Automatic action extraction for short text conversation using unsupervised learning

Senthil Ganesan Yuvaraj[1], Shayan Zamanirad[1], Boualem Benatallah[1], and Carlos Rodriguez[2]

[1] University of New South Wales, Sydney, Australia
{senthily,shayanz,boualem}@cse.unsw.edu.au
[2] Universidad Católica Nuestra Señora de la Asunción
carlos.rodriguez@uc.edu.py

Abstract. Collaboration tools are important for workplace communication. The amount of conversation data produced in workplaces are increasing rapidly, placing a burden on workers. There is a necessity to analyze large amounts of data automatically to extract actionable information. Multiple studies were conducted on action extraction to identify actions such as promises and requests. Most of these studies used supervised learning methods. The key problem discussed in this paper are (i) the automatic extraction of action types from short text conversations in collaboration tools such as Twitter and Slack, and (ii) leveraging large amounts of data using unsupervised learning. Data labelling is an important issue when dealing with large datasets for training and extending the corresponding algorithms across different actions and domains. In this paper, we propose an unsupervised learning approach using a combination of relation extraction techniques and word embedding to leverage large amounts of data. The *action2vec* model is created to identify specific actions from short text of conversation data. We have evaluated our unsupervised method against supervised learning and the results are comparable. The action type extractor is integrated with Slack to provide assistance for action type extraction. Thus, the contributions of this paper include an unsupervised learning method to utilize large amounts of data, an automatic extraction of action types from short text and the integration of our approach with state of the art collaboration tools.

Keywords: Action type extraction · Text embedding · Action recognition · Text mining · Short text conversation

1 Introduction

Collaboration tools (e.g., Slack[3]) have become a primary communication means in the workplace and information workers generate large volumes of conversation data in their day-to-day tasks. Email, blogs, wikis and Twitter have become increasingly available and accepted in workplace communications [23]. More specifically, email and instant messaging (IM) are considered key tools in this context.

[3] https://slack.com/

The final authenticated version is available online at https://doi.org/10.1007/978-3-030-62008-0_8

Email is typically used on a daily basis across organizations and it is an important tool for non face-to-face communication [8]. Instant messaging (e.g., IBM Lotus Instant Messaging) are multi-tasking tools that allow people to communicate and, at the same time, engage in other activities [16]. It is more immediate than email and communication happens in (near) real time [16].

Despite the advantages provided by these tools, users are typically overwhelmed with the amount of data produced by such communications resulting in a problem known as information overload [24, 4]. Workplace productivity can be affected if such information is not properly organized in order to effectively manage ongoing tasks [24]. A key requirement to address this issue is that of automatically identifying actions from email conversations and short texts [10]. Here, an *action* is something that is done by an actor (e.g., a person) or it can be a request, making a commitment to the user interest in natural language conversations [7, 15, 18]. An *action type* represents a particular action in a process. For example, the e-commerce process includes action types such as "*Deliver, Cancel, Return, Payment and Purchase*". For instance, *"I will contact customer services if it doesn't arrive on time"* is an example of a particular action type: "*Deliver*". The identification of such action types are important for the categorization and handling of conversations.

Conversations typically have many actions, which can be identified using manual annotation [11]. This approach can lead to unidentified actions that may result in delays and other issues that affect the organization's activities. The automatic extraction of actions from emails and instant messages can help workers increase their productivity. In this context, multiple studies were conducted in the past to identify actions from emails and collaboration tools [4, 7, 14, 22]. These studies typically use supervised methods for action extraction, which requires labelled data. Such requirement results in important limitations when these supervised approaches need to be extended to various actions. Data labelling is one of the main concerns for action extraction using these approaches and it is important to use large unlabelled data to improve the results [6, 27, 12]. In this paper, we propose an unsupervised approach and focus on the extraction of *action types* such as "*Deliver, Cancel, Return, Payment and Purchase*" from short text conversation, then we showcase the integration of our approach into a widely used collaboration tool (Slack).

The key contribution of this paper consists in an approach that allows for (i) analyzing conversation data from collaboration tools involving short text conversations, (ii) training large volumes of data without labelling, which can help overcome the limitations of supervised learning approaches, and (iii) seamlessly and effectively adapting to scenarios and domains with multiple actions.

