VoC-DL: Revisiting Voice of Customer Using Deep Learning

Susheel Suresh
Adobe Systems, Bangalore
susuresh@adobe.com

Guru Rajan T S
Adobe Systems, Bangalore
gts@adobe.com

Vipin Gopinath
Adobe Systems, Bangalore
vgopinat@adobe.com

Abstract

In the field of digital marketing, understanding the voice of the customer is paramount. Mining textual content written by visitors on websites or social media can offer new dimensions to marketers and CX executives. Traditional tasks in NLP like sentiment analysis, topic modeling etc. can solve only certain specific problems but don't provide a generic solution to identifying/understanding the intention behind a text. In this paper we consider higher dimensional extensions to the sentiment concept by incorporating labels like product enquiry, buying intent, seeking help, feedback and pricing query which give us a deeper understanding of the text. We show how our model performs in a real-world enterprise use case. Word2Vec embeddings are used for word representations and later we compare three algorithms for classification. SVM's provide us with a strong baseline. Two deep learning models viz. vanilla CNN and RNN's with LSTM are compared. With no use of hard-coded or hand engineered features, our generic model can be used in a variety of use cases where text mining is involved with ease.

Introduction

Voice of Customer is a process used to capture the requirements/feedback from the customer (internal or external) to provide the customers with the best in class experience. This process is all about being proactive and constantly innovative to capture the changing requirements of the customers with time.

With the millennials change in mentality towards business's marketing efforts compared to their forebears, companies have to make the best efforts to sell them an experience rather than just products/services. Reciprocal relationships appeal the most to new-age customers rather than marketed products. [1]

By having robust VoC programs in place, businesses can greatly benefit from the insight it provides. They can monitor customer expectations over time, get feedback on new products and services and departments viz. marketing, customer care and operations can make informed decisions.

There are a number of customer management platforms which serve enterprises with solutions that offer VoC capabilities like feedback collection, integration of feedback with other data in a centralized data hub, analysis, reporting, and closed-loop action management. Yet according to a 2017 Forrester [2] evaluation only 42% of the top 1000 companies have a VoC Program in place. Only 19% of them have some kind of automated analysis of customer's voice. We believe the reason for this is because advanced text analytics is not integrated into current VoC pipelines.

Gartner [3] estimates that structured data constitutes only 20% of the complete enterprise data. A major portion of the enterprise information is unstructured data, which among others include communications from the customers. Industry-leading CX (Customer Experience) experts say 87% of the time they find high value in unstructured and text data [4].

With the rising volumes of textual customer voice content, having an automated analytics framework that can understand the true meaning or intent of the text is necessary. We need a system that goes beyond sentiment analysis to understand the intentions of the customer who wrote that text.

Our work is in line with the above thought. We build a supervised multi-label text classification system that can handle a wider range of expressions and signals like product enquiry, buying intent, feedback, seeking help etc. Our belief is that when this model is integrated within VoC pipelines, it will become effective in channelizing text properly thereby increasing business efficiency.

The rest of this paper is organized as follows - We review related work and the current state of the art text classification architectures. Then we introduce our case study. We go on to explain our data collection and annotation process, the proposed architecture and later provide results of our experiments. Finally, we conclude with some discussions and state possible future work.

Related Work

There have been few works in the field of intention analysis of free text (Goldberg et al. 2009; Ramanand, Bhavsar, and Pedanekar 2010). Both use linguistic rules to classify but they cant be extended easily. Sentiment analysis which is also a similar problem, focuses on identifying positive and

[1] Source Gartner: https://www.gartner.com/doc/3069817

[2] https://go.forrester.com/

[3] http://www.gartner.com

[4] Forresters Q4 2016 Customer Experience Programs Survey

negative sentiments in free text (Pang, Lee, and others 2008) and machine learning techniques are used, it is important to note that our work is not a sub-task of sentiment analysis but supplements the area. Our classification task aims to detect the intention of a customer using the comments written in free text, thereby extending the understanding of the text rather than just positives/negatives.

Another work which tackles the problem of deep understanding of a text is (Gupta et al. 2014). They try to find if free text posts on social media have any potential intent to purchase, by classifying them into PI (purchase intent) or non-PI category. They extract generic word, phrase based features and grammatical dependency based features and then use SVM Classifier for detection. The problem with this approach is that creating these features is hard and quite complex.

