Document Understanding

Module Introduction

Document Features

We've treated documents as sequences of words so far, but documents have much richer structure and information.

This module surveys many of the extra features we can use as clues of relevance.

http://en.wikipedia.org/wiki/Susan_Dumais

Structural Features

Some of these features are structural: the document's organization gives clues about its topic.

- The title, headings, and menu give fine-grained topic and subtopic information.
- Links and their anchor text provide clues about the relevance of other pages related to this one.

http://en.wikipedia.org/wiki/Susan_Dumais

Topical Features

Other features are topical: the document's text may contain special words and phrases that pertain to a certain topic.

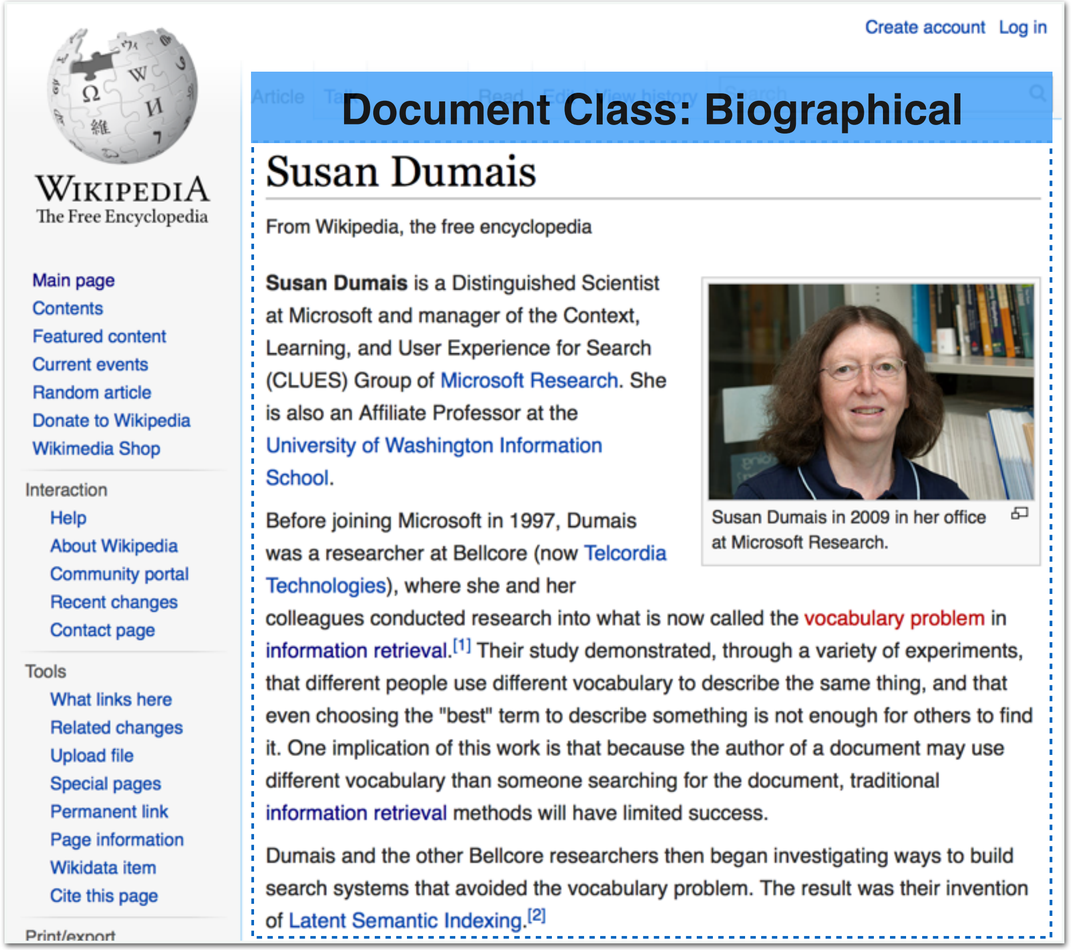
- Named entities (people, companies, places, events…) are strong topical clues.
- Topic modeling discovers the vocabulary that tends to be used when speaking of a certain topic.

http://en.wikipedia.org/wiki/Susan_Dumais

Features from ML

Tools from Machine Learning can be used to generate additional features for a page.