2 Related Work

The problem of action identification from text has been studied extensively in previous research. For example, emails are classified based on the intention of the email sender using nouns *(e.g., information, meeting, task)* and verbs *(e.g., pro-*

pose, commit, deliver) called email speech act [7], which identifies the intention in emails such as *"request, propose, amend, commit or deliver"*. Previous research highlighted that users feel overloaded when tracking large number of threads and length of the intervals between messages in email threads [24, 4], which involves manual identification of action hidden in the emails. Another study focused on finding the strength of request and commitment in workplace emails using exploratory annotation tasks [15]. The result encourages automatic detection of request and commitment in emails. The classification- or prioritization-based methods are used to group email based on actions [13, 21, 25]. This method works at the level of the entire email including all statements from the email and do not focus on individual statements [13]. eAssistant [18] learns action verbs and related set of features to identify actionable items such as requests and promises. These studies are focused on identifying generic classes such as request and commitment actions in emails. In our work, instead, we focus on *action type* extraction to identify specific actions such as "*Deliver, Cancel, Return, Payment and Purchase*" from short text conversation data. These short texts have less than 200 characters (i.e., data is sparse) and as a result information extraction becomes harder [9].

The Semanta system [20] was implemented for identifying actions based on conent summarization of emails. It extracts types of actions (e.g., file request, task assignment) from emails. A rule-based method [21] is used to classify actions into categories such as "*Request, Suggest, Assign and Deliver*". These methods need human intervention to classify emails and involve manual annotation and user reviews [20, 21]. In contrast to the text typically found in e-mails, instant messaging texts are fairly short and include idioms and abbreviation [1], making the task of automatic extraction of action types more challenging.

Finally, the work presented in [26] proposes an approach to the problem of event type recognition, which leverages word embedding techniques [17] to find events by averaging the vectors of words in n-grams. Word embedding techniques can help group semantically similar words to find relevant actions. In this paper, we propose an unsupervised method leveraging word embedding for short text conversation.

3 Action type framework

The action type extractor framework is illustrated in the Figure 1. The initial layer has *relation extractor* to extract relation tuples *(subject, relation, object)* from seed data. The open information extraction techniques [19] are used for this task. The relations from these tuples are used to initially identify the action types such as *"Deliver, Cancel, Return, Payment and Purchase"*. The extracted relation includes relevant actions, other actions or no actions. In *Seeds selection*, the relations that are related to action types such as *"Deliver, Cancel, Return, Payment and Purchase"* are selected. If the selected seed relations are used directly for action extraction then it will be able to filter only limited actions (those matching the seed relations). We convert the seed relation words to its vector

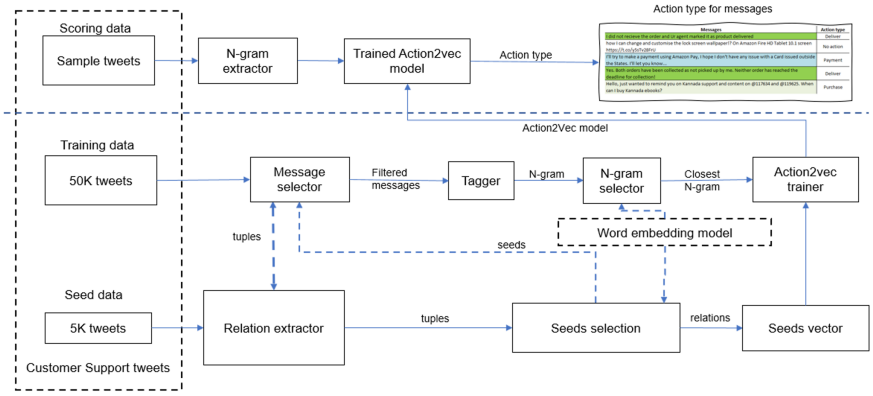

Fig. 1. Action type extractor framework

form using word embeddings [17] in order to capture the words semantically similar to each other. This can help us capture actions semantically related to the selected seed relations (i.e., action types such as *"Deliver, Cancel, Return, Payment and Purchase"*). The *message selector* filters the messages that match the seeds for each action type. This is, it picks the relevant message from large datasets to train the action2vec model. The *tagger* extracts nouns, verbs and adjectives from the message, which are used to formulate bi-gram, tri-grams, etc. The n-grams with the closest distance w.r.t. the seeds vector are obtained by the *N-gram selector*. The *Action2vec trainer* improves the vector and produces a final trained *action2vec model* used to find the closest action types for messages.

