NTUA-ISLab at SemEval-2019 Task 9: Mining Suggestions in the wild

Rolandos Alexandros Potamias, Alexandros Neofytou, Georgios Siolas
Intelligent Systems Laboratory,
National Technical University of Athens,
Zografou, Athens , Greece
rolpotamias@gmail.com, alex.neofytou@gmail.com
gsiolas@islab.ntua.gr

Abstract

As online customer forums and product comparison sites increase their societal influence, users are actively expressing their opinions and posting their recommendations on their fellow customers online. However, systems capable of recognizing suggestions still lack in stability. Suggestion Mining, a novel and challenging field of Natural Language Processing, is increasingly gaining attention, aiming to track user advice on online forums. In this paper, a carefully designed methodology to identify customer-to-company and customer-to-customer suggestions is presented. The methodology implements a rule-based classifier using heuristic, lexical and syntactic patterns. The approach ranked at 5^{th} and 1^{st} position, achieving an f1-score of 0.749 and 0.858 for SemEval-2019/Suggestion Mining sub-tasks A and B, respectively. In addition, we were able to improve performance results by combining the rule-based classifier with a recurrent convolutional neural network, that exhibits an f1-score of 0.79 for subtask A.

1 Introduction

Nowadays, online platforms and e-commerce sites have become increasingly popular. In such online environments people are willingly sharing views, feelings and opinions about specific products and services by conveying written recommendations on several topics. The pool of advising items that float in contemporary e-commerce environments posts a critical task: how to identify, capture and extract useful information from them. Digging into these information pools of unstructured commenting dialogues about ideas and thoughts expressed by reviewers on the product, its features, components or functions can be seen as an Opinion Mining (OM) problem (Banitaan et al., 2010), with the relevant research gaining increasingly significant attention.

However, OM strategies attempt to extract the sentiment of users written passages, failing to correctly capture putative advice, especially in the presence of mixed emotions. For example, a suggestion in an online trip advising platform could contain the sentence *"If you sleep lightly, this would be a problem because of all the trams that go by - so ask for an interior room."*. The passage conveys a negative sentiment which is impossible to grasp correctly. Confronted with this problem, computational methodologies that automatically identify and highlight clear-cut advice in written recommendations raises as a major need. Towards this end, Suggestion Mining (SM) strives to track suggestions and tips in passages. Suggestive sentences could be positive, negative or they may not contain any sentiment at all, a fact that makes their tracking quite puzzling (Negi and Buitelaar, 2015). In general, advice may be expressed by using either *explicit suggestions*, which propose direct and conventionalized forms of recommendations (Martínez Flor, 2005), or *implicit suggestions* where the precise recommendation is in a way veiled and should be inferred (Negi and Buitelaar, 2015).

The work reported in this paper copes mainly with explicitly expressed suggestions, ignoring indirect recommendations, such as *The best thing about the Westin, as everybody has already stated, is the location.* where, via the expressed positive response, the writer implies a recommendation about *Westin* hotel. In particular, we tackle the problem of detecting suggestive comments by carefully devising a number of heuristic features to represent various lexical and imperative patterns that are indicative of explicit advice. The basic contributions of the proposed SM approach are:

- Implementation of a robust and extended identifier of imperative, prompting and solic-

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 1224–1230
Minneapolis, Minnesota, USA, June 6–7, 2019.

itation clauses in sentences that are mostly related with advice and suggestions.

- Extension and elaboration on existing dictionaries and respective lexical patterns that are indicative of advice and suggestions, focusing on reviews related to electronics and hotels.
- Composition of a fully equipped and highly accurate suggestion mining framework that compares and outperforms other related approaches.

