Project 2: Don’t Say Pay

Let’s begin by clarifying the goals of this project. Our understanding is that we are building building an intelligent filtering tool for a planned employee instant messaging service with the following requirements:

- Identify the sentiment and subject of the chat messages
- Compare this to a list of disallowed topics, and filter accordingly
- Allow management to flag messages for removal

Outside of the functionality of the filter itself there are other things we will need to account for:

- The terms of service the users of the chat app agree to
- Laws regarding message logging consent
- Work with the chat app team for key functionality of the chat app itself
    - How the app handles and reports management flagging
    - Where the chat messages are logged
    - When and how in the message process is the filter used
    - Any message encryption

Are there any major requirements missing here? Any additional features that should be considered?

Is there already an existing list of topics or words we can be provided?

What are the planned terms of service on this chat service? Can we collect messages for training purposes? How will collected messages need to be anonymized? What legality does our team need to account for in doing so?

Will the fact there is a filtering algorithm be initially hidden from the users? Is there a plan for when it is discovered?

Acquiring data does not appear to pose too much of an issue to me. Not only could we find plenty of existing data online – for example the Chatbot Dataset Topical Chat set on Kaggle – but with 1.6 million Amazon employees as potential users we could quickly amass extremely relevant data from within the chat app itself.

Without getting too technical we will be facing challenges with using either of those options. Our goal is to identify and label chat messages on whether they include our “taboo” topics, which can be a difficult task for a computer to learn without human-identified data to learn from. I propose we take the cost of outsourcing labeling a few thousand sample messages for topics and sentiments in order to produce a much more initially accurate system.

With properly labeled training data we then have a few possible options for predictive models that will suit our needs. The two I'm proposing are Naive Bayes or Convolutional Neural Networks.

Naive Bayes is the simpler option, but a tried-and-true one nonetheless. Your email likely uses Naive Bayes for spam detection and that's been considered "solved" for decades. The benefits of Naive Bayes when compared to a CNN are that we can start with a smaller dataset, it's more computationally efficient, and we will spend less time both in development and training the model.

However, the resources available to us at Amazon also make a Convolutional Neural Network an attractive option. With a higher upfront cost of the larger labeled dataset we'll need and the potentially higher cost of the added computation power of multiple layers of convolutions we can produce a noticeably more accurate system. This could balance out the cost equation though as fewer false positives and missed messages will undoubtedly lead to the filter remaining obscure for longer, which leads to less attempts to circumvent it.

In summary, Naive Bayes:

- Has a lower upfront cost
- Will be quicker to develop, train, and deploy
- Will not be ultimately as accurate

And a Convolutional Neural Network:

- Will be more accurate
- Will likely remain hidden longer
- Demands more labeled data
- Will take longer to build

Whichever predictive model we chose for deployment we will need to get with the team building the chat service itself. I imagine the filter will take action between the user pressing "send" and the message either being delivered to the recipient or being broadcasted publicly. But exactly what that looks like will depend on the requirements of the chat app team.

While the app is in service we can constantly take in new and relevant data for training from the app use itself. Messages that pass through the filter and are correctly identified can be added as training data. Reports and flags from managers especially will be useful in training the model as users attempt to adapt and chat around the filter.

Performance can be measured quantitatively with the frequency of correctly identified messages against the frequency of false positives and negatives. And qualitative performance is measured in just how apparent the filter is to users and the lengths they go to in their attempts to bypass it, the less we require adapting to them the better.