How Does the Postal Service Sort Mail?

Gwyn Whieldon

Hood College

April 14, 2012
The US Postal Service: Facts and Figures

Postal Service Statistics

Gwyn Whieldon

How Does the Postal Service Sort Mail?
The US Postal Service: Facts and Figures

Postal Service Statistics

- The Postal Service employs over 574,000 people (making it the second-largest civilian employer in the United States.)
The US Postal Service: Facts and Figures

Postal Service Statistics

- The Postal Service employs over 574,000 people (making it the second-largest civilian employer in the United States.)
- The USPS delivers approximately 700 million pieces of mail per day, on average (which is less than used to be sent.)
Postal Service Statistics

- The Postal Service employs over 574,000 people (making it the second-largest civilian employer in the United States.)
- The USPS delivers approximately 700 million pieces of mail per day, on average (which is less than used to be sent.)
- This works out to around 1200 pieces of mail processed per employee – which would be impossible to sort by hand.
# The US Postal Service: Facts and Figures

**Postal Service Statistics**

- The Postal Service employs over 574,000 people (making it the second-largest civilian employer in the United States.)
- The USPS delivers approximately 700 million pieces of mail per day, on average (which is less than used to be sent.)
- This works out to around 1200 pieces of mail processed *per* employee – which would be impossible to sort by hand.
- This is where technology will come in!
Where’s This Letter Go?

- Since 1965, the USPS has been using something called *Optical Character Recognition* or OCR, for short.
Where’s This Letter Go?

- Since 1965, the USPS has been using something called *Optical Character Recognition* or OCR, for short.

- This is where they scan an image of the delivery address on the envelope, and convert that address into text.
Where’s This Letter Go?

After reading this address with a machine called a multiline optical character reader (MLOCR), the destination address will be looked up in their database. With this in hand, the letter is stamped with a printed barcode which allows it to be automatically sorted – all the way to the delivery person!
Where’s This Letter Go?

After reading this address with a machine called a *multiline optical character reader (MLOCR)*, the destination address will be looked up in their database.
Where’s This Letter Go?

- After reading this address with a machine called a *multiline optical character reader (MLOCR)*, the destination address will be looked up in their database.
- With this in hand, the letter is stamped with a printed barcode which allows it to be automatically sorted – all the way to the delivery person!

![Image of a letter with an address and barcode]
Where’s This Letter Go?

- After reading this address with a machine called a *multiline optical character reader (MLOCR)*, the destination address will be looked up in their database.
- With this in hand, the letter is stamped with a printed barcode which allows it to be automatically sorted – all the way to the delivery person!
The Math Behind the Magic

We’d like an algorithm to perform the following task:

- **Input:**
- **Output:**
The Math Behind the Magic

We’d like an algorithm to perform the following task:

- **Input:** Picture/scan of text
- **Output:**

Gwyn Whieldon

How Does the Postal Service Sort Mail?
The Math Behind the Magic

We’d like an algorithm to perform the following task:

- **Input:** Picture/scan of text
- **Output:** Content of text
The Math Behind the Magic

We’d like an algorithm to perform the following task:

- **Input**: Picture/scan of text
- **Output**: Content of text
The Math Behind the Magic

We’d like an algorithm to perform the following task:

- **Input:** Picture/scan of text
- **Output:** Content of text

Hood College
401 Rosemont Ave.
Frederick, MD 21701
The Math Behind the Magic

We’d like an algorithm to perform the following task:

- **Input:** Picture/scan of text
- **Output:** Content of text

We’ll use something called a *Bayesian network* for the task.
Bayesian Networks: A Definition

Definition (Bayesian Network)

A *Bayesian network* (also called a *directed acyclic graphical model*) is a directed, acyclic graph with a node for each random variable, and an directed edge from $X \rightarrow Y$ if $Y$ has a conditional dependence on $X$. 
Bayesian Networks: A Definition

Definition (Bayesian Network)

A Bayesian network (also called a directed acyclic graphical model) is a directed, acyclic graph with a node for each random variable, and an directed edge from $X \rightarrow Y$ if $Y$ has a conditional dependence on $X$. 

Gwyn Whieldon How Does the Postal Service Sort Mail?
Bayesian Networks: A Definition

Definition (Bayesian Network)

A Bayesian network (also called a directed acyclic graphical model) is a directed, acyclic graph with a node for each random variable, and an directed edge from $X \rightarrow Y$ if $Y$ has a conditional dependence on $X$. 