2 Related Work

Even if SM is still in its youth, few related studies aim towards the extraction of suggestions. Goldberg et al. (2009) introduced a pattern-based approach to capture the desires of customers regarding company suggestions, utilizing a Support Vector Machines (SVM) classifier. A similar study, conducted by Ramanand et al. (2010), on extracting suggestions and purchase desires for product services is founded on a pattern-based approach and the devise of respective rules. The first study that focused on the extraction of exact suggestions from product reviews and utilized feature-based classification techniques is reported in (Brun and Hagege, 2013). The same line of research is followed in Negi and Buitelaar (2015) where, an SVM approach is employed to classify heuristically devised sequential features. In a follow-up work (Negi et al., 2016) it was demonstrated that the utilization of deep learning techniques, such as Convolutional Neural Network (CNN) and Long-Short-Term-Memory (LSTM) architectures, may improve prediction performance. In a similar setting, in Golchha et al. (2018) a hybrid deep learning framework is introduced, which is composed by both recurrent and convolutional network architectures, coupled with linguistic features. A different approach is proposed in Gottipati et al. (2018) that exploits suggestions on student feedback comments utilizing a decision tree classification approach.

3 Experimental Setup

3.1 Dataset

The SemEval-2019/SM task 9 (Negi et al., 2019), targets the handling of different suggestion forms. It provides a 9K training dataset with sentences related to customer suggestions that were extracted from the "Universal Windows Platform (available from `uservoice.com`) .The provided dataset is imbalanced as only 26% of cases are classified as suggestions. The provided test dataset for subtask A was also imbalanced as it contains only 87 suggestive instances from a total of 833 sentences. Data for subtask B was extracted from TripAdvisor forum[1], and carries different aspects of advice between customers. The provided test set contains 824 sample instances, 348 of which indicate explicit suggestions.

3.2 Preprocessing: *To do or not to do?*

Suggestions often comprise phrases, expressions and lexical patterns to denote their suggestion content. Such patterns may contain several *stopwords* such as *be* or *should*. We claim that retaining stop-words is needed in order to identify special lexical patterns like for example, "be sure" or "would be nice", which are indicative of the suggestion mood and content of sentences. To this end, we keep the prepossessing step as simple as possible by just lowering uppercase letters and deleting repeated punctuation. We also apply `pyspellchecker`[2] to correct any misspelled words.

4 Method

Our approach to the SM task is founded on the careful identification of lexical patterns and the assignment of contextual importance weights to them. A respective rule-based classifier is then formed that computes the degree of a sentence's suggestion content according to the importance of the included patterns. The identified patterns for both subtasks A and B are acquired by utilizing a common dictionary of suggestion patterns and related corpora, as well as imperative featured patterns described in Section 4.1. In addition, the submitted classifier for subtask A was further improved by coupling it with a convolutional recurrent neural network (Section 5).

4.1 Detection of Imperative Forms

As a linguistic property the imperative mood is found to be deeply connected to suggestions in various studies Wicaksono and Myaeng (2012). Thorough observation and analysis of the linguistic characteristics related to both subtasks, led us

[1] https://www.tripadvisor.com.gr/
[2] https://pypi.org/project/pyspellchecker/

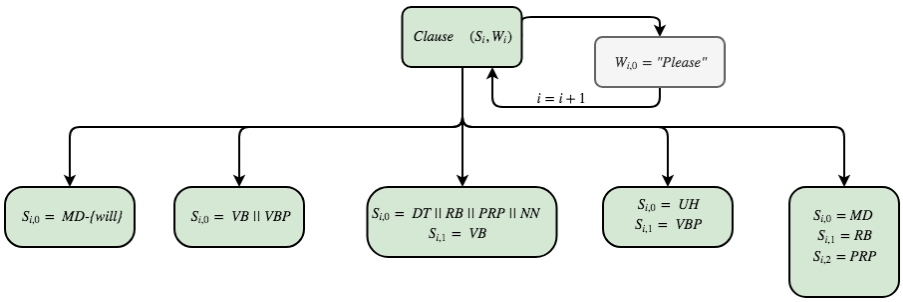

Figure 1: Imperative mood patterns based on Part of Speech Tags. Notation: $S_{i,j}$ denotes POS tag of word j in clause i and $W_{i,j}$ the mentioned word.

