Text Summarization & Sentiment Analysis of Food Reviews

Janardhana Swamy Adapa UIN: 827006191
Prakhar Mohan UIN: 927008520
Sai Eswar Epuri UIN: 927008613
Venkata Naidu Marineni UIN: 727004731

1. Introduction

In today's digital world, we have lot of text information and it is important for us to store this information if possible in shorter format so that it would be easier to understand and analyze the data. One of the best ways of reducing size of the data is to extract summaries of data and then use the summaries instead of entire text in our predictive models. We have explored the same in our current problem in which we represent reviews of amazon products with just summaries and their ratings. Summaries capture sentiment of the review whether it is good or bad and rating gives how good or bad the product is. To analyze the loss of relevant information, we compared the predicted ratings of the product given the review and product given the summary.

2. Related Work

Multiple techniques have been proposed for text summarization like Graph based ranking[10], in which multiple graph algorithms like PageRank, Undirected Graphs and Weighted Graphs are used for automatic sentence extraction. Attentional Encoder-Decoder Recurrent Neural Networks[11] have also been used for text summarization.

Classifying sentence using machine learning techniques has been studied extensively. It has been done by Naive Bayes, and multi-variate Bernoulli model[5] which uses Bayesian Network without dependencies between words and word features. Other methods which can be used are SVM, kNN as mentioned in [6]. Text classification can also be done using CNN[3], by applying 1-D Convolutions on word embeddings. Bidirectional LSTM[8] can also be used for this task.

3. Dataset

The Dataset used for this project is a collection of 568,453 reviews of amazon food products[7]. The columns of interest are 'Text' and 'Score' in the dataset. The column 'Text' has the reviews of the food products and the column 'Score' has the rating of the food products ranging from 1 to 5. The length of review varies from 3 to 3432 words. The distribution of the ratings is shown in table 1.

Rating	Number of reviews
1	52268
2	29769
3	42640
4	80655
5	363122

Table 1: Distribution of reviews according to ratings.

4. Text Summarization

-Sai Eswar Epuri

In the present world where we have lot of textual information, it is very important to have a shorter/summarized version of data that would help in making understanding faster, easier and would help in eliminating redundant information.

4.1. Different Approaches

There are two different approaches for summarizing the given text. The **Extractive** approach extracts important statements or phrases from the sentence and concatenate them to get the summary. It is similar to highlighting the important parts of given text and returning it as output. In **Abstractive** approach, new statements are generated based on the intent and content of the review. This is a generative model and more useful in our case as most of the reviews contain only 2 or 3 lengthy sentences and extractive way would not be very useful.

4.2. Model Architecture

In our proposed model for generating summaries, we are going to use encoder decoder approach[1].

4.2.1 Encoder Architecture

In Encoder model, we first have an embedding layer that converts the text into embedding. we train our own embeddings during training phase. Next we have a stacked LSTM that captures the accurate representation of our input that is better than a single LSTM. LSTM is a special kind of RNN that can capture long term dependencies. LSTM

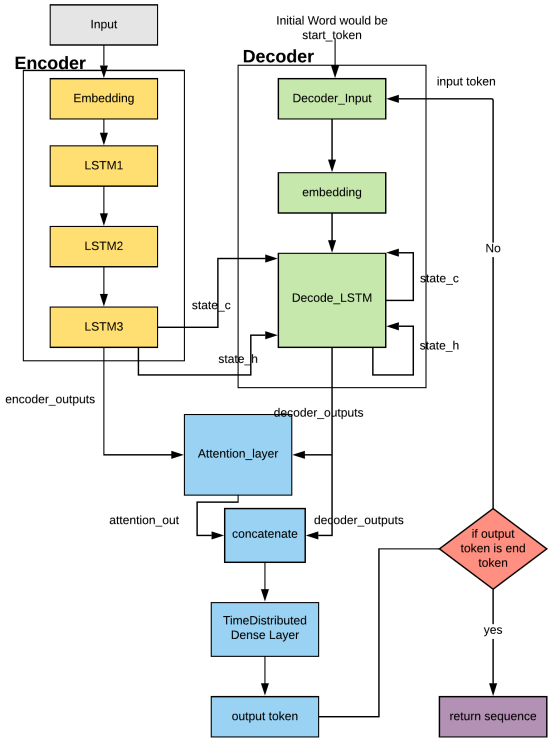


Figure 1: Encoder-Decoder model for Text Summarization

has additional cell state which is more like a conveyor belt that allows easy movement of information from beginning to end of sequence. Controlling this additional information through gates helps in easy management of long term dependencies.The final cell state and hidden state of LSTM in encoder layer is output to LSTM in the decoder layer.