(Korpusik et al. 2016) also performs a similar task of finding purchase intent in twitter posts. They use word2vec for word embeddings and employ RNN with LSTM for the classification task. Our approach goes further compared to previous work as we provide a deeper holistic understanding of the intentions behind the written text, rather than sentiments or purchase intents.

Representation of words as vectors i.e. Vector Space Models for performing natural language processing (NLP) tasks have a long and rich history. Word2Vec (Mikolov et al. 2013) is a computationally-efficient predictive model for learning word embeddings from raw text. These vectors encode semantic features of words in their dimensions. Semantically close words like e.g. male-female, verb tense, country-capital and even product-company are likewise close (in Euclidean or Cosine distance) in lower dimensional vector space. This has been successfully used in many canonical NLP tasks like named entity recognition, speech tagging etc.

Deep Learning models have achieved remarkable results in solving computer vision (Krizhevsky, Sutskever, and Hinton 2012)and speech recognition (Graves, Mohamed, and Hinton 2013) problems in recent years. Convolutional Neural Networks (CNNs) (LeCun et al. 1998) invented for computer vision tasks has been used a lot for NLP tasks. For text classification, (Kim 2014) trains a simple CNN with embedding layer followed by one convolution layer. They show that despite little tuning, it performs significantly well or equally compared to traditional approaches. We make use of the CNN architecture described in the above paper and compare its performance with RNN's.

Case Study

Overview

A true test for the proposed system will be real-world enterprise textual customer data. Most enterprises have varied data sources for customer's voice. They may be from third party review websites/apps [5], social media [6], in-house help forums etc. The kind of data all these channels generate is similar in nature. All of them are written by customers and they voice their concerns and intents.

[5]TripAdvisor, Yelp, IMDB

[6]Twitter, Facebook

Example Text Data	Correct Label
Looking for information on Adobe Analytics and Marketing Cloud. We need a complete end-end solution for our marketing needs.	Product Enquiry
I am a photographer looking to purchase 10 licences of Adobe Creative Cloud for my studio. Please call me asap.	Buying Intent
Organize and do interesting things with your images quickly, Photoshop express is a great app !	Feedback
How to perform triptych composition using Adobe Lightroom ?	Seeking Help
We are an NGO based out of Sao Paulo. Wanted to know if there is special non profit pricing on Adobe Creative Cloud licences.	Pricing Query

Table 1: Example text and its correct annotation.

In our case study, we train our algorithm with data from these channels. One important point to consider while developing the classes for our classification task is that it should be fairly industry generic.

Below are our generic classes with definitions:

- Product Enquiry (*PE*) : Expression signifying an act of asking for product/service related information.
- Buying Intent (*BI*) : Expression signifying an intention to purchase or consume a product/service.
- Feedback (*F*) : Expression signifying some reaction to a product/service.
- Seeking Help (*SH*) : Expression signifying an act of seeking help related to a product/service.
- Pricing Query (*PQ*) : Expression signifying a question directed specifically towards pricing of a product/service.

Dataset

As there are no annotated corpora available publicly for detection of varying levels of intent, we created our own. Modern-day CRM systems perform an excellent job of data ingestion from varied sources and just with a simple query, all the relevant data can be gathered. We collected 15 thousand data points from CRM systems of a Fortune 500 computer software enterprise. In our case study the data came from Web RFI Forms and social media [7].

To generate the ground truth, we crowdsourced the annotation task. We provided the annotators with aforesaid definitions and a number of examples of labeled text to help them understand the task. The annotators were asked to read the text and annotate them with the most suitable label. To ensure a reliable classification, we had each text annotated by 3 different annotators and we used the majority voting

[7]The company in the case study has dedicated social media handles. All the data is readily available in CRM systems

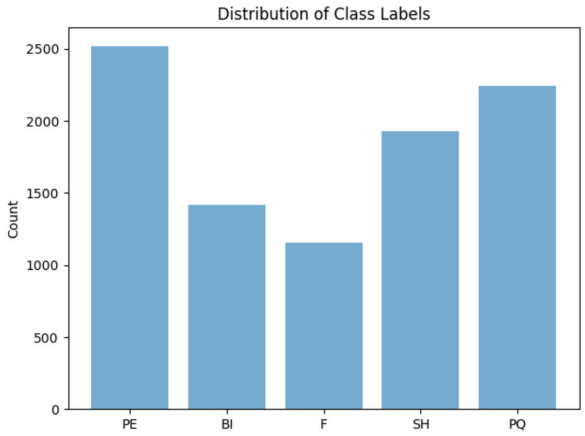

Figure 1: Distribution of class labels in the dataset.

scheme to come up with final labels for the text. Table 1 shows some example labeled data points.