Pattern list $\mathbf{P_c}$ - Task A	**Pattern list $\mathbf{P_b}$** - Task B
should [not/be/take/include/start]	[do not]/[if only]
be [better]	[so/before/can/for/if] you
[that way]/[so that]/[why not]	you [will/need/can/may]
[suggestion is]/[good solution]/[the idea]	[make/be sure]/[watch out]
to allow	[go/going/asking/wishing] for
would make	would advise
[will/would] be [`Positive Sentiment`]	[will/would/could] be [`Positive Sentiment`]
[to/would/could] enable [i/would/id] [like/prefer]	be [prepared/careful/warned/forewarned] [i/would/i'd] [like/prefer]
am asking for	highly recommended
look into	[look/looking] [into/for/up/around]
make it	why not
at least	is there
we need	we need

Table 1: Sequential patterns indicating suggestive mood; left column: patterns related to subtask A; right column: patterns related to subtask B; "/" denotes logical *or*.

to extend the idea of an imperative mood detector for a task-independent and rule-based detection algorithm, able to identify the presence of prompting or solicitation clauses. The ensemble of these key features is accomplished by a mixture of rules that check both word and Part-of-Speech (POS) tag combinations, as POS tags usually avoid the sparse nature of word-based patterns. Our algorithmic approach unfolds in three steps:

- Each sentence undergoes basic preprocessing, including the replacement of each word with a tuple that contains the word and its POS tag, and enable the formation of mixed rules. In order to minimise POS errors that could lead to misclassification due to incorrect pattern recognition, a combination of the TextBlob library API[3] and the tagged Brown Corpus were utilized to assure correct word tagging.

- Each sentence is split into separate clauses that may independently indicate imperative mood, depending on their first word or verb phrase. Sentences are split at certain punctuation marks, special characters and the word 'please', as the latter provides almost absolute evidence of a following verb phrase.

- Each clause is then checked for the occurrence of certain POS tag patterns (see Figure 1), which are strong indicators of imperative mood.

4.2 Subtask A

The devised rule-based classifier assigns confidence scores to sentences on the basis of lexical-patterns organised in pre-specified categories and respective lists (detailed below). For subtask A we developed two lexical lists and one pattern-based list that vary according to their semantic content. The confidence score for each sentence is computed by a weighed additive formula that sums the hit-rates of all engaged features. A sentence is considered as suggestion if it exceeds a score of 0.15. The specific threshold is fixed after extensive experimentation on the given input dataset. Furthermore, we applied a 0.2 penalty for short phrase sentences that contain less than five words, as they

[3] https://textblob.readthedocs.io/

are considered to lack explicit suggestion content. In Table 2 we present performance results for each of these lists, as well as the prediction performance figures achieved by their combination (submitted results).

List $\mathbf{P_a^A}$. The first high rated list is composed by word features that are indicative and capture writers suggestive mood, e.g., *suggest , should , shouldn't, needs, idea, helpful, consider , allow, disallow*. Such words urge developers and supply companies to improve products and services by tracking words with directive, e.g., (*should, suggest*) or advisory content, e.g., (*idea, consider, helpful*). Sentences containing features from this list are considered as putative suggestions, assigned a 0.3 confidence score

List $\mathbf{P_b^A}$.The second list is composed by features that represent lexical patterns which, even if they do not contain individual words indicative of suggestive expressions, could be utilised when they emerge together in order to better grasp suggestive content and improve classifier's confidence. This list contains modal words, *would, could*, as well as their negations, *wouldn't, couldn't*. In addition, the list is enhanced with two developer related verbs, *create, include*. A sentence that contains both *could* and *create*, and which is rationally considered as suggestion, is: *"Read/write access to email sync settings could enable applications to create custom sync profiles (based on week-days time position etc)."*. Each hit from this list assigns a 0.1 weight to the sentence confidence score.

Patterns $\mathbf{P_c^A}$. Along with single word features we enhance our classifier with pattern-based features, elaborating on an earlier related work (Goldberg et al., 2009). To this end, we utilized several sequential lexical features as shown in Table 1. In addition, claiming that phrases like *will/would be* followed by words of positive polarity, e.g., *helpful, very very nice* are most of the times indicative of suggestions, we counted the total polarity of the three words following the phrase *will/would be*. When such patterns occur, the sentence's confidence score is increased by 0.25.

4.3 Subtask B

Following the same methodology as in subtask A we acquire a list of lexical features along with a list of weighted sequential lexical patterns. To develop lexical and n-gram patterns we extracted the most frequent patterns from Wachsmuth et al. (2014), keeping those with the most advising content. For this subtask we consider a sentence as suggestive if it exceeds a 0.1 confidence score.