Gwyn Whieldon How Does the Postal Service Sort Mail?
Toy Bayesian Network: Medical Diagnoses

A → S, A → C, F → S, F → C, F → B

Random Variables, Symptoms:
- (S)neezing \([0,1]\)
- (C)oughing \([0,1]\)
- (B)legh-ing \([0,1]\)

Random Variables, Illnesses:
- (A)llergies \([0,1]\)
- (F)lu \([0,1]\)
Toy Bayesian Network: Medical Diagnoses

Random Variables, Symptoms:

(S)neezing

(C)oughing

(B)legh-ing

Random Variables, Illnesses:

(A)llergies

(F)lu
Toy Bayesian Network: Medical Diagnoses

Random Variables, Symptoms:
- (S)neezing

Random Variables, Illnesses:
- Allergies
- Flu
- Coughing
- Blushing
Toy Bayesian Network: Medical Diagnoses

Random Variables, Symptoms:
- (S)neezing
- (C)oughing

Random Variables, Illnesses:
- Allergies
- Flu

How Does the Postal Service Sort Mail?
Toy Bayesian Network: Medical Diagnoses

Random Variables, Symptoms:
- (S)neezing
- (C)oughing
- (B)legh-ing

Random Variables, Illnesses:
Toy Bayesian Network: Medical Diagnoses

Random Variables, Symptoms:
- (S)neezing
- (C)oughing
- (B)legh-ing

Random Variables, Illnesses:
- (A)llergies
Toy Bayesian Network: Medical Diagnoses

Random Variables, Symptoms:
- (S)neezing
- (C)oughing
- (B)legh-ing

Random Variables, Illnesses:
- (A)llergies
- (F)lu
Toy Bayesian Network: Medical Diagnoses

Random Variables, Symptoms:
- (S)neezing [0,1]
- (C)oughing [0,1]
- (B)legh-ing [0,1]

Random Variables, Illnesses:
- (A)llergies [0,1]
- (F)lu [0,1]
Toy Bayesian Network: Medical Diagnoses

- $A \rightarrow S$, $A \rightarrow C$

Random Variables, Symptoms:
- (S)neeze: $[0,1]$
- (C)ough: $[0,1]$
- (B)legh: $[0,1]$

Random Variables, Illnesses:
- (A)llergies: $[0,1]$
- (F)lu: $[0,1]$
Toy Bayesian Network: Medical Diagnoses

Random Variables, Symptoms:
- (S)neezing [0,1]
- (C)oughing [0,1]
- (B)legh-ing [0,1]

Random Variables, Illnesses:
- (A)llergies [0,1]
- (F)lu [0,1]
Bayesian Networks and OCR

When we're trying to convert images of text into the text itself, we're going to make a simplifying assumption – that I've already broken up my text into characters. Then the simplest form of our Bayesian network looks like:
Bayesian Networks and OCR

When we’re trying to convert images of text into the text itself, we’re going to make a simplifying assumption – that I’ve already broken up my text into characters.
Bayesian Networks and OCR

- When we’re trying to convert images of text into the text itself, we’re going to make a simplifying assumption – that I’ve already broken up my text into characters.
- Then the simplest form of our Bayesian network looks like:
Bayesian Networks and OCR

In reality though, not all pairs are created equal.
Bayesian Networks and OCR

In reality though, not all pairs are created equal.

\[ X_2 \rightarrow \begin{cases} 
    a \\
    u 
\end{cases} \]
Bayesian Networks and OCR

In reality though, not all pairs are created equal.

\[ X_2 \rightarrow \begin{cases} a \\ u \end{cases} \quad \text{and} \quad X_1 \rightarrow q \]
Bayesian Networks and OCR

In reality though, not all pairs are created equal.

\[ X_2 \rightarrow \begin{cases} a \\ u \end{cases} \quad X_1 \rightarrow q \]

\[ P("qu") > P("qa") \]
Bayesian Networks and OCR

In reality though, not all pairs are created equal.

\[ X_2 \rightarrow \begin{cases} a \\ u \end{cases} \quad X_1 \rightarrow q \]

\[ P(“qu”) > P(“qa”) \quad \text{Our conditional probability should reflect this!} \]
Bayesian Networks and OCR

In reality though, not all pairs are created equal.