4.2.2 Decoder Architecture

In decoder model, the first layer is embedding layer that converts summary into embedding. Then we use a decoder LSTM that takes state_c, state_h from encoder for the first decoding word and later uses state_c and state_h from previous instances of decoder.

4.2.3 Attention layer and Dense Layer

Before generating the decoded word, we send the output of encoder and decoder to attention layer and then concatenate the attention output and decoder output that is then sent to time distributed dense layer where we apply the softmax function to generate the decoded word. Attention layer helps in capturing the context of given sentence before generating the next word.

4.3. Training Phase

We restrict the length of the summaries to 8 and length of text to 40 words by removing the stop words and truncating the review text above the given length. During Training phase, we add two fixed tokens sostok(start token) and eostok(end token) to summary in order for the decoder to learn summaries accurately. Embedding are also learnt during the training phase.

4.4. Summary Generation Phase

During generation of summaries, we give the entire text to the encoder as input and start with start token and we keep generating decode words until we encounter the end token.

4.4.1 Example Summaries

Some of the summaries that are generated are:
Complete Review
I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more like a stew than a processed meat and it smells better. My Labrador is finicky and she appreciates this product better than most.
Summary
good quality.
Complete Review
Even with small containers, they don't fill them up. These little tins are less than half filled and at the price charged it seems a rip-off. Is there some exotic ingredient as costly as gold contained in those tiny squares? Or how about the cereal ploy, they were filled at the factory but settled in transport.Can manufacturers be honest in their dealings?
Summary
not worth the price
Complete Review
I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more like a stew than a processed meat and it smells better. My Labrador is finicky and she appreciates this product better than most.
Summary
great flavor

5. Rating Prediction/ Sentiment Analysis

We explored multiple machine learning models like Multinomial Naive Bayes (MNB), convolutional neural networks (CNN) and recurrent neural networks (RNN) to predict the ratings of the reviews. For all these models, we predicted rating once by using complete review and once

just using the summary. These methods are discussed in the following sections.

5.1. Data Pre-processing

5.1.1 Pre-processing for MNB Classifier

The pre-processing steps for MNB classifier are as follows. Each sentence is split into words and then stop words and the words with document frequency of less than 20 are removed. We tokenize the whole sentence and represent each sentence by a set of tokens (unique numbers) and the count of each token(number of times the token appears in the sentence). Stop words are the words that are highly frequent like the, a, as, that etc. As these words are present in almost all the documents they do not contain any information specific to a class so they can be removed.

5.1.2 Pre-processing for CNN and RNN

We do the following pre-processing steps on the input text review. We first tokenize the sentences into words, as we work with words instead of the whole sentences. We build vocabulary of the whole text corpus of the food reviews from tokenized words. This vocabulary is used to generate embeddings for each word using the pre-trained word2vec model. Padding is done on the input sentences such that each sentence will have same length equal to the maximum length of a sentence in data. For RNN, we clip the number of words in each padded sentence to 200, and use the clipped sentences as input to the RNN model in 5.4. Whereas for CNN model in 5.3, we use the whole padded data as input.

5.2. Multinomial Naive Bayes(MNB) Classifier

-Venkata Naidu Marineni

Rating Prediction can be considered as a multi class classification task, as for each review we need to assign a rating from the five possible ratings. We consider the five possible ratings as five classes. Each review is considered as a document and the task is to determine the class to which the document belongs to by using the words in the document. The MNB classifier gets it's name from Bayes rule in probability shown in Eq. 1. It is one of the ways to represent the conditional probability *P(x/y)*

$$P(x/y) = \frac{P(y/x)P(x)}{P(y)} \tag{1}$$

The Bayes rule can be applied to document classification as follows. Given a document *d* we find the conditional probability *P(c/d)* for each class *c belongs to C*. The estimated class $\hat{c}$ for document *d* is the class with highest conditional probability which is given by Eq. 2 .

