that are filtered for a specific keyword. However, we decided to use tweets that posted a photo because the photo could be inserted into the WordsEye scene. This method requires reading the 40 most recently posted Twitter images from the live feed on twitcaps.com and extracting the tweet and username associated with each image. Unfortunately, this option provides less information about the user who posted the tweet than is provided by the Twitter API. Fig. 3 shows an example of an image and its associated tweet which would appear on twitcaps.com.

The next step after obtaining the tweets is to normalize them, or to standardize the language of the tweets so that the Twitter lingo is replaced with standard English. For example, emoticons are replaced with the emotion words they represent, and acronyms are replaced with the full words. Hashtags and @usernames are removed from the tweet, but are saved for the purpose

Fig. 3. *A photo and tweet posted to Twitter, taken from twitcaps.com/feed.*

of tweeting back. This step also includes searching for unknown words that contain multiple repeated letters and testing whether the word is known when the repeated letters are replaced with only a single letter or a letter that is only repeated once. For example, ”wooooooooow” would be replaced with ”wow”.

Tweets that are not English, or contain greater than 50% unknown words (i.e. they do not appear in a list of English words), are immediately thrown out. We decided to only use tweets that have one of the four verbs ”like”, ”love”, ”hate”, and ”see” or that have an emotion, since emotions are commonly expressed through tweets. I created several emotion categories, including happy, sad, and angry, and assigned emotion words from LIWC (the Linguistic Inquiry and Word Count) to the appropriate category. I could then check whether tweets mentioned a word from any of the emotion categories. We have iconic representations for emotions, as well as other abstract ideas. For example, like and love are represented by hearts, while hate is represented by a heart inside a not sign.

For each verb and emotion, I used part of speech (POS) tags to extract the subject and object, attributes of the subject and object (if any), and whether or not the verb or emotion was negated. For verbs, I also checked whether the subject or object was experiencing an emotion. The subject and object attributes can be known WordsEye colors/textures/national flags, or they could be unknown to WordsEye and displayed in the scene with text above the corresponding subject or object. If the verb or emotion is negated, it can be represented by a not sign around the entire scene. If the subject or object is a gendered word such as ”he” or ”she”, or is the author of the tweet, I turned the subject into ”man” or ”woman”, since these are known WordsEye objects. In order to determine the gender of the author, I tried to match the username with a name on a list of baby names for boys and girls. If the search was unsuccessful, I represented the author

as a generic ”person”, which is also a recognized WordsEye object.

Certain celebrities, such as Justin Bieber, are often mentioned in tweets. I used the program FaceGen to create 3D models of the faces of celebrities and politicians mentioned most often on Twitter. If one of these people is mentioned in a tweet, then the model of that person can be included in a WordsEye scene. An example of some of the 3D faces I modeled by fitting to photos found on Google Images is shown in Fig. 4.

Fig. 4. *Examples of celebrity faces modeled in 3D using FaceGen.*

Extracting the subject and object of verbs required two algorithms, one for active verbs and one for passive verbs. This is because the subject comes before an active verb, and the object comes after an active verb, whereas the placement is reversed for passive verbs. In the example ”The woman likes the flowers”, the subject is ”woman”, and the object is ”flowers” because ”likes” is an active verb. However, in the sentence ”The flowers are liked by the woman”, the object is still ”flowers” even though it comes before the verb.

Extracting the subject and object of emotional adjectives also required two algorithms, one for direct emotion words and one for implied emotion words. I noticed that there are emotion words which directly describe how a person is feeling, whereas other types of emotion words imply that the speaker is feeling a certain way. In the examples ”I am so happy” and ”He is bored”, the emotion words ”happy” and ”bored” directly describe how the subject of the emotion is feeling. However, in the examples ”This movie is dull” and ”The view is beautiful!”, the word ”dull” implies that the speaker is bored by the movie, and the word ”beautiful” and the exclamation point imply that the speaker is happily excited about the view.

Finally, I realized I needed to also specify whether an emotion verb was active or passive (note that these terms do not refer to the part of speech of the verb, but to how the placement of the subject and object relate to the verb). For example, in the sentence ”She loves music”, the subject of the emotion (or the person experiencing the emotion) is the same as the subject of the verb, and the object of the emotion (or whatever is causing the emotion to be experienced)

is the same as the object of the verb. However, in the sentence ”That sound annoys me”, even though the verb ”annoys” would be tagged as an active verb, the subject of the emotion comes after the verb, while the object causing the emotion comes before the verb. For emotion verbs such as ”annoys”, I had to use the algorithm for extracting the topic of a passive verb, even though ”annoys” is technically an active verb.