4 Action type extraction

The customer support tweets [3] are considered as data source for training our action2vec model. The action type extractor is used to extract actions from tweets. The nouns, verbs and adjectives are extracted from each tweet to identify actions. Verbs such as *"Deliver, Cancel, etc."* and noun such as *"Product, Time, etc."* and adjective such as *"Many, Next, etc."* are extracted to identify actions. In our approach, we extract n-grams and do not restrict ourselves to bi-grams or tri-grams only. This will help us improve the action2vec model to include all keywords from tweets so that we can train our model without such restrictions.

The action type extractor leverages the vector space model (VSM) [17] to extract vectors for each word (nouns and verbs) found in a tweet and calculate a *tweet vector* by averaging the vectors of individual words contained in the tweet. The similarity ratio between the *tweet vector* and action2vec model of all action types are then calculated. The action type which has highest similarity ratio with the *tweet vector* and above a given threshold is considered as action type for a given tweet.

4.1 Seed generation

The seed generation is an initial step to build n-grams for action types. The tuples *(subject, relation and object)* are extracted using Stanford OpenIE [2]. For example, the tweet "I've purchased a gift card for a second time and the first arrived within minutes" has tuples (first, arrive within, minute). The relation "arrive within" related to action type *"Deliver"* is considered as a seed. The *"Deliver"* action type has relations such as *"arrive within", "deliver product", etc.* after excluding stop words. The seeds are extracted from tweet samples for each action type. The word embedding model [17] is used to calculate vectors for seeds in order to create our so-called seeds vectors.

The n-grams vector is created by calculating the average of vectors for each word in the n-grams [26]. For example, *vector("arrive")* and *vector("within")* are extracted for the seed "arrive within". Thus, the individual vectors are averaged to create a vector for the seed "*arrive within*". Similarly, the average of all seeds of a particular action type *("arrive within", "deliver product", etc.)* is used as seeds vector. Eq. 1 and 2 show the calculation of seed and seeds vector.

$$\overrightarrow{seed} = \frac{\sum_{i=1}^{m} \overrightarrow{word_i}}{m} \tag{1}$$

$$\overrightarrow{seeds} = \frac{\sum_{j=1}^{n} \overrightarrow{seed_j}}{n} \tag{2}$$

Here, $\overrightarrow{seed}$ is a vector of a seed by averaging all words for each relation. $\overrightarrow{seeds}$ is a vector of seeds by averaging all seeds for each action type. m is number of words in the relation. n is number of seeds in the action type.

The seeds are created based on specific keywords from a small sample. We need to train seeds vector with larger training sample to find all relevant tweets that belong to the action type. We create action2vec model using a larger sample based on seeds of each action type.

4.2 Action2Vec model

A large set of tweets are considered to create our action2vec model. The relation phrase are extracted for each tweet. The tweets with relation phrases matching with our seeds are selected for training the action2vec model. The nouns, verbs and adjectives of the selected tweets are extracted for training. The vector is extracted for each word of n-grams *(nouns, verbs and adjectives)* and averaged to create a message vector. This message vector is compared with the seeds vector to find the similarity ratio. The cosine similarity is calculated based on the orientation of two vector (x,y). Eq. 3 shows the dot matrix of two vectors (x,y).

$$\cos\theta = \frac{\overrightarrow{x} * \overrightarrow{y}}{||\overrightarrow{x}||||\overrightarrow{y}||} \tag{3}$$