The mean Cohens kappa coefficient of inter-annotator agreement between the sets of annotations was 0.701, indicating a good agreement. This score shows that the class definitions gave the annotator a reasonably good delineated view of the labels. It also shows that the annotation is fairly trustworthy.

After performing certain pre-processing steps viz. removal of texts containing languages other than English [8], replacing special characters with white spaces, normalizing case and removing duplicates we obtained 9,268 data points from the original 15,000. Figure 1 shows the distribution of annotated labels in our dataset.

Embedding Layer

Within NLP, much of the work has involved learning word vector representations through neural language models (Bengio et al. 2003) (Mikolov et al. 2013).

Word Vectors (words projected onto a lower dimensional vector space) essentially feature extractors that encode semantic features of words in their dimensions. In these dense representations, semantically close words are likewise close in Euclidean or Cosine distance in the lower dimensional vector space.

For simplistic use cases, W is initialized to have random vectors for each word. It later learns to have meaningful vectors in order to perform some task. Initializing word vectors with those obtained from an unsupervised neural language model is a popular method to improve performance in the absence of a large supervised training set. We use the publicly available word2vec [9] vectors that were trained on 100 billion words from Google News. The vectors have dimensionality of 300 and were trained using the continuous bag-of-words architecture (Mikolov et al. 2013). Words not present in the set of pre-trained words are initialized randomly.

Classification Methods

Here we introduce the various models that were considered in our case study. We start with a baseline model and then compare two state of the art deep learning paradigms.

Baseline Model Support Vector Machines (SVM's) are predominantly used for classification tasks and they are known to provide strong baselines (Wang and Manning 2012). The goal here is to construct a hyperplane or set of hyperplanes in a high dimensional space, which can be used for classification. For decades, hard-coded/engineered features were given to SVM to perform text classification. As we don't follow the procedure of feature extraction manually from the input text, we obtain vectors for each word in a text from word2vec and consider as the input to the SVM's. We follow the approach taken by (Lilleberg, Zhu, and Zhang 2015). For our experiments, we use a SVM classifier provided by scikit-learn [10] which is based on LIBSVM (Chang and Lin 2011) toolkit with RBF (non-linear) kernel.

Deep Learning Models A basic logistic regression model combines the input with a weight matrix and bias vector and feeds it through a softmax classification layer that yields probabilities for each class i. The class i with the highest probability is the output. An artificial neural network (ANN) network enables more complex functions to be computed through the addition of hidden layers below the softmax.

- An extension of the ANN network for sequences is a **recurrent neural network (RNN)**, in which the hidden layer from the previous timestep is fed into the current timesteps hidden layer:

$$h_t = (W_x x_t + W_h h_{t-1} + b)$$

where h_t is the hidden state, x_t is the input vector, W is a learned weight matrix and b is a learned bias vector.

Thus, information from earlier words are preserved across time and is still accessible upon reaching the final word for making a prediction. However in RNN's, error gradient vanish as sequence becomes long, and the long-term dependencies are no longer stored. To compensate for this, **LSTM** (long short-term memory)(Hochreiter and Schmidhuber 1997) uses input, output, and forget gates to control what information is stored or forgotten within a memory cell over longer periods of time. This happens to solve the problem that RNN's have. Figure 2 shows the RNN + LSTM architecture we have used.

- We compare the above RNN + LSTM model with another deep learning paradigm called **convolutional neural network (CNN)**. The most used and recent CNN architecture for text classification was developed by (Kim 2014) and we use the same. The results presented in the original

[8] Google translate plug-in has a feature which can detect languages of texts. This tool was used.