- Document classifiers are used to identify news articles, blogs, reviews, and other types of specialized pages.
- Document clustering can find very similar pages, which is useful for providing diverse result lists and “more like this” functions (e.g. clustering news articles by story).



http://en.wikipedia.org/wiki/Susan_Dumais

Putting it Together

When we have collected all the document features we're interested in, we can use standard Machine Learning classifiers to learn how to predict document relevance from document features.

This makes scoring functions such as BM25 simply one component of a more complicated relevance predictor.

We will see later how to rank using these features. For now, let's focus on the features themselves.

DocID	Body BM25	Title BM25	In-links	…	Rel?
1	1.23	1.2	3		1
2	1.2	1.4	12		1
3	17.3	13.2	2		0
4	10.55	0	207		0

Let’s get started!

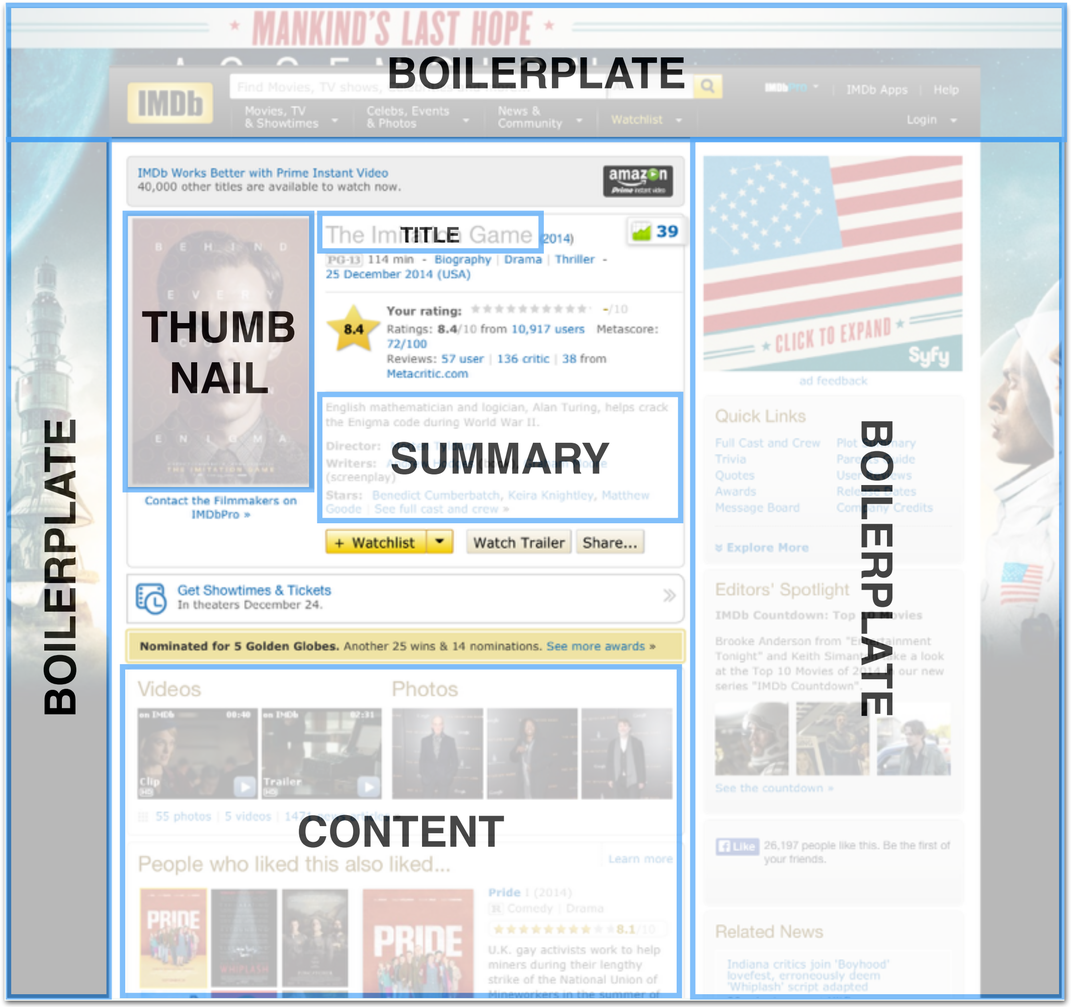
Boilerplate Detection

Document Understanding, session 2

Document Boilerplate

In a naive retrieval model, we treat all text on the page identically. This doesn't match real page content well.