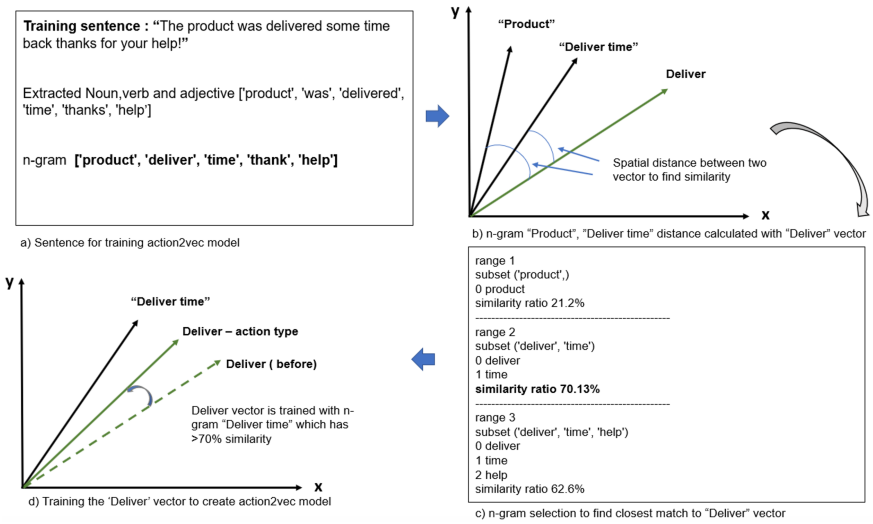

Fig. 2. Training action2vec model

The threshold 70% is considered (based on experiments) to find the relevant action type for the tweet. If similarity ratio exceeds 70% then the message vector and seeds vector are averaged to train action2vec model. This process is repeated for all tweets in the training data. Figure 2 shows training of action2vec model from training data.

During training, words in tweets representing nouns, verbs and adjectives do not have equal importance. This means that some of the words increase the similarity while others decrease it with the seeds vector. To overcome this issue, we have come up with combinations of up to 5 words for n-grams *(noun, verb and adjective)* to find similarity with the seeds vector. The vector from each combination are compared with the seeds vector and then the highest similarity vector is selected to train action2vec model. This method filters out irrelevant combinations for training the action2vec model and helps reduce noise in the model. Figure 2 c) shows the selection of n-grams based on similarity with seeds vector.

5 Dataset and Evaluation

In order to evaluate the feasibility of our approach, we used tweets data to train our action2vec model and compared it with seeds vectors. The large collection of customer support Twitter data [3] was used for the analysis. There are approximately 3 million tweets from multiple brands such as Apple, Amazon, Uber and British Airways. It contains inbound tweets from customers and replies from customer support representatives. The customer support Twitter data has short

text conversations and different types of action. The major e-commerce brand Amazon is selected because it has multiple action types such as *"Deliver, Cancel, Return, Payment and Purchase"* for e-commerce processes. A total of 128K Amazon inbound tweets are considered for our analysis.

5.1 Seed data

The sample data for seeds are extracted from Amazon tweets data. A random sample of 5K tweets (out of 128K tweets) are extracted to find seeds for action types. The Stanford OpenIE [2] is used to find the relations out of these 5K tweets. The extracted relation phrase from tuples are used to identify the relevant seeds for each action type. We focused on action types such as *"Deliver, Cancel, Return, Payment and Purchase"*, which are available in the majority of tweets. For example, the tweet *"I don't want to return all 3 items though, I just want to send one back and have it replaced with the correct one, will that be ok?"* has the tuples *(I, send back, one)*. The seed *"send back"* is relevant to *"Return"* action so it is included in seeds for *"Return"*. The relation that are not relevant to any action types are ignored. For example, *"I tried talking to your chatbot. It wasn't very helpful."* has the tuples (I, talk to, your chatbot) and not related to any of action types. For instance, a total of 57 seeds out of 5K tweets was extracted for the *"Deliver"* action type. Table 1 shows the number of seeds and examples for each action type.

Table 1. Seeds data

Action type	No. of seeds	Example
Deliver	57	arrive within, deliver thing to
Cancel	21	cancel order via, ask refund for
Return	16	send back, replacement
Payment	18	manage payment through, take money from
Purchase	32	order package for, place order on

5.2 Training data

A large set of tweets (50K out of 128K tweets) is used for training action2vec model. The relation phrases are extracted from each tweet in the training data. The relation phrases that match with seeds are filtered for training specific action types. For example, the tweet *"It should arrive in the next couple of days. Still not same day."* has the tuple (it, should arrive in, couple of day), where the relation "should arrive in" matches with seeds in the *"Deliver"* action type. This tweet is considered for training the action2vec model for the *"Deliver"* action type. Using this approach, for instance, a total 1,863 tweets were filtered out of 50K tweets for the *"Deliver"* action type.