[9] https://code.google.com/p/word2vec/

[10] http://scikit-learn.org

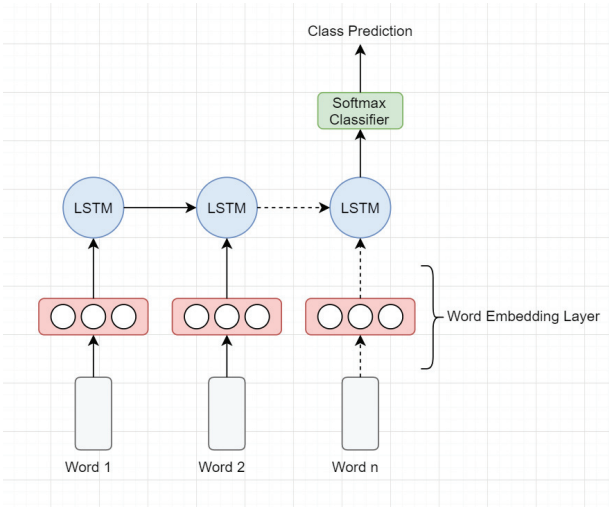

Figure 2: An illustration of RNN + LSTM Architecture.

paper were compelling enough to compare how CNN's perform w.r.t RNN's.

Mathematically, **Input Layer Sequence** x contains n entries. Each entry is represented by a d-dimensional dense vector, thus the input x is represented as a feature map of dimensionality $d \times n$.

Convolution Layer is used for representation learning from sliding w-grams. For an input sequence with n entries: $x_1, x_2, ..., x_n$, let vector $c_i \in R^{wd}$ be the concatenated embeddings of w entries $x_{i-w+1}, ..., x_i$ where w is the filter width and $0 < i < s + w$. Embeddings for $x_i, i < 1$ or $i > n$, are zero padded. We then generate the representation $p_i \in R^d$ for the w-gram $x_{i-w+1}, ..., x_i$ using the convolution weights $W \in R^{d \times wd}$:

$$p_i = tanh(W \cdot c_i + b)$$

where bias $b \in R^d$.

Maxpooling All w-gram representations $p_i(i = 1s + w1)$ are used to generate the representation of input sequence x by maxpooling:

$$x_j = max(p_{1,j}, p_{2,j},)(j = 1, , d)$$

Our architecture is shown in Figure 3. The figure shows only one feature being extracted using one filter, in our implementation we use multiple filters of varying window sizes to obtain multiple features. These features form the penultimate layer and are passed to a fully connected softmax layer whose output is the probability distribution over labels.

Experiments and Results

After the pre-processing steps (explained in the Dataset section) we had 9,268 texts in our dataset. An 80:20 training - test split gave us 7,414 data points for training and the remaining 1,854 data points were set aside as test data. We

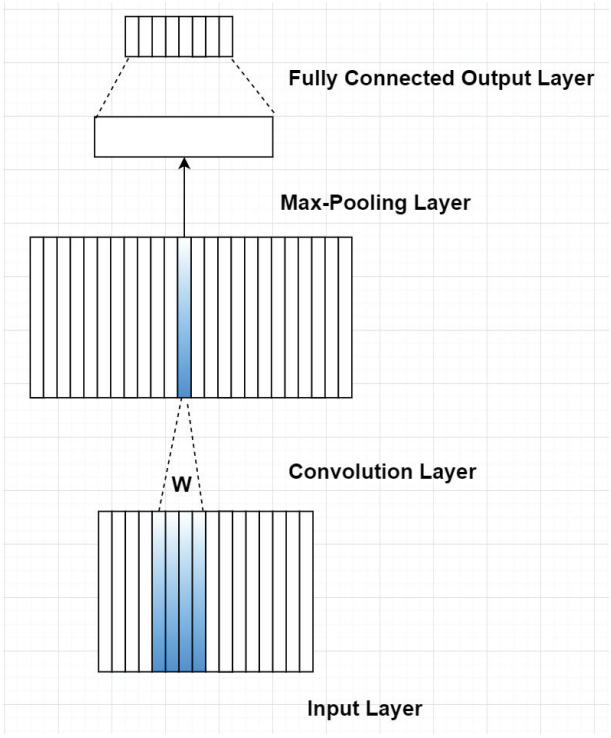

Figure 3: An illustration of CNN Architecture.

utilized pre-trained word2vec embeddings for all four experiments. Each word is represented as a 300-dimensional vector.

We conducted four sets of experiments: first using SVM model which serves as a baseline; second, employing RNN's; third with uni-directional LSTM; fourth using single channel CNN's.