- Site menus, ads, and other "boilerplate" have little bearing on the topic of the page.
- Some regions of the page, such as the title and headings, deserve extra emphasis compared to the main page content.



http://www.imdb.com/title/tt2084970

Document Zones

In order to account for different document zones, we label document text based on its zone type in the index.

In structured documents such as email, we might create a separate index for each field.

In a free-text document, we store zone information as a label for a contiguous region of the document. In HTML, this often means labeling a subtree of the DOM based on its offset within the file.

E-mail Fields

```
Date: Fri, 05 Dec 2014 14:55:05 -0800
From: "John Fluevog Shoes" <noreply@newsletter.fluevog.com>
Reply-To: noreply@newsletter.fluevog.com
X-Outgoing: echo
Subject: Fluevogian Gifts
```

HTML Token Ranges

Title

Summary

```
<h1 class="header">
    <span class="itemprop" itemprop="name">The Imitation Game</span>
    <span class="nobr">(<a href="/year/2014/?ref_=tt_ov_inf">2014</a>)</span>
</h1>
<div class="infobar">
    <span title="PG-13" class="us_pg_13 titlePageSprite absmiddle" itemprop="
    contentRating" content="PG-13"></span>
        <time itemprop="duration" datetime="PT114M"> 114 min</time>  - 
        <a href="/genre/Biography?ref_=tt_ov_inf"><span class="itemprop" itemprop="
        genre">Biography</span></a>
        <span class="ghost">|</span>
        <a href="/genre/Drama?ref_=tt_ov_inf"><span class="itemprop" itemprop="genre
        ">Drama</span></a>
        <span class="ghost">|</span>
        <a href="/genre/Thriller?ref_=tt_ov_inf"><span class="itemprop" itemprop="
        genre">Thriller</span></a>  -  <span class="nobr">
        <a href="/title/tt2084970/releaseinfo?ref_=tt_ov_inf+" title="See all
        release dates"> 25 December 2014<meta itemprop="datePublished" content="
        2014-12-25">(USA)</a>
    </span>
</div>
```

Zone Identification

Many approaches to identifying the zones of a web page have been successfully implemented.

- Rule- or template-based zone identification, for hand-tailored or automatically learned rules. May involve building a template for each major web domain (Wikipedia and IMDB need different rules).

- Render the HTML and use image processing on the rendered page to find rectangular regions of interest. Use visual cues such as font size, horizontal lines, etc. Then find the HTML code which produced the regions of interest.

- Simple heuristics based on text features also work well, and are simpler to implement.

Heuristic-based Boilerplate Detection

Kohlschütter et al (2010) developed a successful approach based on the observation that content and boilerplate have very different structural patterns, and simple heuristic features can often tell the difference.

They also provided a fast implementation which is used in many places.

Boilerplate Algorithm

1. Split an HTML document into contiguous blocks of text and `A` tags; discard other document tags.

2. Extract textual features (described next).

3. Train a machine learning classifier to label each block as `CONTENT` or `BOILERPLATE` based on the features.

Paper, data, and implementation at: http://www.l3s.de/~kohlschuetter/boilerplate/

Features for Boilerplate Detection

In contrast to prior work, they largely ignore bag-of-words and deep document structural features.

Surprisingly, they perform as well or better than methods that use these more complex features, or that use sophisticated image processing techniques.

They conclude that the majority of HTML blocks are either boilerplate “short text” blocks, or content “long text” blocks.

Feature	Discussion
Structural Tag Presence	Binary features indicating whether the block is enclosed by tags such as `H1`, `H2`, `H3`, `P`, `DIV`, or `A`.
Block Position	The absolute and relative position of the block on the page.
Text Features	Average word length, average sentence length, number of words.
Text Density	Number of words divided by number of lines
Link Density	Number of words in `A` tags divided by number of words
Heuristic Features	Number of capitalized or all-caps words, number of date/time tokens, and ratios of these to other words.