Feature Set	Dev			Test		
	Prec	Rec	F1	Prec	Rec	F1
$P_a^A + P_b^A$	0.86	0.4	0.55	0.67	0.64	0.66
P_c^A	**0.87**	0.35	0.5	**0.7**	0.41	0.53
Imperative	0.83	0.23	0.36	0.67	0.21	0.32
Submitted-All	0.81	**0.71**	**0.76**	0.66	**0.86**	**0.74**

Table 2: Comparison between different lexical patterns on the development and test sets for subtask A.

Feature Set	Dev			Test		
	Prec	Rec	F1	Prec	Rec	F1
P_a^B	0.94	0.47	0.62	0.93	0.49	0.65
P_b^B	**0.98**	0.52	0.68	0.89	0.49	0.64
Imperative	0.97	0.38	0.54	**0.93**	0.36	0.52
Submitted-All	0.94	**0.78**	**0.86**	0.90	**0.82**	**0.86**

Table 3: Comparison between different lexical patterns on the development and test sets for subtask B.

List $\mathbf{P_a^B}$. In contrast with subtask A, where suggestions appear between customers and companies, in subtask B most of the suggestions refer just to customers. Thus, we acquired a number of lexical features indicating a warning, an advice or a desire in order to capture suggestions in travel forums.

Caution words: *avoid, beware,don't, expect, remember*

Advice words: *tip, advise, advice, recommended, recommendation, suggest, suggestion, ask, bring, pick, consider, spend, expect, can*

Wish words: *please, can, hopefully, enjoying, want, wanting, prefer*

The occurrence of each of the above listed words in a sentence is assigned a weight of 0.25.

List $\mathbf{P_b^B}$. As in subtask A we used several sequential lexical patterns to identify and capture bigrams and tri-grams with suggestive content. All lexical patterns of this list are summarized in the second column of Table 1.

5 Improving Predictions with Recurrent Convolutional Neural Networks - (R-CNN)

As simple and carefully devised lexical patterns are able to capture suggestive content in reviews and customer-to-customer conversations, we attempted to create an even more robust classifier

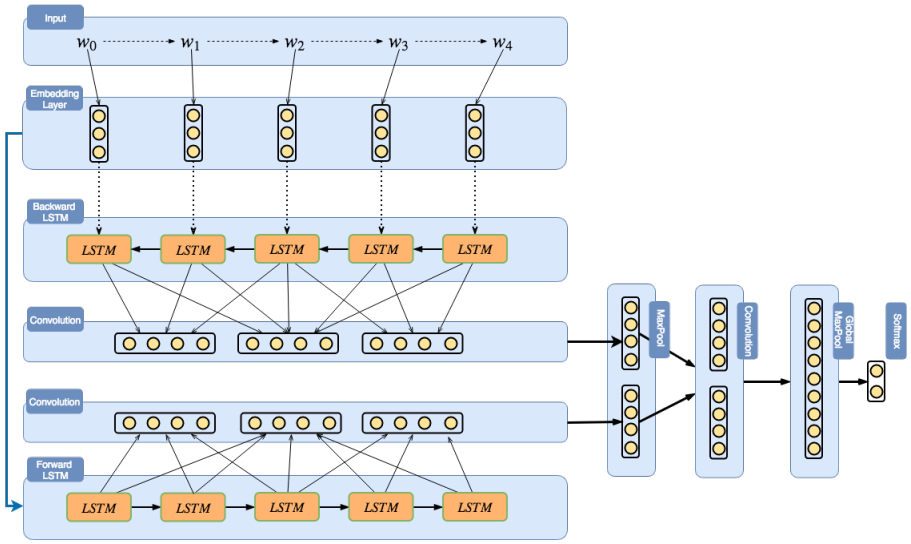

Figure 2: The recurrent convolutional neural network architecture (R-CNN).

on customer-to-companies suggestions. Thus, we coupled and enhanced our subtask A submitted classifier with a deep learning (DL) LSTM-CNN neural architecture.

