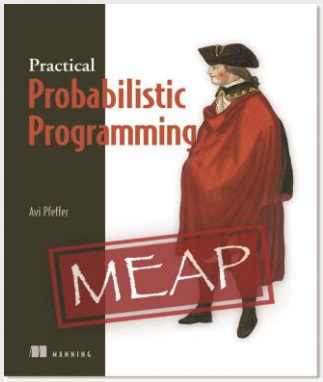

Architecture of a Spam Filter Application

By Avi Pfeffer

A spam filter consists of two components. In this article, based on my book Practical Probabilistic Programming, first describe the architecture of the reasoning component and then the learning component architecture.

A spam filter consists of two components. There is an online component, which performs probabilistic reasoning to classify an email as normal or spam, and filter it or pass it on to the user appropriately. Then there is an offline component, which learns the email model from a training set of emails. In this article, I'll first describe the architecture of the reasoning component and then the learning component architecture. As you'll see, they have a lot of parts in common.

Reasoning Component Architecture

When deciding on the architecture for a component, the first things we need to decide are what the inputs are, what the outputs are, and what the relationships are between them. For our spam filter application, our goal is to take an email as input and determine whether it is a normal email or spam. Because this is a probabilistic programming application, the application won't simply produce a Boolean spam/normal classification as output. Instead, it will produce the probability with which it thinks the email is spam. Also, to keep things simple, we will assume the email input is a text file. To summarize:

Spam Filtering Reasoning Component

- Input:
  - A text file representing an email
- Output:
  - A Double representing the probability that the email is spam

Now, we're going to proceed to build up the architecture of our spam filter step by step.

Step 1: With this in mind, we refer to the basic architecture for a probabilistic reasoning system, and get our first cut at the architecture of our spam filtering application, shown in

Figure 1. Our *Spam Filter Reasoning Component* is used to determine whether an email is normal or spam. It takes as evidence the text of the email. The query is whether the email is spam. The answer it produces is the probability the email is spam.

To get this answer, the reasoning component uses an *Email Model* and an *Inference Algorithm*. The Email Model is a probabilistic model that captures our general knowledge about emails, including both the properties of an email itself and whether or not the email is spam. The Inference Algorithm uses the model to answer the query of whether or not the email is spam given evidence.

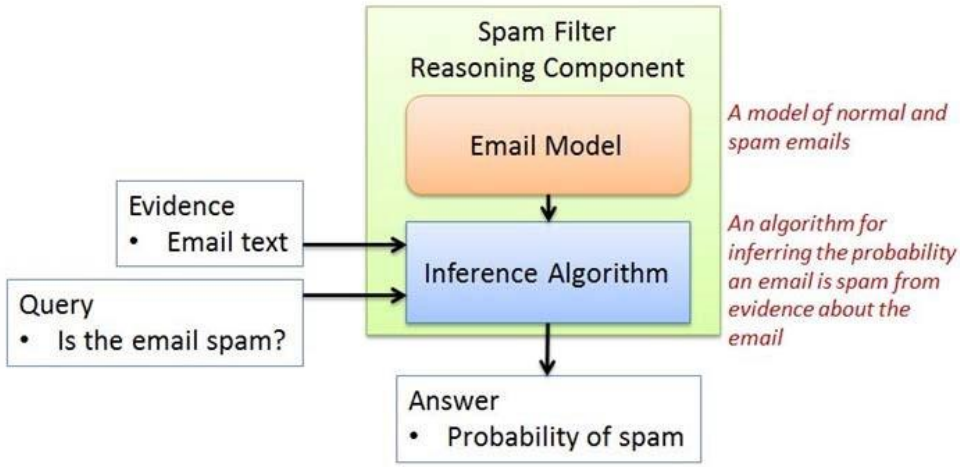

Figure 1: First cut at spam filter reasoning architecture. The spam filter uses an inference algorithm operating on an email model to determine the probability an email is spam given the email text.

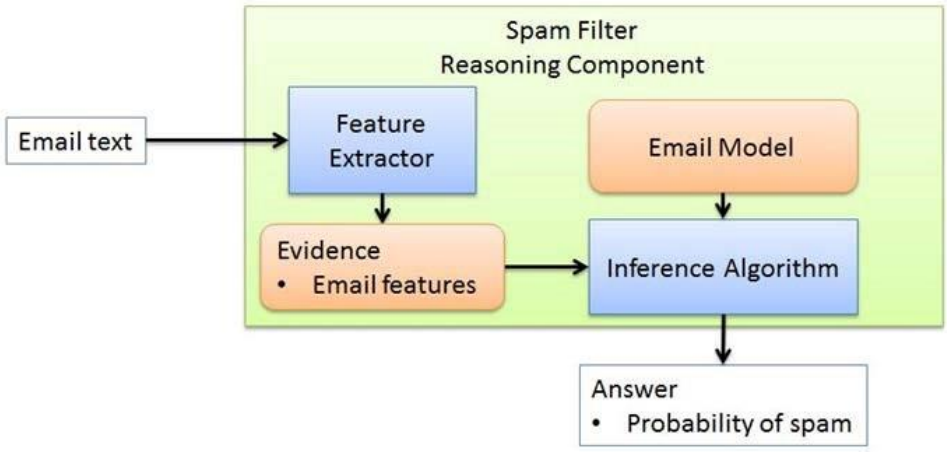

Figure 2: Second cut at spam filter reasoning architecture. We've added a feature extractor and removed the query, since the query is always the same.