Armed with these training data (tweets), we train our action2vec model for each action type. Figure 2 shows examples of the training tweets for action2vec model. The message vector from tweets that have highest similarity with seeds vector are used to train the action2vec model. The number of n-grams selected to train action2vec model are *"Deliver"* (753), *"Cancel"* (148), *"Return"* (167), *"Payment"* (236) and *"Purchase"* (268).

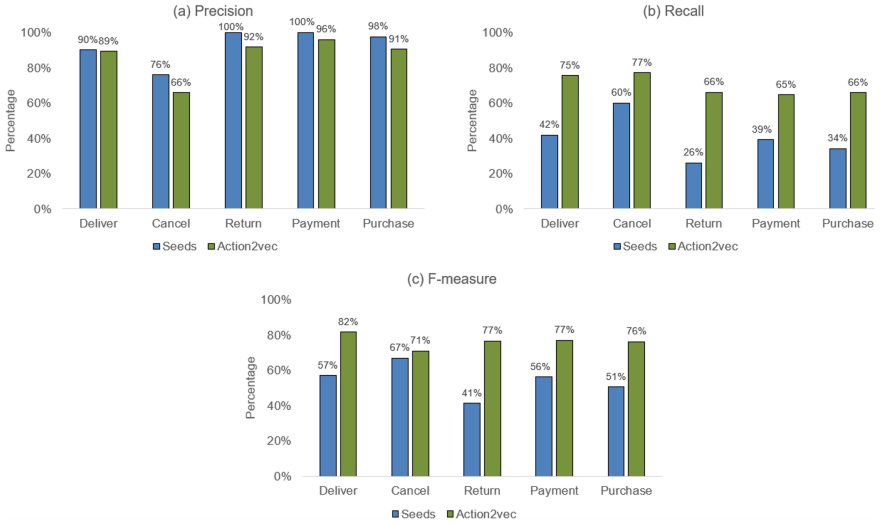

Fig. 3. Seeds vector and action2vec model comparison

5.3 Experiment and results

There are 5 action types considered for evaluation. We prepared an evaluation dataset with tweets for action types *"Deliver"* (215), *"Cancel"* (217), *"Return"* (211), *"Payment"* (204) and *"Purchase"* (225). Each category contains 50% of tweets related to particular action type and the remaining 50% either have other action types or no actions whatsoever.

The evaluation is done for seeds vector and action2vec model separately. We do this to find out the benefits of using our unsupervised training method. We feed the tweets from our evaluation dataset to both seeds vector and action2vec model to get the respective scores. The threshold is set to 60% based on our experiments as the similarity ratio threshold for each action type. We use precision, recall and F-measure metrics to compare the results of the seeds vector and action2vec model.

The seeds vector achieved better precision than action2vec model for all action types as shown in Figure 3. For example, for the action type *"Purchase"*

seeds vector has 98% precision comparing to 91% precision of action2vec model. The recall is very low for seeds vector comparing to action2vec model. This shows action2vec model is able to identify most of the actions compared to seeds vector. The *"Cancel"* action type has achieved the highest recall (60%) for seeds vector among all action types. The action2vec model has more than 70% in recall for *"Deliver"* and *"Cancel"* and more than 60% for other action types. The action2vec model has higher F-measure for *"Deliver"* (82%), *"Cancel"* (71%), *"Return"* (77%), *"Payment"* (77%) and *"Purchase"* (76%). The results show that a large set of training data has improved action2vec model and it is more effective in finding all action types. Figure 3 shows all the metrics used for comparison of seeds vector and action2vec model results.