$$\hat{c} = \underset{c \in \mathcal{C}}{\operatorname{argmax}} P(c/d) \tag{2}$$

By combining Eq. 2 and Eq. 1 we get Eq. 3

$$\hat{c} = \underset{c \in \mathcal{C}}{\operatorname{argmax}} \frac{P(d/c)P(c)}{P(d)} \tag{3}$$

The denominator *P(d)* can be dropped from Eq. 3 because it's value remains the same for all the classes. After this we get 4

$$\hat{c} = \underset{c \in \mathcal{C}}{\operatorname{argmax}} P(d/c)P(c) \tag{4}$$

A document is a collection of words in a particular sequence, so *P(d/c)* can be expressed as Eq. 5. *n* is the number of features(each word is a feature) in the document.

$$P(d/c) = P(w_1, w_2, ...w_n/c) \tag{5}$$

As computing *P(d/c)* is very complex Naive Bayes Classifiers make a simple assumption that the probabilities $P(f_i/c)$ are independent given the class c. The value *P(c)* can be computed from the distribution of the classes. Thus the final equation for the class chosen by Naive Bayes Classifier is

$$\hat{c} = \underset{c \in \mathcal{C}}{\operatorname{argmax}} P(c) \prod_{i=1}^{n} P(w_i/c) \tag{6}$$

5.2.1 Training MNB Classifier

The training phase of NB Classifier is estimating the values of $P(w_i/c)$ from the given training data. In our case the training data is the collection of food reviews and their ratings. In the following discussion we consider each food review as a document and their ratings which range from 1 to 5 as the five classes. The maximum likelihood estimate of $P(w_i/c)$ is nothing but the fraction of times the word w_i) appears among all words in all documents of topic *c* should be computed for all the words from the given training data. Let *tf(w_i, c)* denote the frequency of word w_i) in the collection of all the documents of class *c*. Let *V* be the vocabulary of all the classes i.e, the union of all the words from all the documents. The MLE of $P(w_i/c)$ is as follows

$$\hat{P}(w_i/c) = \frac{tf(w_i, c)}{\sum_{w \in \mathcal{V}} tf(w, c)} \tag{7}$$

But there is problem with 7. Suppose that during the training a particular word does not appear in any of the documents of class *c* then the $P(w_i/c)$ word becomes 0. In Naive Bayes classifier, as all the probabilities are multiplied, the probability for this class *c* becomes zero irrespective of the other words in the document. To simplest solution for this problem is to add-one(Laplace Smoothing) to the term frequencies in Eq. 8

$$\hat{P}(w_i/c) = \frac{tf(w_i, c) + 1}{\sum_{w \in \mathcal{V}} (tf(w, c) + 1)} = \frac{tf(w_i, c) + 1}{\sum_{w \in \mathcal{V}} tf(w, c) + V} \tag{8}$$

5.2.2 Tf-Idf

Suppose that a particular word w_1 appears in half of the documents of all the classes and another word w_2 occurs only in half of the documents of the class c. Although w_2 coneys more information of c than w_1, the MLE in Eq. 8 estimates the same probabilities for both words. To overcome this we can use *Tf-idf* value of the word instead of *tf*. The *Tf-idf* value for a word *w* and class *c* is computed as follows

$$Tf - idf(w, c) = tf(w, c).idf(w) \tag{9}$$

$$idf(w) = log(\frac{n_d}{n_d(w)}) \tag{10}$$

where n_d is the total number of reviews and $n_d(w)$ is the number of reviews which have word *w*. By substituting *Tf-idf*(w_i, c) in the place of $tf(w_i, c)$ in Eq. 8 we get 11.

$$\hat{P}(w_i/c) = \frac{Tf - idf(w_i, c) + 1}{\sum_{w \in \mathcal{V}} Tf - idf(w_i, c) + V} \tag{11}$$

5.2.3 Improved MNB Classifier

The input features for the simple MNB Classifier is the bag of words(unigrams) of a document. The performance of the MNB Classifier can be increased if we consider the bigrams(pairs of adjacent words) along with unigrams of the document. This is due to the fact that some pairs of words can covey more information than the words treated as independently. The training process still remains the same except that there are more features(unigrams and bigrams) and the *Tf-idf* values should be computed for all the features.