Step 3: It's time to take a closer look at the model. Figure 3 shows a refinement of the reasoning architecture that breaks up the model into three parts: the Process, which contains structural knowledge programmed in by the model designer, the Parameters, which are learned from the data, and auxiliary Knowledge that is not directly related to elements but is used for reasoning.

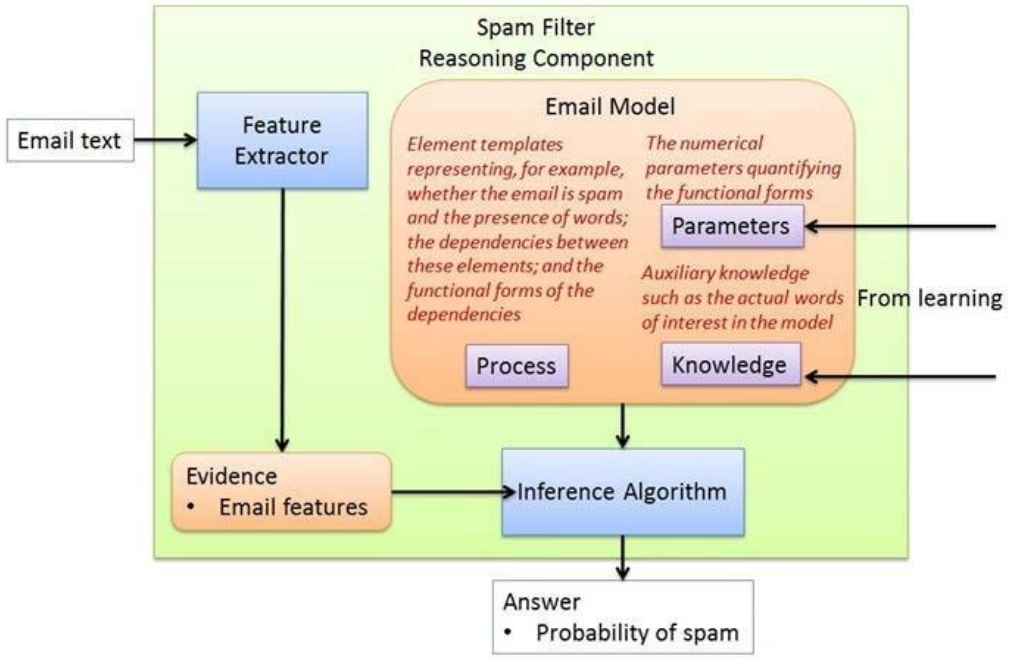

Figure 3: Third cut at spam filter reasoning architecture. The model is broken out into Process, Parameters, and Knowledge. The Parameters and Knowledge come from learning.

Creating a model takes skill, but in general, a probabilistic model has the same kinds of constituents in it. In general, the model contains five separate constituents:

- Templates defining which elements are in the model. For example, there will be an element representing whether the email is spam, and there will be elements representing the presence of particular words in the email. Exactly which words these are is not yet specified; that is part of the Knowledge, which is explained below.
- The dependencies between the elements, for example, that an element indicating the

presence of a particular word depends on the element representing whether the email is spam. In a probabilistic model, the direction of the dependencies is not necessarily the direction of reasoning. You use the words to determine if the email is spam, but in the model, you can think of the email sender as first deciding what kind of email to send ("I'm going to send you a spam email") and then deciding what words go in it. We typically model dependencies in the causal direction. Since the kind of the email causes the words, the dependency goes from the element representing whether the email is spam to each of the words.

- The functional forms of these dependencies. The functional form of an element indicates what type of element it is, such as an If, an atomic Binomial, or a compound Normal. The functional form does not specify the actual parameters of the element. For example, the above dependency takes the form of an If with two Flips: if the email is spam, then the word "rich" is present with one probability, otherwise it is present with some other probability.
- These first three constituents, which make up the Process in Figure 3, are what is assumed by the spam filter application to be known before learning. They are defined by the designer of the application. Together, they constitute the structure of the email model.
- The numerical parameters of the model, which make up the Parameters in Figure 3. For example, the probability that an email is spam is one parameter. The probability that the word "rich" is present if the email is spam is another parameter, while the probability "rich" is present if the email is not spam is a third parameter. The reason the Parameters are separated out from the Process is because the Parameters will be learned from training data.
- The Knowledge in Figure 3 represents auxiliary knowledge used in constructing the model and applying evidence to it in a particular situation. I use this term for anything learned from the training data that isn't specifically related to the elements, their dependencies, functional forms, or numerical parameters. In our spam filter application, this knowledge will consist of a dictionary of the words that appeared in the training emails along with the number of times each word appeared. For example, we need the Knowledge to tell us that the word "rich" is one we're interested in. As a result, the Knowledge helps define what the parameters of the model actually are; the probability "rich" appears if the email is spam is only a parameter if "rich" is a word we're interested in.