Using the extracted information, I generated descriptive, grammatical sentences that could be input into WordsEye. The patterns shown in Fig. 5 are how I turned the verb, emotion, and photo information into grammatically correct sentences that are compatible with WordsEye. The verbs would change tense depending on whether or not there was negation and whether the subject was singular or plural. I also manually assigned each emotion word its correct preposition. For example, you might say ”happy about X”, but ”bored by Y”.

Verbs: “The [subject attribute] [subject] [verb] the [object attribute] [object].”

Emotions: “The [subject attribute] [subject] is [emotion] [preposition] the [object attribute] [object].”

Photos: “There is a small [photo filename] billboard.”

Fig. 5. *The three templates I used for generating sentences with the extracted verb and emotion information.*

V. Results

Fig. 6 shows an example of a tweet that might be obtained randomly and would be processed and filtered. The tweet contains several hashtags, as well as a username and a link. Normalizing the tweet would result in ”It’s a nice chill afternoon. Happy with the races today.” This tweet would pass through the filter because it contains the emotion words ”nice” and ”happy”. The output sentence would include the subject and object with the appropriate emotion: ”The person is happy about the chill afternoon.”

Fig. 6. *A sample tweet.*

An example of a sentence that could be generated by my Python script is ”The woman hates the platypus.” Inputting this sentence into WordsEye generates the 3D scene shown in Fig. 7. The platypus is placed upon a table in front of the iconic representation for hate. As shown in Fig. 8, the sentence ”The girl is excited about Justin Bieber” results in a scene with Justin Bieber inside a thought bubble above a girl’s head. Although Justin Bieber’s face is too small to recognize, the scene does use the FaceGen model of his face.

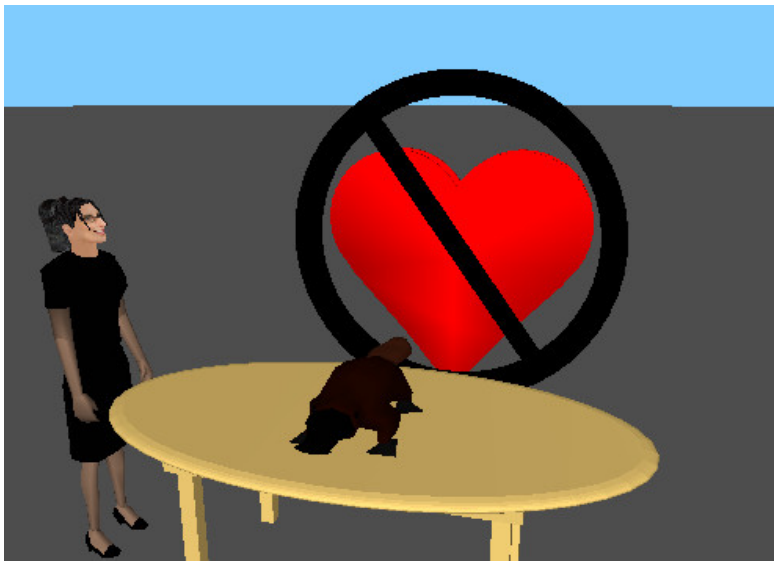

Fig. 7. *A WordsEye scene generated from the sentence ”The woman hates the platypus”.*

Fig. 8. *A WordsEye scene generated from the sentence ”The girl is excited about Justin Bieber”.*

The way we incorporated photos into scenes was by inputting the sentence ”There is a small [filename] billboard” into WordsEye. WordsEye would place the photo with the corresponding filename onto a billboard in the background of the scene, as shown in Fig. 9. Finally, the sentence ”Barack Obama is angry at the Republican elephant” would use the FaceGen model of Barack

Obama and shape it into an angry expression (see Fig. 10).