5.1 Embedding Layer

DL is a very powerful classification approach, with the *Word Embeddings* machinery to be an important and necessary part for their training, especially for recurrent neural network (RNN) architectures (Mikolov et al., 2013). Word embeddings project each word to its semantic and highly dimensioned vector representation, and are induced by exhaustive training of huge corpora. In the present work we utilized 300-dimensional pretrained Standford GloVe embeddings (Pennington et al., 2014).

5.2 Bidirectional LSTM

Due to the sequential nature of textual information neural architectures able to capture time-depended information are needed. This property is offered by RNN, and especially Long-short-term-memory (LSTM) architectures (Hochreiter and Schmidhuber, 1997). Forward LSTMs processes input information $\mathbf{x} = (x_0, x_1, ..., x_n)$ from first word x_0 to the last one x_n educing relative hidden states h_t for each time step t. However, a lot of sequential information may be hidden in long distant dependencies. Thus, to enhance forward LSTMs performance we utilized bidirectional LSTMs (Bi-LSTM) processing input from both directions, forward (from x_0 to x_n) and backward (from x_n to x_0). Thus, for a given time sample t, the hidden state h_t is defined as the concatenation of forward and backward hidden states:

$$h_t = \overrightarrow{h_t} || \overleftarrow{h_t}, \quad h_t \in R^{2*L} \tag{1}$$

where $\overrightarrow{h_t}$, $\overleftarrow{h_t}$ denotes forward and backward hidden states respectively, and $mathitL$ denotes the number of units in each LSTM cell.

5.3 Convolutional Layer

Convolutional layers are the basic component of Convolutional Neural Networks (CNN), containing m convolutional filters aiming to reduce frequency variations. The convolution layer maps input matrix $\mathbf{S} \in R^{l \times D}$ into $c \in R^{|s|+h-1}$ using convolutional filters $\mathbf{F}$ with length h, and calculating each component c_i by:

$$c_i = \sum_{k,j} (\mathbf{S}_{[i:i+h]})_{k,j} \cdot \mathbf{F}_{\mathbf{k,j}} \tag{2}$$

Filters $\mathbf{F}$ slide over the input matrix, performing element wise product between a column slice of input s and filter matrix k, in order to produce each vector component c_i. Finally, vectors c_i aggregate over all filters, producing a feature map matrix $\mathbf{C} \in R^{m \times |s|+h-1}$, which is passed through a non-linear ReLu activation function.

5.4 Pooling Layer

To reduce the spatial size of the convolutional layer output we use a pooling layer. Pooling layers manage to tune and reduce the amount of network parameters preventing overfitting. In the current work, a MaxPooling layer is used over the convolutional layers, performing a maximum element

selection of n non-overlapping intervals. Thus, MaxPooling layer results in $\mathbf{C_{pool}} \in R^{m \times \frac{|s|+h-1}{n}}$.

5.5 The Enhanced Classifier

The devised recurrent convolutional neural network (R-CNN) architecture is shown in Figure 2. Each Bi-LSTM layer, comprised by 164 units each, was fed with 300-dimensions embedding vectors. The output of Bi-LSTM is processed by two 1-d convolutional-pooling layers containing 128 ReLU activated filters of length 5. Each convolutional layer is followed by a max-pooling layer of size 5 to reduce networks parameters. For the final pooling layer we utilize a global pooling that maps all sequential patterns to a feature vector, followed by a *softmax* activated dense network. The classification result coming from the R-CNN network couples the rule-based classification (presented in Section 4.2) following a parallel fail-safe mode, that is: in the event that the softmax output exceeds a certain *confidence threshold limit* for a specific class label (as shown in Table 4), and the rule-based classification induces the opposite label, the finally assigned class label is the one induced by the R-CNN.

5.6 Training

The proposed rule-based and R-CNN coupled model was trained following a combination of train-and-trial one epoch mode. In addition, we applied several regularization techniques to prevent overfitting. In particular, we adopted Dropout (Srivastava et al., 2014; Gal and Ghahramani, 2016), empirically set to 0.3, in order to randomly deactivate 30% of recurrent neuron connections between LSTM units. Finally, to optimize network training performance we adopted Adam optimizer as proposed by Kingma and Ba (2014), using cross-entropy loss function.