5.2.4 Testing MNB Classifier

The food reviews data is split into train and test data in 3:1 ratio respectively. The classifier is trained using the tf-idf values of the train data. We trained three different MNB models, the first model was trained with only unigrams of the food reviews, the second model was trained using both unigrams and bigrams of the food reviews, the third model was trained with unigrams and bigrams of the summary produced in section 4. The pre-processing steps of the train data are applied on the test data as well. As expected Improved MNB Classifier performed better than the MNB Classifier.

5.3. Convolutional Neural Network

- Janardhana Swamy Adapa

In this section, we are going to discuss CNN models which are used to predict the rating of text reviews. Since CNN model has been effective for various NLP tasks, we have used it for the current task. We have experimented with two different architectures which are discussed in sections 5.3.1 and 5.3.2.

5.3.1 CNN Model Architecture

The architecture[3] of the CNN model is shown in Figure 2. To generate embeddings for each word in text review, we have used pre-trained word2vec model trained on Google-News corpus. By using this word2vec model, we have generated 300-dimension embedding vectors for each word in the text review, which will be the input to 1-D convolution layers. We have used three 1-D convolution layers with filter sizes 3,4,5 to extract features from word embeddings. Each filter size will consider different window size in the embedding, which results in generating features of different characteristics. We have applied 1-D max pooling with a pool size of 2 on the outputs of convolution layers. This will capture the most important feature for every two words, and reduces the dimension of the convolution layer outputs before flattened and concatenated. As shown in Figure 2, a dense layer of 50 nodes followed by another dense layer of 5 nodes will predict the rating from the features obtained after concatenation. Here, each node in the output layer will predict whether the input text review is the corresponding rating value from 1 to 5.