At this point, the architecture of the reasoning component is fairly complete. It's time to look at the learning component.

Learning Component Architecture

We know that the job of the learning component is to produce the email model used by the reasoning component given a dataset of emails. Just like we did for reasoning, the first thing

to do is decide what the inputs and outputs of the learning component are. Looking at Figure 3, we know that the output needs to be the Parameters and Knowledge of the email model. But what about the input? We can assume that we're given a training set of emails from which to learn, so that makes up part of the input. But learning is much easier if someone has also labeled some of those emails as normal or spam. We're going to assume that we are given a training set of emails, as well as labels for some of those emails.

However, not all the emails have to be labeled; we can use a large dataset of unlabeled emails in addition to a smaller dataset of emails with labels.

Spam Filtering Learning Component

- Inputs:
  - A set of text files representing emails
  - A file with normal/spam labels for a subset of the emails
- Output:
  - Numerical parameters characterizing the generative process o Auxiliary knowledge to use in the model

Figure 4 shows the architecture of the spam filter learning component. It uses many similar elements to the reasoning component, but with some important differences:

- The learning component centers on a *Prior Email Model*. This model contains the same parts as the Email Model in the reasoning component, except that we use *Prior Parameters* for the parameters instead of parameters produced by the learning component. Prior parameters are the parameters of your model before you've seen any data. Since the learning component has not seen any data before it operates, it uses prior parameter values. The other two parts of the model, the *Process* and the *Knowledge*, are exactly the same as for the reasoning component.
- The Knowledge itself is extracted directly from the training emails by a *Knowledge Extractor*. For example, the Knowledge might consist of the most common words appearing in the emails. This Knowledge is also output from the learning component and sent to the reasoning component.
- Just like in the reasoning component, the *Feature Extractor* extracts features of all the emails and turns them into evidence. The normal/spam labels are turned into evidence for the emails for which we have labels.
- Instead of an inference algorithm, we have a *Learning Algorithm*. This algorithm takes all the evidence about all the emails, and uses the model to learn parameter values. Like reasoning algorithms, Figaro provides a number of learning algorithms.

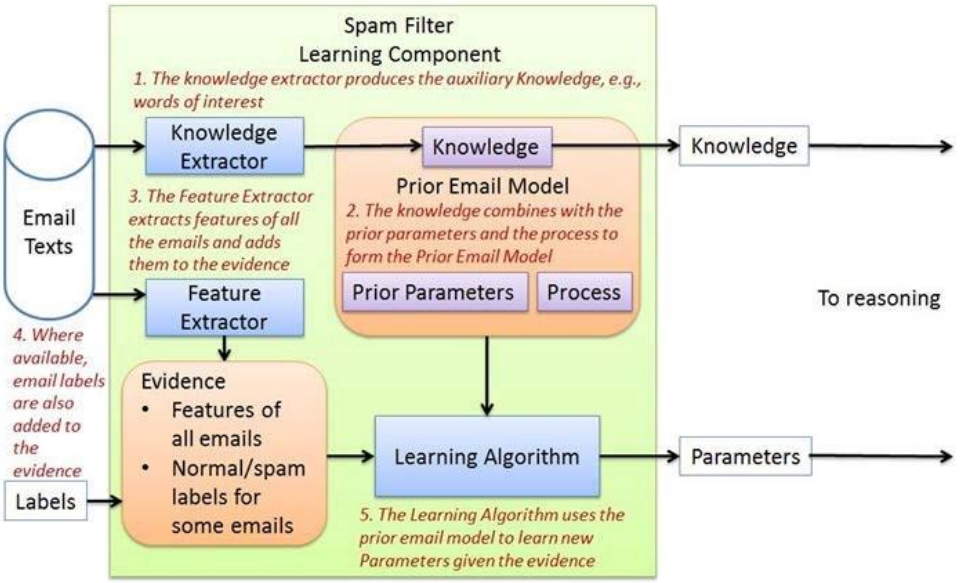

Figure 4: Learning component architecture

Those are all the pieces of the learning component. One of the nice things about our architecture, with its division into reasoning and learning components, is that the learning component can be run once and the results can be used many times by the reasoning component. Learning from training data can be a time consuming operation, and we wouldn't want to do it every time we want to reason about a new email. Our design allows us to do the learning once, export the learned parameters and knowledge, and use them to reason rapidly about incoming emails.