\[ X_2 \rightarrow \begin{cases} \ a \\ u \end{cases} \quad X_1 \rightarrow q \]

\[ P(\text{"qu"}) > P(\text{"qa"}) \quad \text{Our conditional probability should reflect this!} \]
Bayesian Networks and OCR

We can go one step further and consider triplet factors. For an alphabet of 26 letters though, this gives 17,567 different conditional probabilities we'd have to record per triple of letters in a word – not desirable! Take top 2000 instead.
Bayesian Networks and OCR

- We can go one step further and consider triplet factors. For an alphabet of 26 letters though, this gives 17,567 different conditional probabilities we’d have to record per triple of letters in a word – not desirable!
Bayesian Networks and OCR

- We can go one step further and consider triplet factors. For an alphabet of 26 letters though, this gives 17,567 different conditional probabilities we’d have to record per triple of letters in a word – not desirable!
Bayesian Networks and OCR

- We can go one step further and consider triplet factors. For an alphabet of 26 letters though, this gives 17,567 different conditional probabilities we’d have to record per triple of letters in a word – not desirable! Take top 2000 instead.
Constructing an Inference Engine

- We can see that this *still* didn’t guarantee 100% accuracy. However, this was a fairly simplistic model – and our inference engine wasn’t optimized for our “handwriting”.

Can add “SimilarityFactors”, which increases the probability that similarly written characters will be given the same values. Our character and word accuracy for each of these was given by:

- **singletonFactors**
  - charAcc: 0.767
  - wordAcc: 0.220
- **pairwiseFactors**
  - charAcc: 0.792
  - wordAcc: 0.260
- **tripletFactors**
  - charAcc: 0.800
  - wordAcc: 0.340
- **similarityFactors**
  - charAcc: 0.816
  - wordAcc: 0.370
Constructing an Inference Engine

- We can see that this *still* didn’t guarantee 100% accuracy. However, this was a fairly simplistic model – and our inference engine wasn’t optimized for our “handwriting”.

- Can add “SimilarityFactors”, which increases the probability that similarly written characters will be given the same values.
Constructing an Inference Engine

- We can see that this *still* didn’t guarantee 100% accuracy. However, this was a fairly simplistic model – and our inference engine wasn’t optimized for our “handwriting”.

- Can add “SimilarityFactors”, which increases the probability that similarly written characters will be given the same values.

Our character and word accuracy for each of these was given by:

<table>
<thead>
<tr>
<th>Factors</th>
<th>charAcc</th>
<th>wordAcc</th>
</tr>
</thead>
<tbody>
<tr>
<td>singletonFactors</td>
<td>0.767</td>
<td>0.220</td>
</tr>
<tr>
<td>pairwiseFactors</td>
<td>0.792</td>
<td>0.260</td>
</tr>
<tr>
<td>tripletFactors</td>
<td>0.800</td>
<td>0.340</td>
</tr>
<tr>
<td>similarityFactors</td>
<td>0.816</td>
<td>0.370</td>
</tr>
</tbody>
</table>
Constructing an Inference Engine
Constructing an Inference Engine

Can buy programs which “train” themselves to read your writing perfectly.
Constructing an Inference Engine

- Can buy programs which “train” themselves to read your writing perfectly.
- Typically use different OCR for handwriting vs. printed block text,
  ...a la Google Books.
Constructing an Inference Engine

- Can buy programs which “train” themselves to read your writing perfectly.
- Typically use different OCR for handwriting vs. printed block text, ...a la Google Books.
- For tablet writing, often add in “stroke analysis” – meaning, how you write a character is as important as what you write.
Thanks!

Acknowledgements: This talk came out of a programming assignment in the Stanford online course: “Probabilistic Graphical Models” by Daphne Koller. While I coded the factor constructions, the overall code structure and inference engine are from her course materials. I would highly recommend this course to anyone interested in these materials!
Thanks!

- Thanks to the Organizers for the opportunity to speak!
Thanks!

- Thanks to the Organizers for the opportunity to speak!
- Acknowledgements: This talk came out of a programming assignment in the Stanford online course: 
  
  “Probabilistic Graphical Models” by Daphne Kollar

While I coded the factor constructions, the overall code structure and inference engine are from her course materials.
Thanks!

- Thanks to the Organizers for the opportunity to speak!
- Acknowledgements: This talk came out of a programming assignment in the Stanford online course: “Probabilistic Graphical Models” by Daphne Kollár.
  While I coded the factor constructions, the overall code structure and inference engine are from her course materials.
- I would highly recommend this course to anyone interested in these materials!