Providing Employee Support Through Named Entity Recognition

Harsh Trivedi, Michael Hsieh, Mohit Patel, Tithi Patel

***Abstract*—Employee productivity and satisfaction are absolutely essential to businesses. If employees are unable to quickly get support on problems they have, frustrations rise, output drops, and a business's sales will suffer in the long run. In a large company with many employees spread across departments with different responsibilities, it takes time to forward an issue to someone who can address it. A support team may handle an employee support channel in an online workspace, such as Slack. The team must read through all employee messages, decide who to send each message to, and coordinate further actions between the people involved. Much cost and effort is spent to do this.**

SWIFT is a Slack plug-in application which will automate sending employee messages to the right people. It looks at individual messages in a Slack support channel and routes each problem to the proper individuals.

Introduction

A company may have a dedicated employee support channel and support team, but the time and cost to send messages to appropriate departments is still prohibitive.

The company for our project has an online workspace in Slack to communicate between departments. This online workspace has an employee support channel, general channel, IT channel, health channel, and finance channel. Each channel corresponds with a specific team or department, for example the finance channel has members of the finance department. The exception is the general channel which is accessible by all departments. An employee in the company would like to be able to send a message to the support team when he/she has an issue, and not have to be delayed waiting for the support team to send that message to the appropriate channel.

Materials And Methods

The solution needed was to automate the process of sending employee messages to the correct department. This was done by creating an app using the Slack API and identifying words of employee messages which correspond to specific departments using Named Entity Recognition (NER).

First, the app was installed in the Slack workspace with necessary permissions to read and view parts of the workspace. The app would have to identify all relevant channels by their channel ID, read the most recent message posted on the support channel, perform NER on the message, and then send the message to the right channel.

The app functioned similarly to a bot in that it performed actions automatically. The Slack channel list method was used to display all the channels currently available in the workspace and get their channel IDs. These channel IDs were matched to the name of the support channel and each department channel. Then, the app had to detect whenever a message was posted to the support channel. This was accomplished by using the Slack Events API. The API allows the app to subscribe to be notified of events. Events were filtered so the app only detected the event of a message being sent to the support channel. After the event was detected, the Slack channel history method was used to read the latest message in the support channel. Finally, after NER was performed, the message was automatically sent to the right channel with the channel postMessage method. A notification was also sent to the employee poster to confirm his/her message had been sent to a department.

Named Entity Recognition locates and classifies named entities in text into predefined categories. This makes it a good fit to find meaning in employee messages. When NER is performed on an employee message, the named entities contained in the message are extracted which will allow the right department to be found.

Out of the many NER implementations available, we decided to use the JavaScript version of Stanford NER. Stanford NER implements a Conditional Random Field (CRF) machine learning model to classify categories.

The Conditional Random Field Model

The CRF model applies logistic regression to sequential inputs.

The equation is therefore similar to logistic regression.

A CRF model has several feature functions. Each feature function f_j takes as input a sentence s, the position i of the word we're categorizing in the sentence, the label, or category, l_i of the current word, and the label l_{i-1} of the previous word.

A feature function looks like:

$$f_j(s, i, l_i, l_{i-1})$$

We assign each feature function to a weight. We can add up all the weighted features over all words of a sentence to get

a score, and finally we can transform the score into a category probability by exponentiating and normalizing the score.

The score is:

$$score(l|s) = \sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_j f_j(s, i, l_i, l_{i-1})$$

where λ_j is the weight and f_j is the feature function.

The final CRF equation is:

$$p(l|s) = \frac{exp[score(l|s)]}{\sum_{l'} exp[score(l'|s)]} = \frac{exp[\sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_j f_j(s,i,l_i,l_{i-1})]}{\sum_{l'} exp[\sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_j f_j(s,i,l'_i,l'_{i-1})]}$$

CRF takes into account the previous words of a sentence, which is useful because parts of a sentence can depend on previous words.

The original Stanford 3 class model classifies words and phrases into Location, Person, and Organization categories. The model accepts a text file containing sentences as input. When the model is run, the words and phrases associated with the categories are tagged and returned as output. These are the named entities. However, the categories of the original model were not specific enough to be useful.