We demonstrated how we trained and evaluated our unsupervised method for 5 action types. Our approach can be easily extended to different actions. The relevant seeds from sample data need to be identified initially for each action. Then our unsupervised training method can create the action2vec model for action types by training with large datasets. This unsupervised method can be adapted to different action types and domains with less effort as compared to other approaches that heavily depend on data labelling.

a) Action2vec

Actions	Precision	Recall	F-measure
Deliver	89%	75%	82%
Cancel	66%	77%	71%
Return	92%	66%	77%
Payment	96%	65%	77%
Purchase	91%	66%	76%

b) SVM

Actions	Precision	Recall	F-measure
Deliver	84%	75%	79%
Cancel	65%	60%	63%
Return	95%	75%	84%
Payment	93%	63%	75%
Purchase	86%	69%	77%

Fig. 4. Comparison of action2vec and SVM results

6 Comparison with Machine learning method

The unsupervised method action2vec has been evaluated further by comparing with support vector machine (SVM). SVM is a supervised machine learning method that can be used to classify natural language text. We used a SVM classifier from study [5] to find actions from emails.

Given its supervised nature, this classifier requires labelled data. We have prepared labelled tweets for action types such as *"Deliver"* (312), *"Cancel"* (306), *"Return"* (318), *"Payment"* (317) and *"Purchase"* (307) to train the SVM model. The labelled data is balanced to have unbiased training data. The n-grams from nouns, verbs and adjectives of each tweet sentences were extracted. Stop words were removed and verbs were converted to present tense. The vector is extracted for each word of the n-grams and averaged to calculate the message vector. The

message vector has labelled action types that are used to train SVM machine learning model. The SVM model is used to score the evaluation data. The threshold 60% is considered for SVM score to find action types based on experiment to have better precision and recall.

The results show that action2vec model has better F-measure for *"Deliver"* (82%), *"Cancel"* (71%) and *"Payment"* (77%) comparing to the SVM model. This confirms that unsupervised method action2vec has similar results comparing to supervised machine learning approaches. Figure 4 outlines the precision, recall and F-measure numbers for both action2vec and SVM.

7 Integration with collaboration tool

The action type extraction from short messages can be helpful in the workplace environment. In this context, workers are typically engaged with multiple, concurrent tasks from different projects and workflows. The automatic extraction of actions from conversations can help workers as an alert/notification mechanism that contribute to avoid unnecessary delays in the corresponding processes and increasing their productivity.

We have integrated our action type extractor in a state of the art collaboration tool: Slack. Slack is a team messaging application widely used nowdays in the workplace to facilitate inter- and intra-communication in organizations. Our action type extractor was implemented as a Slack channel that extracts actions from text. Here, if a user enters a text in the conversation channel it will be compared with our action2vec model for computing a similarity ratio. The action type extractor then selects an action type based on similarity ratio and provides a relevant action type for the given text. For example, if a user asks *"why does amazon take my money but then cancel my order?"*, the action type extractor identifies the action type *"Cancel"*.

8 Conclusion and future work

We have explored the action type extraction problem on short text conversation data from collaboration tools. The unsupervised method introduced in this paper for action type extraction helps to train models with large amounts of data. It avoids data labelling, which is one of the major issues found in existing approaches for training large text data. We have evaluated the trained action2vec model with an initial seeds vector. The large set of training data is helpful to improve the model. Our action2vec model is compared with an SVM-based machine learning approach. The results show that action2vec model is comparable to the supervised counterpart. Our unsupervised action type extraction method is adaptable and can be extended to other action types and domains. To showcase our proposed solution, we integrated the action type extractor with Slack to provide assistance to workers in extracting action types. In future work, we will collect user feedbacks and apply deep learning algorithms to update the model in near real time.