- The first model we tried was the SVM classifier with non-linear (rbf) kernel. Multi-class classification was performed using a one-vs-one scheme. 10 fold cross validation was used to obtain the accuracy score. As the class labels were not imbalanced, a simple accuracy function:

$$accuracy(y, y') = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1(y'_i = y_i)$$

  where y'_i is the predicted value of the i^{th} sample and y_i is it's true value. $1(x)$ is the indicator function. If the predicted label for a sample strictly matches with the true label, then the subset accuracy is 1.0; otherwise it is 0.0.

  We obtained an accuracy of 61.7 % by using the hyper parameters obtained from grid searching for SVM classifier. This forms as our baseline accuracy for the task at hand.

- Next a RNN was employed to perform classification. We implemented the basic RNN using PyTorch [11] library. We initialized the RNN with an input dimension equal to our word vector i.e. 300. The RNN model was run for a total

[11] http://pytorch.org/

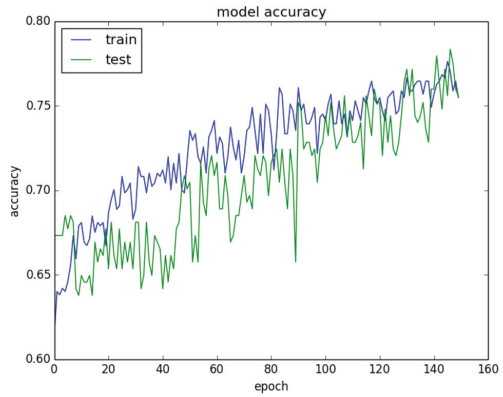

Figure 4: Model Accuracy for RNN

Model	Accuracy %
SVM	61.7
RNN	75.1
CNN	72.2
RNN +LSTM	**82.3**

Table 2: Accuracy of classification models

of 150 epochs. Cross entropy loss was used as the loss function and the Adam algorithm with a learning rate of 0.01 was used for performing gradient descent.

Figure 4 shows the accuracy of both train and test data points. The final accuracy of 75.1 % was obtained.

- For the CNN model, input words represented in 300 dimension word2vec vectors are given as input and 128 filters of sizes 3,4,5 each are applied on the embeddings. One time max pooling is performed for all three convs and 128 feature maps are generated for each region size. For our experiments, we used TensorFlow (Abadi et al. 2015) library for developing the CNN architecture. A final accuracy of 72.2 % was obtained.
- RNN's with LSTM were implemented in PyTorch. All the hyper parameters were similar to the RNN, the state size was set to 10 (additional hyper-parameter in LSTM's). Figure 5 shows the training and test accuracy. The accuracy boosted to 82.3 % in this case. It is clear from the figure that the accuracy for test plateaus after a few hundred batch iterations. This shows that the model has converged.

The results are tabulated in Table 2. 10-fold cross-validation was used to obtain all of the performance results.

The CNN model performs similarly to the basic RNN model. The time it took for training the CNN is small compared to RNN. RNN"s are much harder to train in general. The reasoning behind this is that CNN can get local context well based on the filter sizes. RNN's, on the other hand, can theoretically store context, but because of the vanishing errors, they don't perform very well in practice. The LSTM cell performs the best among all three models. The ability to hold key contextual information makes it possible to obtain the best accuracy.

Dataset	CNN	RNN	RNN + LSTM
TREC	92.8	93.6	95.4
SST - 1	43.0	42.9	46.3
SST - 2	82.1	83.4	87.4
MR	75.4	77.2	82.7

Table 3: Evaluation on other standard datasets

Evaluation on other standard datasets

To show that the deep learning models used for our task are state of the art, we present results that show how these models perform on some standard text classification datasets in Table 3. Below is a list of standard datasets used:

- TREC: TREC question dataset - task involves classifying a question into 6 question types (whether the question is about person, location, numeric information, etc.). [12]
- SST-1: Stanford Sentiment Treebank - an extension of MR but with train/dev/test splits provided and fine-grained labels (very positive, positive, neutral, negative, very negative). [13]
- SST-2: Same as SST-1 but with neutral reviews removed and binary labels.
- MR: Movie reviews with one sentence per review. Classification involves detecting positive/negative reviews. [14]

Conclusion

We have shown how deep learning models with word2vec embeddings perform in text classification with the help of a real-world use case.The case study can bring about positive changes in the way enterprises handle customer pain points, feedbacks and intents. Companies can get better quality leads, target the right customers and improve customer experience. Superior channeling of text data is possible and better insights can be generated on top of existing sentiment analysis.

