Development Process for an Automated Tagger for Notmuch

Gifan Thadathil

1 Introduction

Notmuch is an excellent email system for efficient email storage and powerful querying. To provide better email organization, it makes use of "tags." Notmuch only provides the means to tag emails, but leaves the actual tagging to the user. Some users have manually created rule-based scripts to automate this tagging, however updating these rules can become tedious.

To avoid this tediousness, we have created a program that will automatically learn the tagging scheme used by the user from their existing Notmuch database of emails. After adding a call to the learned model in the Notmuch configuration files, any new emails will be automatically tagged.

This document is a detailed account of the exact process taken to develop the tagger. It details each step of development as well as reasoning behind various development decisions. Roughly the development process can be broken down into the following steps:

1. Problem Formulation
2. Literature Review
3. Proposed Solution
4. Data Retrieval
5. Learning Implementation

2 Problem Formulation

Since tagging is assigning a label to an email, we have a multi-label classification problem. That is, if E is our set of emails and T is our set of tags, we want to correctly assign each

email e a subset of tags in T. Furthermore, since email is typically text, this is an instance of text classification, which has a rich literature. Before diving into the literature, however, we need to note the real-world constraints and expectations of the user.

First we discuss some constraints brought by the email format and Notmuch interface. This should give us the background to flesh out a detailed usage scenario from the perspective of a user to get a sense of expectations. Then we will consider some issues that could arise.

2.1 Email Format

The object we are working with are emails, whose basic specification is given in RFC 5322. An example email message is given below.

```
From: Alice <alice@example.com>
To: Bob <bob@example.com>
Subject: Hello
Date: Mon, 1 Jan 2100 10:30:00 -0400
Reply-To: Alice <alice@example.com>

Hi Bob!
```

Note that the message is separated by a blank line into two section. The top section is called the "message header", and the bottom section is called the "message body." The header is comprised of fields that are essentially key-value pairs separated by a colon. A typical user simply can partially read the body of the message in addition to basic headers such as "From" and "Subject" and is able to tag the message appropriately. Therefore, we will make use of the message and the header.

However, we will not use the entire header. The header can contain some standard fields as well as custom ones. A list of standard headers is curated by the IANA. To keep things simple, we will only use the most commonly used standard header fields. Using custom fields and uncommon fields will be unhelpful for the classifier as there will most likely not be enough emails using such fields to help with classification.

2.2 Interfacing with Notmuch

Notmuch does not pull email from a server. Instead it already expects the user has downloaded their mail onto their device in the appropriate format. Once Notmuch is aware of this database of mail, the user can simply type `notmuch new` into a terminal and their mail will be indexed by Notmuch for quick searching. The user can then add tags as they wish. On receieving new email and pulling it locally to their device, the user must again execute

`notmuch new` to index their new mail. Typically, users will run a script with some rules to do "initial tagging." Our tagger should work in a similar fashion. The idea of this program is to eliminate manual creation of the initial tagging script. The user should be able to run the tagger in the same way: as a post-new hook within the Notmuch configuration or within the configuration of their program to pull mail.

Notmuch makes use of a shared C library, but has a well documented API for Python, so we will create our program using Python. In addition, the python bindings allow us to read headers directly from the file rather than the Notmuch database. This is beneficial because Notmuch only indexes some of the headers. We may want more information.

2.3 Usage Scenario

After downloading the program, the user should simple place a call to it in one of their email system configuration files and be set. The only requirement is that the call to the program be executed after a call to `notmuch new`. The program should initialize itself on the first ever call and generate the classification model. From then on, it should automatically tag new emails. It is likely the model will incorrectly tag or fail to tag some emails. If the user fixes these, the model should note these differences and generate a better model. The idea is that the classifier work completely in the background without ever needing explicit work from the user.

2.4