Text Mining for Economics and Finance
Introduction

Stephen Hansen

Introduction

Most empirical work in economics relies on inherently quantitative data: prices, demand, votes, etc.

But a large amount of unstructured text is also generated in economic environments: company reports, policy committee deliberations, court decisions, media articles, political speeches, etc.

One can use such data qualitatively, but increasing interest in treating text quantitatively.

We will review how economists have done this until recently, and then discuss more modern machine learning approaches.

We shall also see that the empirical analysis of text is part of a more general problem of treating high-dimensional count data, and discuss other applications.

Textual Databases

A single observation in a textual database is called a *document*.

The set of documents that make up the dataset is called a *corpus*.

We often have covariates associated with each document that are sometimes called *metadata*.

Example

In "Transparency and Deliberation" we use a corpus of verbatim FOMC transcripts from the era of Alan Greenspan:

- 149 meetings from August 1987 through January 2006.
- A document is a single statement by a speaker in a meeting (46,502).
- Associated metadata: speaker biographical information, macroeconomic conditions, etc.

Data Sources

There are many potential sources for text data, such as:

1. PDF files or other non-editable formats
2. Word documents or other editable formats
3. Web pages
4. Application Programming Interfaces (API) for web applications.

From Files to Databases

Turning raw text files into structured databases is often a challenge:

1. Separate metadata from text
2. Identify relevant portions of text (paragraphs, sections, etc)
3. Remove graphs and charts

First step for non-editable files is conversion to editable format, usually with optical character recognition software.

With raw text files, we can use regular expressions to identify relevant patterns.

HTML and XML pages provide structure through tagging.

If all else fails, relatively cheap and reliable services exist for manual extraction.

What is Text?

At an abstract level, text is simply a string of characters.

Some of these may be from the Latin alphabet—'a', 'A', 'p' and so on—but there may also be:

1. Decorated Latin letters (e.g. ö)
2. Non-Latin alphabetic characters (e.g. Chinese and Arabic)
3. Punctuation (e.g. '!')
4. White spaces, tabs, newlines
5. Numbers
6. Non-alphanumeric characters (e.g. '@')

Key Question: How can we obtain an informative, quantitative representation of these character strings? This is the goal of text mining.

Outline of Course

1. String pre-processing and document-term matrix.
2. Dictionary methods and the vector space model.
3. Basic unsupervised learning methods.
4. Bayesian methods for count data.
5. Latent Dirichlet allocation.
6. Variational inference.
7. Supervised learning.

Grading and Activities

20 hours of lecture covering statistical ideas.

Two two-hour practical sessions with Paul Soto.

40% of your grade will come from assignments (mainly programming in Python, some statistical).

60% of your grade will come from a final project due at the end of term. Your assignments should already lay the foundation for the project.

No final exam.

Textbooks

The following two books will cover the statistical ideas for the course:

1. Manning, Raghavan, and Schütze (2009). *An Introduction to Information Retrieval*. Cambridge University Press.

2. Murphy (2012). *Machine Learning: a Probabilistic Perspective*. MIT Press.

A free copy is available of the first, and other machine learning textbooks' content overlaps considerably with the second.

Focus of Course

We will mainly be concerned with the application of tools in economics and finance rather than questions important to computer scientists.

Examples of the latter might include:

1. What are efficient data structures for holding vectors of word counts?
2. How to process massive corpora like Wikipedia?
3. Which text mining algorithms can be parallelized?

These are less crucial for social science research:

1. Many interesting datasets are not particularly "Big".
2. Much more emphasis on the question being asked with the data.
3. Can seek out specialized help when necessary.

Pre-Processing I: Tokenization

Tokenization is the splitting of a raw character string into individual elements of interest: words, numbers, punctuation. Often we may strip out all non-alphanumeric or non-alphabetic elements.

Simple rules work well, but not perfectly. For example, splitting on white space and punctuation will separate hyphenated phrases as in 'risk-averse agent' and contractions as in 'aren't'.

In practice, you should probably use a specialized library for tokenization.

Pre-Processing II: Stopword Removal

The frequency distribution of words in natural languages is highly skewed, with a few dozen words accounting for the bulk of text.

These *stopwords* are typically stripped out of token lists as they take up memory but do not help distinguish one document from another.

Examples from English are 'a', 'the', 'to', 'for' and so on.

No definitive list, but example on `http://snowball.tartarus.org/algorithms/english/stop.txt`.

Pre-Processing II: Stopword Removal

The frequency distribution of words in natural languages is highly skewed, with a few dozen words accounting for the bulk of text.

These *stopwords* are typically stripped out of token lists as they take up memory but do not help distinguish one document from another.

Examples from English are 'a', 'the', 'to', 'for' and so on.

No definitive list, but example on `http://snowball.tartarus.org/algorithms/english/stop.txt`.

Also common to drop rare words, for example those that appear is less than some fixed percentage of documents.

Pre-Processing III: Linguistic Roots

For many applications, the relevant information in tokens is their linguistic root, not their grammatical form. We may want to treat 'prefer', 'prefers', 'preferences' as equivalent tokens.

Two options:

- *Stemming*: Deterministic algorithm for removing suffixes. Porter stemmer is popular.

  Stem need not be an English word: Porter stemmer maps 'inflation' to 'inflat'. Sometimes equivalence between tokens is misleading: 'university' and 'universe' stemmed to same form.

- *Lemmatizing*: Tag each token with its part of speech, then look up each (word, POS) pair in a dictionary to find linguistic root.