With the use of a generic model, we are able to overcome two major drawbacks of hard-coded features: the high cost of making rules by analyzing and their low coverage.

We reiterate the fact that word2vec word embeddings truly represent universal features and CNNs are very much capable of text classification. They give similar results when compared to basic RNN's. Originally CNNs were designed for computer vision problems, but with the advent of superior vector representations CNNs can be used to solve specific NLP problems. The best deep learning model for our task is a RNN+LSTM architecture. With its ability to capture temporal relationships in a text and solve the vanishing gradient problems in vannila RNN's, LSTM's gave us an accuracy increase of 13 % compared to basic CNN's and RNN's.

[12] http://trec.nist.gov/data/qa.html

[13] https://nlp.stanford.edu/sentiment/

[14] https://www.cs.cornell.edu/people/pabo/movie-review-data/

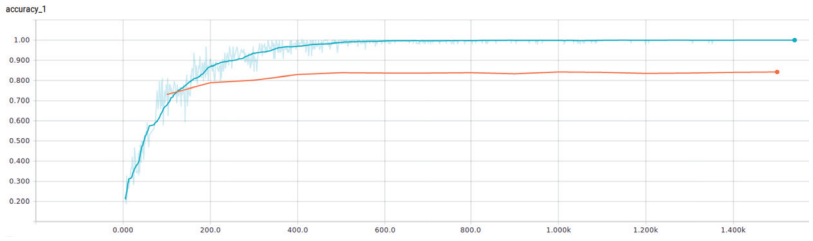

Figure 5: Model Accuracy for RNN + LSTM

Future Work

Different word embeddings can be used to compare accuracy. One could also use Doc2Vec which represents entire sentences in vector formats rather than individual words. A comparative study can be done using both kinds of embeddings.

We are only scratching the surface of the problem of fully understanding the intent of a customer written text. Supervised methods can only go so far in truly understanding the text. We need new unsupervised approaches that can perform text classification to higher level concepts.

Acknowledgements

The authors would like to thank the numerous annotators for labeling the dataset. We would also like to thank Prof. Subramaniam K.V of P.E.S University, Bangalore [15] and Deepak Gopalakrishnan of Adobe Systems [16] for their support and guidance.

References

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; and Ghemawat, S. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Bengio, Y.; Ducharme, R.; Vincent, P.; and Jauvin, C. 2003. A neural probabilistic language model. *Journal of machine learning research* 3(Feb):1137–1155.

Chang, C.-C., and Lin, C.-J. 2011. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)* 2(3):27.

Goldberg, A. B.; Fillmore, N.; Andrzejewski, D.; Xu, Z.; Gibson, B.; and Zhu, X. 2009. May all your wishes come true: A study of wishes and how to recognize them. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 263–271. Association for Computational Linguistics.

Graves, A.; Mohamed, A.-r.; and Hinton, G. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, 6645–6649. IEEE.

Gupta, V.; Varshney, D.; Jhamtani, H.; Kedia, D.; and Karwa, S. 2014. Identifying purchase intent from social posts. In *ICWSM*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Kim, Y. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Korpusik, M.; Sakaki, S.; Chen, F.; and Chen, Y.-Y. 2016. Recurrent neural networks for customer purchase prediction on twitter. In *CBRecSys@ RecSys*, 47–50.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Lilleberg, J.; Zhu, Y.; and Zhang, Y. 2015. Support vector machines and word2vec for text classification with semantic features. In *Cognitive Informatics & Cognitive Computing (ICCI* CC), 2015 IEEE 14th International Conference on*, 136–140. IEEE.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Pang, B.; Lee, L.; et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval* 2(1–2):1–135.

Ramanand, J.; Bhavsar, K.; and Pedanekar, N. 2010. Wishful thinking: finding suggestions and'buy'wishes from product reviews. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, 54–61. Association for Computational Linguistics.

Wang, S., and Manning, C. D. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, 90–94. Association for Computational Linguistics.

[15] subramaniamkv@pes.edu
[16] dgopalak@adobe.com