Wrapping Up

Ignoring document boilerplate text is important for improving retrieval performance. This text can easily mislead a ranker.

It's also common to weight text differently when it comes from different zones. For instance, title terms often count more than standard content terms.

This zone information can either be stored in a separate index for each field type, or with labeled document regions in a full text index.

Document Classification

Document Understanding, session 5

Classification

Classification is the Machine Learning task of assigning an object to one of a set of classes. Most tasks involving making a decision from a discrete set of options can be framed as classification.

- During indexing, is a document encoded as ASCII, UTF-8, …? Is it a web page, or some other document format?
- During crawling, is a document related to the vertical search topic? Is a URL likely to lead to spam?
- What language is a document written in?
- Is a document spam? Does it contain adult content?
- Is a document relevant to a query, or not?

Classification Formalism

Classification is a standard ML task that we'll learn more about in the next module. Here are the general steps.

- The developer selects a model (e.g. SVMs, or Logistic Regression) and a set of features to represent the objects being classified.
- A *training collection* of sample objects is built, and model parameters are chosen to perform well on the training collection.
- Once the model is trained, it can make predictions on future objects, given their features.

ML Model = Hypothesis Space
$P(Y\|X;\vartheta)$

Feature 1	…	Feature m	Label
X	…	X	Y
0.25		14	1
0.3		17	1
0		301	0
0.6		0	1

Example: Spam Classification

Suppose you want to create a classifier to decide whether a given web domain is run by spammers.

As a first step, you need to decide what information you can collect to provide evidence of spam content. This means choosing features which are predictive of spam, or of its opposite – high quality content.

- Page quality features, such as Page Rank, number of in-links, whether out-links have corresponding in-links (“edge reciprocity”).

- Content features, such as the number of pages on the domain, number of words on the home page, average page title length, etc.

- Other features…? (We will explore this more in the module on Adversarial IR)

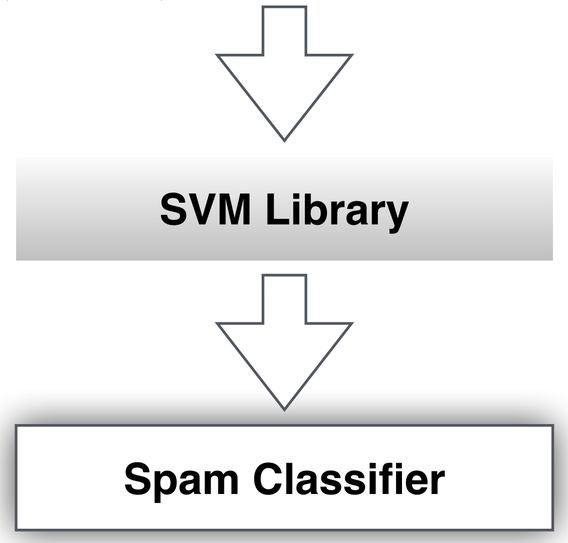
Example: Spam Classification

Once you have chosen your features, you need to prepare a data set. For instance, if you have access to a large Internet crawl you can sample a collection of spammy and non-spammy web pages.

Calculate the values of your chosen features for the selected web pages, and divide the web pages into two groups: a training set and a test set.

Train a ML model on the training set, and evaluate the model by measuring classification error on the test set.

PageRank	…	Avg. Page Title Length	Spam?
X	…	X	Y
12.5		3.6	0
13.5		5.7	1
0.35		2.3	0
0.6		10.9	1



Wrapping Up

We will spend the next module learning more of the details of classification. For now, we'll treat it as a black box.

It's straightforward to use standard ML libraries to train effective classifiers, so often the most important step is selecting the right features for your classification task.

Many of the features described in this module are also suitable for training arbitrary text classifiers for IR.

Named Entity Recognition

Document Understanding, session 8

Information Extraction

So far, we have focused mainly on ad-hoc web search. This usually starts from a user query and tries to find relevant documents.

Another possible approach is to construct a database of facts inferred from online text. This database can be used to enhance document understanding for better ranking and to answer questions more directly. This process is called **Information Extraction**.