Model training and testing

We needed to train the NER model on a new training set of more specific categories and evaluate the model with a test set. To do this, we needed training data, formatted into a file the NER model would accept, and similarly test data in the same format as the training data.

To get data, we used the Mockaroo website to obtain several CSV files. The CSV files contained 1,000 rows each of department relevant features. We had random usernames, IP addresses, and MAC addresses for IT, drug brand names for health, and credit card types for finance.

To do training, the Stanford NER model accepts files in TSV (Tab-Separated Values) format. This means sentences must be broken down into one word each line, and next to each word is the category the word is associated with. We could not use the original categories provided by the model since these were too generic. Our new categories were the channel names of the Slack workspace, namely IT, finance, and health. We converted the CSV files to tab-separated lines, and combined them together to form test and train TSV files.

In the train TSV file, we added additional words common to each specific department. For example, the IT department would likely get issues involving the word “computer”.

In the case our model did not tag any relevant entities, we decided to send the message which was read to the general channel. We added non-related words from a large text corpus into the train TSV file to aid our model in catching unrelated messages.

Then we were able to run training on the TSV file to create a new NER model.

We used this model to perform NER tagging. This means the message most recently sent to the employee support channel was read by the NER model, then the named entities were extracted by the model, along with the predicted category. We then used the category to send that message to the correct department channel.

Results

The issue which we solved here was to decrease the overhead time in forwarding the support requests from an employee in an organization to the support staff of an organization. The employees can post messages on the support channel of the organization in Slack which will be forwarded to the concerned department based on Named Entity Recognition tagging. The application will be automatically forwarding the requests without any human help by categorizing the complaint.

The application will forward the messages to the four channels i.e. Health, Finance, IT Support and General. Once the message has been posted to the relevant department’s Slack workspace, the acknowledgement will be sent to the user from Slack’s built-in Slackbot. The application will result in saving a lot of time and effort for support staff and will result in faster response time to requests.

Future Enhancements

The SWIFT Slack plug-in application is currently able to only filter out messages for certain support departments. In the future we intend to increase the number of departments the SWIFT application can support. This will be an instrumental step as different organizations will have different support departments to handle diverse support requests from employees.

Acknowledgment

We are extremely thankful to Dr. Rakesh Ranjan for his inputs and views on our project’s initial definition that helped us mold our definition into what we have today. We are privileged to have received guidance and support from him in successfully executing our project. Also, because of the hard deadlines set by him we were able to finish our project duly.

The curriculum for the course Enterprise Software Technologies exposed us to numerous technologies being used around the world. We ended up incorporating a lot of these technologies in our project. Using these technologies enabled us to make a robust application.

We would also like to thank our seniors for providing support throughout the project. They helped us overcome

many hurdles in the development of our project. The suggestions that we got from our seniors were very valuable. Also, constructive criticism from our peers helped us in shaping our application to meet industry standards.

We consider ourselves fortunate to get constant encouragement and support from our friends and the Computer Engineering Department.

Conclusion

SWIFT is a Slack plug-in application to help medium- to large-scale companies with several departments better communicate their problems. SWIFT resends any messages sent in the support channel of the workspace to the appropriate department. The user sending the message is notified of which channel his/her message has been broadcasted to. This reduces the time and effort spent by employees in going through all the messages in the support channel and in turn boost productivity. Also, the issues reported by the employees are addressed as soon as possible. To sum it up, SWIFT increases the productivity and efficiency of a workplace.

References

[1] https://blog.echen.me/2012/01/03/introduction-to-conditional-random-fields/

[2] https://towardsdatascience.com/conditional-random-fields-explained-e5b8256da776

[3] https://towardsdatascience.com/named-entity-recognition-applications-and-use-cases-acdbf57d595e

[4] https://medium.com/ml2vec/overview-of-conditional-random-fields-68a2a20fa541

[5] https://www.youtube.com/watch?v=rc3YDj5GiVM&t=755s

[6] https://api.slack.com/methods

[7] https://x-team.com/blog/slack-app-step-by-step/