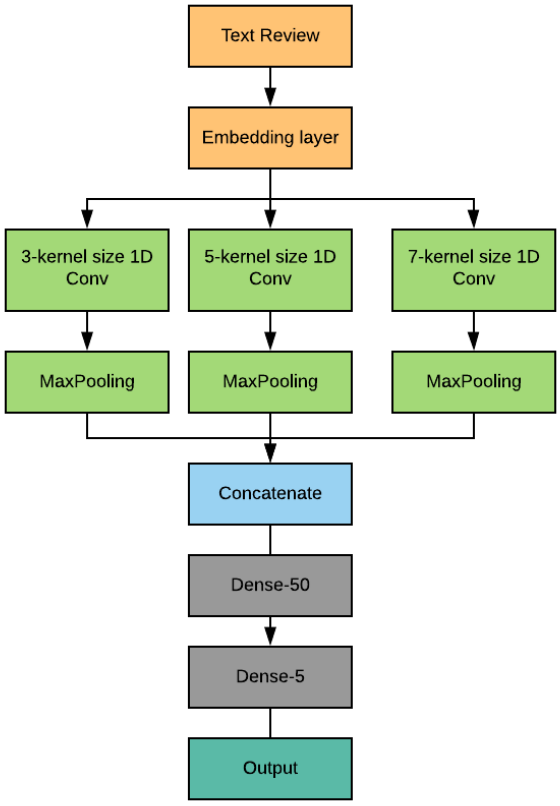


Figure 2: CNN Model Architecture

5.3.2 CNN with Review and Sentiment

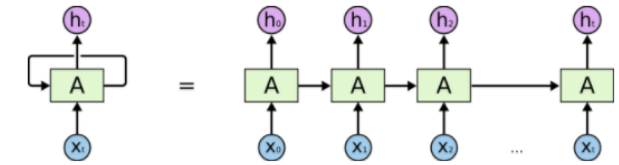
In this section, we are going to explain the modified CNN architecture that was explored to do the rating prediction. The key insight is that using the information whether the review is positive or negative in addition to the text description of the review is beneficial in prediction of the rating from 1 to 5. The reasoning behind this is that as we already know the sentiment of the review whether it is positive or negative, CNN model can use this information and predict the rating value (1-5) more accurately. For this, we implemented two CNN models with similar architecture to Figure 2. The first CNN model, *Model-1* will have text review as input and predicts whether the review is positive or negative. Here, we define positive label as the ratings 4, 5 and negative label as the ratings 1,2,3. *Model-2* takes the text review and output of *Model-1* as input and predict the rating from 1 to 5. The architecture of *Model-2* is similar to the architecture of CNN model shown in Figure 2. Here, the output of the *Model-1* is concatenated to the output of the Dense-50 layer. The concatenated output is used by the final output layer to predict the rating. Since the information of whether a input is positive or negative (*Model-1* output) is a significant component in the prediction of rating, the *Model-1* output is concatenated to the output of the penultimate layer (Dense-50). Another advantage of using architecture, since we have reduced the complexity of task by dividing it into two sub-tasks, we can reduce the size of the model so that the model will get trained quickly as compared to the CNN model in 5.3.1.

5.4. Recurrent Neural Network - LSTM

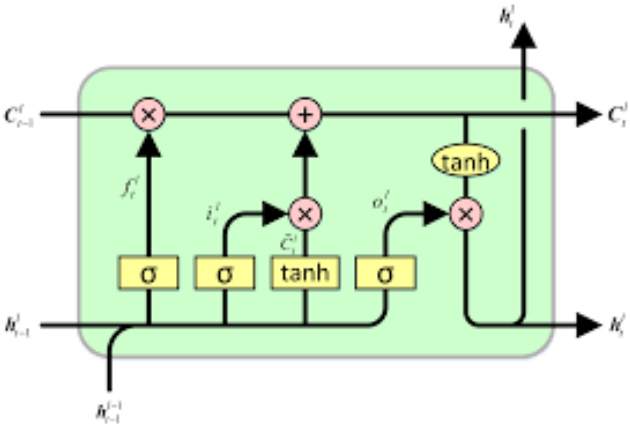
- Prakhar Mohan

Recurrent neural networks (RNN) are a type of artificial neural network where the output of previous step is provided as the input of the current step. RNN can be understood as a small network trained in a loop for every step of a sequence. Figure 3a gives an overview of the an RNN unit. The motivation behind using RNN is that since the words of a used sequentially, the model is better able to capture the context of the sequencial arrangement of words. As the traditional RNN suffers from vanishing gradient problems in case of long sentences, for the purpose of this project we have used Long and Short Term Memory (LSTM) units. LSTM preserves the errors that can be back-propagated through time and layers allowing recurrent nets to learn over a large number of steps. Figure 3b should the details of an LSTM layer where

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$



(a) Unfolded RNN model



(b) An LSTM unit

Figure 3: An overview of RNN and LSTM

$$\bar{C}_t = tanh(W_C[h_{t-1}, x_t] + b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \bar{C}_t$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * tanh(C_t)$$

The RNN architecture used in this project is motivated by the model explained in section 6.2 of Deep Learning with Python by François Chollet[8]. For each sample sentence, the LSTM unit takes the embedding vector of each word one by one as input and gives an output of length same as the size of a embedding vector. This output is then fed to a softmax layer of five nodes which represents rating from 1 to 5. The predicted rating is given by the output node with the maximum value. An overview of the used model is provided in figure 4.

6. Evaluation

Evaluation metrics that we have used are Accuracy and Mean Absolute Error(MAE).
Accuracy and MAE is defined by equations 12 and 13 where N is the total number of reviews in the test set, R_i^p and R_i^a are the predicted and actual rating of i^{th} review.

$$Accuracy = \frac{Number\_of\_correct\_predictions}{Number\_of\_reviews\_predicted} \quad (12)$$