References

1. Almeida, T.A., Silva, T.P., Santos, I., Hidalgo, J.M.G.: Text normalization and semantic indexing to enhance instant messaging and sms spam filtering. Knowledge-Based Systems **108**, 25–32 (2016)
2. Angeli, G., Premkumar, M.J.J., Manning, C.D.: Leveraging linguistic structure for open domain information extraction. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). pp. 344–354 (2015)
3. Axelbrooke, S.: Customer support on twitter (2017), https://www.kaggle.com/thoughtvector/customer-support-on-twitter
4. Bellotti, V., Ducheneaut, N., Howard, M., Smith, I.: Taking email to task: the design and evaluation of a task management centered email tool. In: Proceedings of the SIGCHI conference on Human factors in computing systems. pp. 345–352. ACM (2003)
5. Bennett, P.N., Carbonell, J.: Detecting action-items in e-mail. In: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 585–586 (2005)
6. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Proceedings of the eleventh annual conference on Computational learning theory. pp. 92–100 (1998)
7. Cohen, W.W., Carvalho, V.R., Mitchell, T.M.: Learning to classify email into"speech acts". In: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (2004)
8. Ducheneaut, N., Bellotti, V.: E-mail as habitat: an exploration of embedded personal information management. interactions **8**(5), 30–38 (2001)
9. Faguo, Z., Fan, Z., Bingru, Y., Xingang, Y.: Research on short text classification algorithm based on statistics and rules. In: 2010 Third international symposium on electronic commerce and security. pp. 3–7. IEEE (2010)
10. Kateb, F., Kalita, J.: Classifying short text in social media: Twitter as case study. International Journal of Computer Applications **111**(9) (2015)
11. Khoussainov, R., Kushmerick, N.: Email task management: An iterative relational learning approach. In: Proc. Conf. Email and Anti-Spam (2005)
12. Kiritchenko, S., Matwin, S.: Email classification with co-training. In: Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research. pp. 301–312. IBM Corp. (2011)
13. Kushmerick, N., Lau, T., Dredze, M., Khoussainov, R.: Activity-centric email: A machine learning approach. In: Proceedings of the National Conference on Artificial Intelligence. vol. 21, p. 1634. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999 (2006)
14. Lampert, A., Dale, R., Paris, C.: Detecting emails containing requests for action. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. pp. 984–992. Association for Computational Linguistics (2010)
15. Lampert, A., Paris, C., Dale, R., et al.: Can requests-for-action and commitments-to-act be reliably identified in email messages. In: Proceedings of the 12th Australasian Document Computing Symposium. pp. 48–55. Citeseer (2007)
16. Maina, T.M.: Instant messaging an effective way of communication in workplace. arXiv preprint arXiv:1310.8489 (2013)

17. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
18. Nezhad, H.R.M., Gunaratna, K., Cappi, J.: eassistant: Cognitive assistance for identification and auto-triage of actionable conversations. In: Proceedings of the 26th International Conference on World Wide Web Companion. pp. 89–98. WWW '17 Companion, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland (2017). https://doi.org/10.1145/3041021.3054147, https://doi.org/10.1145/3041021.3054147
19. Niklaus, C., Cetto, M., Freitas, A., Handschuh, S.: A survey on open information extraction. arXiv preprint arXiv:1806.05599 (2018)
20. Scerri, S., Davis, B., Handschuh, S., Hauswirth, M.: Semanta–semantic email made easy. In: European Semantic Web Conference. pp. 36–50. Springer (2009)
21. Scerri, S., Gossen, G., Davis, B., Handschuh, S.: Classifying action items for semantic email. In: LREC (2010)
22. Shim, E., Singla, V., Krishnaswamy, V.: Application frameworks and methods for assisting tracking of actionable items. In: Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication. p. 5. ACM (2012)
23. Turner, T., Qvarfordt, P., Biehl, J.T., Golovchinsky, G., Back, M.: Exploring the workplace communication ecology. In: Proceedings of the SIGCHI conference on human factors in computing systems. pp. 841–850 (2010)
24. Whittaker, S., Sidner, C.: Email overload: exploring personal information management of email. In: Proceedings of the SIGCHI conference on Human factors in computing systems. pp. 276–283 (1996)
25. Yang, Y., Yoo, S., Lin, F., Moon, I.C.: Personalized email prioritization based on content and social network analysis. IEEE Intelligent Systems (4), 12–18 (2010)
26. Zamanirad, S., Benatallah, B., Barukh, M.C., Rodriguez, C., Nouri, R.: Dynamic event type recognition and tagging for data-driven insights in law-enforcement. Computing pp. 1–25 (2020)
27. Zelikovitz, S., Hirsh, H.: Improving short text classification using unlabeled background knowledge to assess document similarity. In: Proceedings of the seventeenth international conference on machine learning. vol. 2000, pp. 1183–1190 (2000)