6 Results

Our officially submitted results ranked in the 5^{th} and 1^{st} place for subtasks A and B, respectively (results are presented in Tables 2 and 3). Our rule-based approach that expand proposed lexical and pattern-based lexical features tend to perform significantly well for both suggestion domains, customer-to-company and customer-to-customer. As shown in Table 3, the proposed model achieves almost the same results on both development and test datasets, as well as stable precision and recall performances. The combination of lexical, sequential and imperative mood features attains 0.86 *f1-score* and robust performance over all metrics. Similar to subtask B, the submitted results for subtask A exhibit almost the same *f1-scores* on both development and test datasets, 0.76 and 0.74, respectively. However, and in contrast to subtask B, one may observe a slight unstable performance between precision and recall, as shown in Table 2. To this end, and in order to improve precision performance, we introduced the ensemble-like combination of R-CNN and rule-based classifiers (Section 5.5). As illustrated in Table 4, the combined classifier achieves increased precision figures by up to 8%, and manages to outperform related submitted systems, and attain state-of-the-art prediction scores.

Method	Test-Subtask A		
	Prec	Rec	F1
R-CNN	0.64	0.79	0.71
Submitted	0.66	**0.86**	0.74
Proposed-0.75	0.73	0.78	0.77
Proposed-0.80	0.73	0.84	0.78
Proposed-0.85	**0.74**	0.85	**0.79**

Table 4: Comparison between rule-based submitted results, R-CNN and their combination. The proposed method tested on several confidence thershold limits as defined in Section 5.5.

7 Conclusion & Future Work

The presented work, introduces and implements a pattern/rule-based classification methodology on two SM tasks that shows very good performance. In addition, we introduce and propose a combination of the pattern/rule-based classifier with a carefully devised R-CNN classifier that achieves highly accurate and state-of-the-art results. Due to the different hyperparameters and rule weight optimization needed for each dataset, our implementation is currently split into two content-specific classifiers. Therefore, a robust classifier detecting cross-domain suggestions, regardless of the input content, is an important goal to fulfill and we plan to work towards this target.

References

Shadi Banitaan, Saeed Salem, Wei Jin, and Ibrahim Aljarah. 2010. A formal study of classification

techniques on entity discovery and their application to opinion mining. In *Proceedings of the 2nd international workshop on Search and mining user-generated contents*, pages 29–36. ACM.

Caroline Brun and Caroline Hagege. 2013. Suggestion mining: Detecting suggestions for improvement in users' comments. *Research in Computing Science*, 70(79.7179):5379–62.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Hitesh Golchha, Deepak Gupta, Asif Ekbal, and Pushpak Bhattacharyya. 2018. Helping each other: A framework for customer-to-customer suggestion mining using a semi-supervised deep neural network. *arXiv preprint arXiv:1811.00379*.

Andrew B Goldberg, Nathanael Fillmore, David Andrzejewski, Zhiting Xu, Bryan Gibson, and Xiaojin Zhu. 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*, pages 263–271. Association for Computational Linguistics.

Swapna Gottipati, Venky Shankararaman, and Jeff Rongsheng Lin. 2018. Text analytics approach to extract course improvement suggestions from students feedback. *Research and Practice in Technology Enhanced Learning*, 13(1):6.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Alicia Martínez Flor. 2005. A theoretical review of the speech act of suggesting: Towards a taxonomy for its use in flt. *Revista alicantina de estudios ingleses, No. 18 (Nov. 2005); pp. 167-187*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Sapna Negi, Kartik Asooja, Shubham Mehrotra, and Paul Buitelaar. 2016. A study of suggestions in opinionated texts and their automatic detection. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 170–178.

Sapna Negi and Paul Buitelaar. 2015. Towards the extraction of customer-to-customer suggestions from reviews. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2159–2167.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*, volume 14, pages 1532–1543.

Janardhanan Ramanand, Krishna Bhavsar, and Niranjan Pedanekar. 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*, pages 54–61. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, Gregor Engels, and Tsvetomira Palakarska. 2014. A review corpus for argumentation analysis. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 115–127. Springer.

Alfan Farizki Wicaksono and Sung-Hyon Myaeng. 2012. Mining advices from weblogs. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2347–2350. ACM.