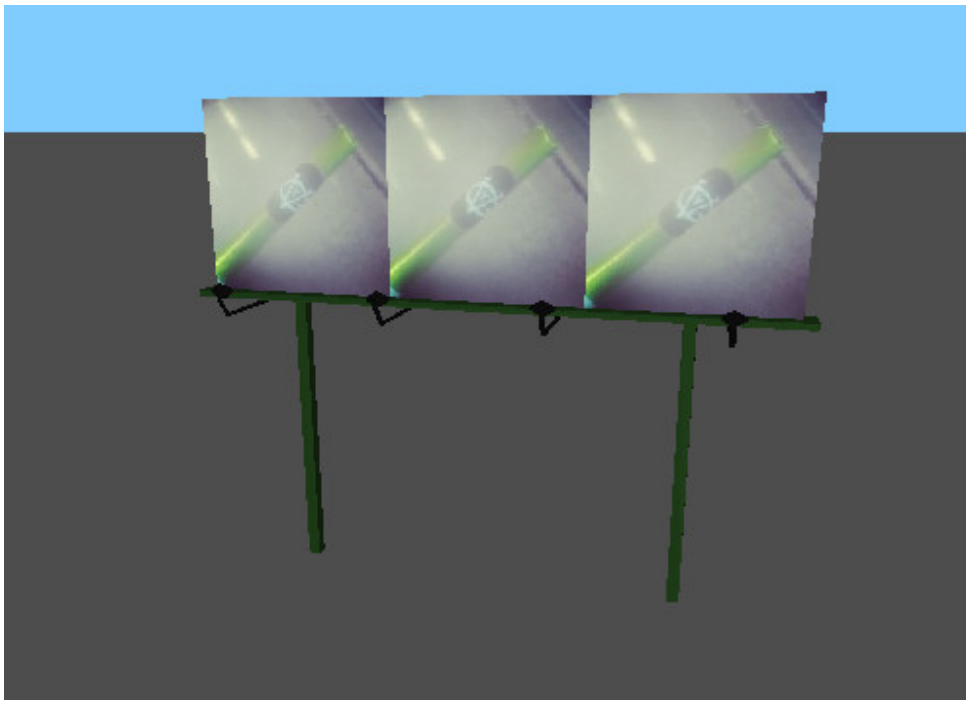

Fig. 9. *A WordsEye scene generated from the sentence "There is a [pic/1.jpg] billboard", where pic/1.jpg is the filename of a photo.*

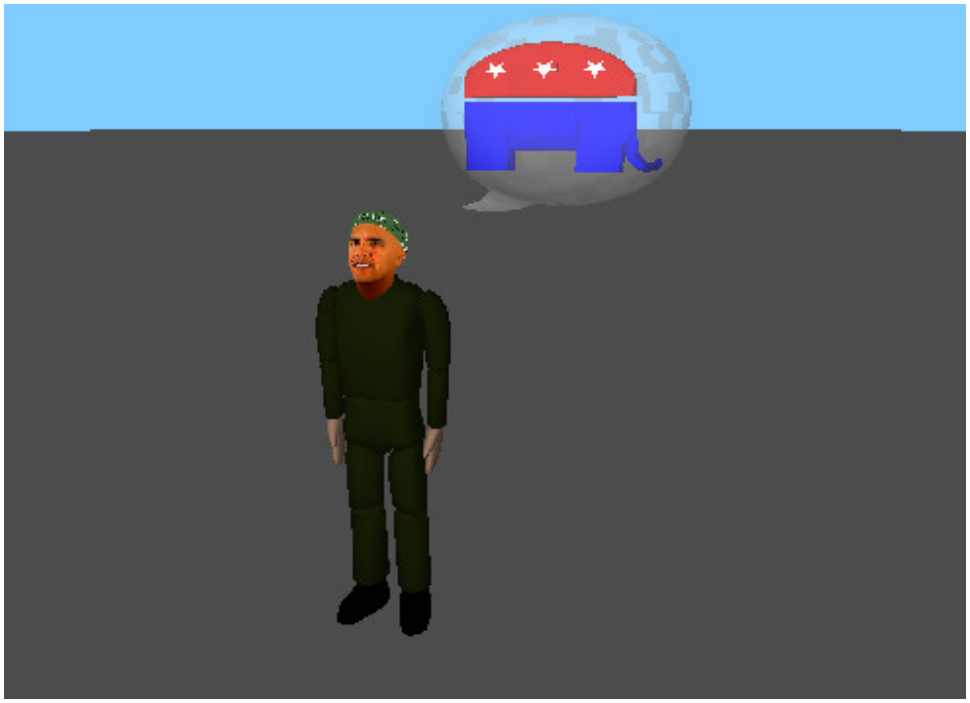

Fig. 10. *A WordsEye scene generated from the sentence "Barack Obama is angry at the Republican elephant."*

Once WordsEye generates a 3D scene, my Python script automatically uploads the image to Twitpic. The benefits of using Twitpic to host the images are that we can include a message not limited to 140 characters and we can use the Twitpic API to do traffic analysis (i.e. determine

the number of times a specific image has been viewed). Fig. 11 shows an example of what the uploaded Twitpic image and message look like. In the message, I describe the process of taking a tweet and transforming it. I also include a link to WordsEye and suggest trying it out.