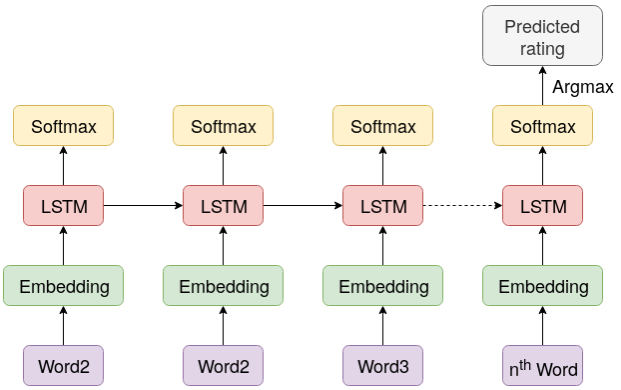


Figure 4: RNN-LSTM model overview

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|R_i^p - R_i^a| \tag{13}$$

MAE gives a better measure of correctness of the model than accuracy. For instance, let us take consider two reviews whose actual rating is 5. If the predicted rating of the reviews are 1 and 4. Accuracy counts both of these as incorrect prediction although the model did a better job at predicted one of the review as 4(close to 5). Where as the MAE metric takes the absolute difference so it differentiates the two cases.

7. Results and Discussions

Table 2 presents the accuracy and MAE values for the three models (Naive Bayes, CNN and RNN) when trained on full text reviews as well as generated text summaries separately. We compared the results of the three models and found that CNN and RNN gave similar accuracy of 81% on the full text reviews and 68.3% on the generated summaries where as the Naive Bayes gave a significantly lower accuracy of 69.2%. This can be attributed to the fact that Naive Bayes is a bag of words approach and is not good at capturing the sequential arrangement of words. Therefore differentiating sentences like "good not bad" and "bad not good" are difficult through Naive Bayes. On the other hand, CNN and RNN are trained on word embeddings which are generating by an extensive training on a huge corpora of words and are able to capture the arrangement of the words in a sentence.

We also found that the prediction accuracy obtained from full text reviews is much higher than the accuracy obtained from generated summaries. As explained above, the generated summaries are much smaller in length and therefore lose a huge amount of information compared to their corresponding full text review. Therefore, from review summaries, it is difficult to identify the extremity of the sentiment. In other words, as the summary generated for a review of rating 5 is similar to the summary generated for a review with rating 4, predicting the accurate rating is more challenging.

	Complete review		Generated Summary	
	Accuracy	MAE	Accuracy	MAE
Naive Bayes	69.2%	0.493	60.0%	0.720
CNN	**80.9%**	0.277	68.3%	0.599
CNN*	76.0%	0.361	-	-
RNN	**81.0%**	0.257	68.2 %	0.598

Table 2: Results for all the explored methods on complete text reviews and the generated summaries. (CNN*: CNN with review sentiment)

Even though the CNN and RNN provides comparable results, we recommend RNN for the task of sentiment analysis and rating prediction. This is because RNN is computationally much less expensive than its equivalent CNN model. Our RNN model has 722,705 trainable parameters whereas CNN model has 14,297,955 which is much higher than the RNN model. ALso, the training duration per epoch for CNN was found to be around 5 times higher than the training of RNN model.

8. Conclusions & Future Work

CNN and RNN on complete text reviews are the preferred models as they can capture the entire context of the review. The performance of these models on summary is reduced due to the reason that they just contain information that will tell us about the sentiment of the review not its extremity. In future, we can explore other models like Gated Recurrent Units (GRU) to predict rating and use other methods like TextRank for text summarization.

References

[1] https://www.analyticsvidhya.com/blog/2019/06/comprehensive-guide-text-summarization-using-deep-learning-python

[2] https://web.stanford.edu/ jurafsky/slp3/4.pdf

[3] Yoon Kim, "Convolutional Neural Networks for Sentence Classification"

[4] https://github.com/junwang4/CNN-sentence-classification-keras-2018

[5] Andrew McCallum, et al., "A Comparison of Event Models for Naive Bayes Text Classification"

[6] M. IKONOMAKIS, et al., "Text Classification Using Machine Learning Techniques", WSEAS TRANSACTIONS on COMPUTERS, 2005

[7] https://www.kaggle.com/snap/amazon-fine-food-reviews

[8] Peng Zhou, et al., "Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling"

[9] François Chollet, "Deep learning with Python", Manning Publications, 2017

[10] Rada Mihalcea, "Graph-based Ranking Algorithms for Sentence Extraction, Applied to Text Summarization", Proceedings of the ACL Interactive Poster and Demonstration Sessions, 2004

[11] Ramesh Nallapati, "Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond", The SIGNLL Conference on Computational Natural Language Learning (CoNLL), 2016