  E.g. 'saw' tagged as verb would be converted to 'see', 'saw' tagged as noun left unchanged.

A related transformation is *case-folding* each alphabetic token into lowercase. Not without ambiguity, e.g. 'US' and 'us' each mapped into same token.

Pre-Processing IV: Multi-Word Phrases

Sometimes groups of individual tokens like "Universitat Pompeu Fabra" or "text mining" have a specific meaning.

One ad-hoc strategy is to tabulate the frequency of all unique two-token (bigram) or three-token (trigram) phrases in the data, and convert the most common into a single token.

In FOMC data, most common bigrams include 'interest rate', 'labor market', 'basi point'; most common trigrams include 'feder fund rate', 'real interest rate', 'real gdp growth', 'unit labor cost'.

More Systematic Approach

Some phrases have meaning because they stand in for specific names, like "Universitat Pompeu Fabra". One can used named-entity recognition software applied to raw, tokenized text data to identify these.

Other phrases have meaning because they denote a recurring concept, like "housing bubble". To find these, one can apply part-of-speech tagging, then tabulate the frequency of the following tag patterns:

$$\text{AN/NN/AAN/ANN/NAN/NNN/NPN.}$$

See chapter on collocations in Manning and Schütze's *Foundations of Statistical Natural Language Processing* for more details.

Example from NYT Corpus

$C(w^1\ w^2)$	w^1	w^2	tag pattern
11487	New	York	A N
7261	United	States	A N
5412	Los	Angeles	N N
3301	last	year	A N
3191	Saudi	Arabia	N N
2699	last	week	A N
2514	vice	president	A N
2378	Persian	Gulf	A N
2161	San	Francisco	N N
2106	President	Bush	N N
2001	Middle	East	A N
1942	Saddam	Hussein	N N
1867	Soviet	Union	A N
1850	White	House	A N
1633	United	Nations	A N
1337	York	City	N N
1328	oil	prices	N N

Pre-Processing of FOMC Corpus

	All terms	Alpha terms	No stopwords	Stems	MWE
# terms	6249776	5519606	2505261	2505261	2438480
Unique terms	26030	24801	24611	13734	13823

From Tokens to Counts

After pre-processing, each document is a finite list of terms.

A basic quantitative representation of a corpus is the following:

- Index each unique term in the corpus by some $v \in \{1, \ldots, V\}$ where V is the number of unique terms.
- For each document $d \in \{1, \ldots, D\}$ compute the count $x_{d,v}$ as the number of occurrences of term v in document d.
- The $D \times V$ matrix $\mathbf{X}$ of all such counts is called the *document-term matrix*.

This representation is often called the *bag-of-words* model, or, in probabilistic modeling, the *unigram* model.

Example

Doc1 = [‘text’, ‘mining’, ‘is’, ‘more’, ‘fun’, ‘than’, ‘coal’, ‘mining’]

has the bag-of-words representation

text	mining	is	more	fun	than	coal
1	2	1	1	1	1	1

Example

Doc1 = [‘text’, ‘mining’, ‘is’, ‘more’, ‘fun’, ‘than’, ‘coal’, ‘mining’]

has the bag-of-words representation

text	mining	is	more	fun	than	coal
1	2	1	1	1	1	1

Note that

Doc2 = [‘coal’, ‘mining’, ‘is’, ‘more’, ‘fun’, ‘than’, ‘text’, ‘mining’]

also shares the same representation.

The bag-of-words model is useful for describing content, but we lose all information about sentence structure.

Ngram Representation

Some limitations of the unigram model can be overcome by counting all unique N-length term sequences in the corpus. This is called the *Ngram* model.

The model with $N = 2$ ($N = 3$) is sometimes called the *bigram* (*trigram*) model.

The bigram representation of Doc1 is

(text, mining)	(mining, is)	(is, more)	(more, fun)	...
1	1	1	1	...

Note that we can still form a matrix of counts with an Ngram model, we just need to redefine the column indices to correspond to the set of unique Ngrams.

Text Mining and Information

Any useful representation of text will throw away some information; that's the essential purpose of text mining.

The question is whether we are keeping the relevant information for our needs, and getting rid of the extraneous information.

The answer cannot be separated from the context in which we are using text. The bag-of-words model can perform very well for identifying documents of interest, and terribly for translation models.

Text as Count Data

For the rest of the course, we will take as given a document-term matrix of a corpus and treat it as a unit for statistical analysis.

But since the document-term matrix is just a matrix of counts, the methods we develop will also be relevant for other applications in which observations are described as count vectors.

Alternative Application I: Survey Data

Ongoing project to document CEO time use (with O. Bandiera, A. Prat, and R. Sadun), and its effect on firm performance.

Data on each 15-minute block of time for one week of 1,114 CEOs' time classified according to

1. type (e.g. meeting, public event, etc.)
2. duration (15m, 30m, etc.)
3. planning (planned or unplanned)
4. number of participants (one, more than one)
5. functions of participants, divided between employees of the firms or "insiders" (finance, marketing, etc.) and "outsiders" (clients, banks, etc.).

There are 4,253 unique combinations of these five features in the data.

One can summarize the data with a 1114×4253 matrix where the (i, j)th element is the number of 15-minute time blocks that CEO i spends in activities with a particular combination of features j.

Alternative Application II: Demand

Increasingly common to get detailed data on consumers' shopping behavior.

Imagine a dataset in which consumer $i \in \{1, \ldots, N\}$ makes multiple visits to a store that sells M possible goods bundles.

Then we can define an $N \times M$ matrix that counts the number of times each consumer buys each bundle.