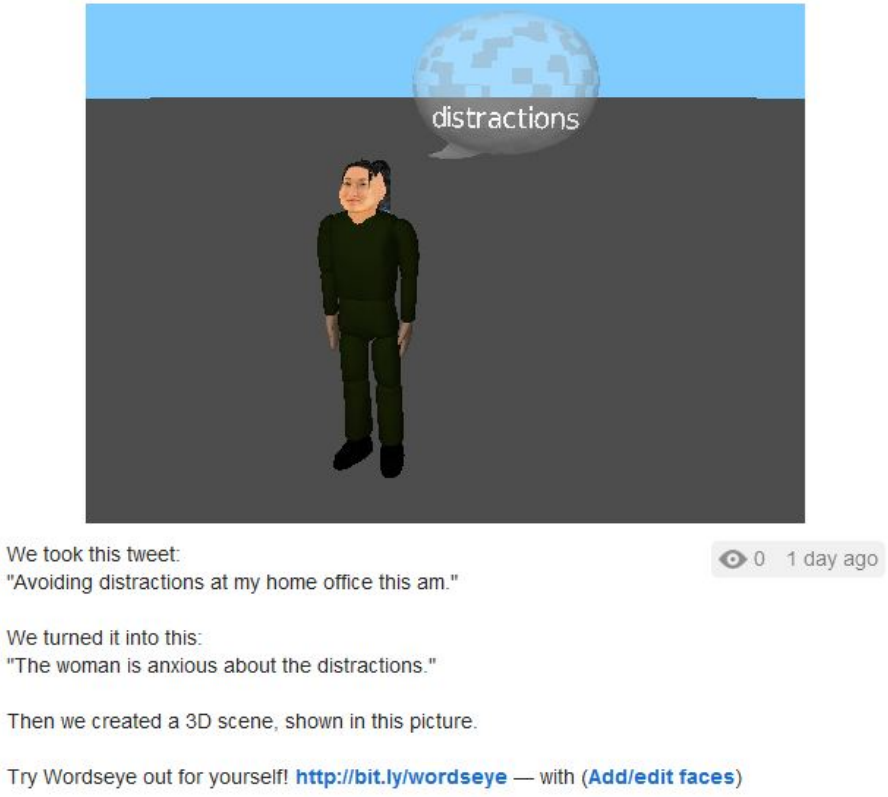

Fig. 11. *A WordsEye scene and the associated message uploaded to the WordsEye Twitpic account.*

The final step of the process is to post a tweet to the WordsEye Twitter account I created. The tweet contains a link to the Twitpic image, as well as a WordsEye hashtag (and the username of the original tweet's author if we have high confidence in the scene). An example is shown in Fig. 12.

Fig. 12. *A tweet posted to the WordsEye Twitter account containing a link to a Twitpic image and the WordsEye hashtag.*

VI. Conclusions and Future Work

The first pass of the process has been successfully accomplished, and we can now run a single Python script to automatically filter tweets, create WordsEye scenes, upload Twitpic images, and post tweets. Approximately 25% of all random tweets are filtered, and about 75% of those

filtered tweets output logical sentences.

However, there are still many improvements that can be made to this whole process. The WordsEye scenes can be made more visually appealing by incorporating WordsEye locations such as a living room or office, as well as the time of day. For example, the scene could be dark at night, or the position of the sun can change depending on the time of day. There are many WordsEye verbs that are currently being annotated and which could be added to the existing four verbs. This would add variety to the types of scenes we can generate. The algorithm for extracting the subject and object of the verbs and emotions can be improved through pattern matching or parsing. Finally, the process requires further testing and debugging.

VII. References

Das, D., & Bandyopadhyay, S. (2010). Extracting Emotion Topics from Blog Sentences – Use of Voting from Multi-Engine Supervised Classifiers. SMUC, 119-126.

Gimpel, K., Schneider, N., O'Connor, B., Das, D., Mills, D., Eisenstein, J., et al. (2011). Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, 42-47.

Gonzalez-Ibanez, R., Muresan, S., & Wacholder, N. (2011). Identifying Sarcasm in Twitter: A Closer Look. Proceedings of the 49th Annual meeting of the Association for Computational Linguistics, 581-586.

Han, B., & Baldwin, T. (2011). Lexical Normalisation of Short Text Messages: Makn Sens a #twitter. Proceedings of the 49th Annual meeting of the Association for Computational Linguistics, 368-378.

Jiang, L., Zhou, M., & Zhao, T. (2011). Target-dependent Twitter Sentiment Classification. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, 151-160.

Kim, S.-M., & Hovy, E. (2006). Extracting Opinions, Opinion Holders, and Topics Expressed in Online News Media Text. Proceedings of the Workshop on Sentiment and Subjectivity in Text, 1-8.

Liu, F., Weng, F., Wang, B., & Liu, Y. (2011). Insertion, Deletion, or Substitution? Normalizing Text Messages without Pre-categorization nor Supervision. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistis, 71-76.

Stoyanov, V., & Cardie, C. (n.d.). Annotating Topics of Opinions.