The information panels beside search results are typically populated from these databases.

Image from bing.com; edited

Named Entity Recognition

In the IE subfield of **Named Entity Recognition** (NER), we use automated tools to identify clauses in text which correspond to particular people, places, organizations, etc.

Clauses are generally tagged with an entity type from a predefined list. Each entity type has its own contextual clues for identifying entities of that type.

For instance, times and dates often follow a few predictable formats. Peoples' names are often introduced in the surrounding text (e.g. "*spokesman* Tim Wagner").

NER Example and Tags

Citing high fuel prices, [ORG United Airlines] said [TIME Friday] it has increased fares by [MONEY $6] per round trip on flights to some cities also served by lower-cost carriers. [ORG American Airlines], a unit of [ORG AMR Corp.], immediately matched the move, spokesman [PERS Tim Wagner] said.

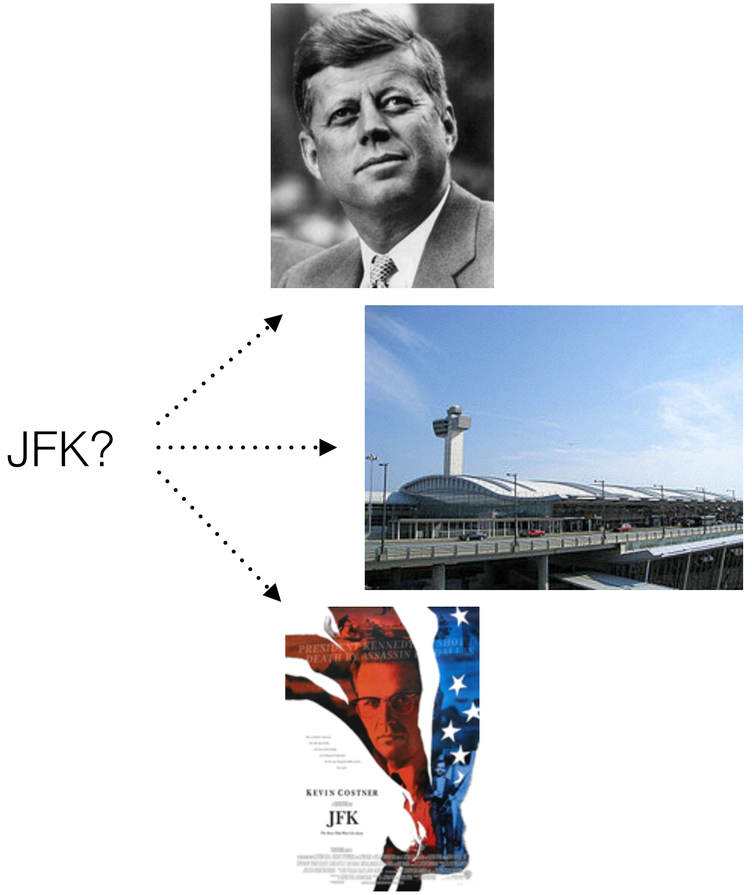
Tag	Entity	Example
PERS	People	Pres. Obama
ORG	Organization	Microsoft
LOC	Location	Adriatic Sea
GPE	Geo-political	Mumbai
FAC	Facility	Shea Stadium
VEH	Vehicles	Honda

Ambiguity in NER

NER systems often have to deal with several important types of ambiguity:

- **Reference resolution**: the same name can refer to different entities of the same type. For instance, JFK can refer to a former US president or his son.

- **Cross-type Confusion**: the identical entity mentions can refer to entities of different types. For instance, JFK also names an airport, several schools, bridges, etc.



Rule-based NER

Rule-based systems for NER are effective for certain entity classes.

Many of them use **lexicons**, which lists names, organizations, locations, etc.

Rules can also be crafted using regular expressions or other pattern matching tools. The rules may be built by hand, or with machine learning.

Entity Patterns

"<number> <word> street" for addresses

"<street address>, <city>" or "in <city>" to verify city names

"<street address>, <city>, <state>" to find new cities

"<title> <name>" to find new names

NER with Sequence Tagging

Sequence tagging is a common ML approach to NER.

Tokens are labeled as one of:

- `B`: Beginning of an entity
- `I`: Inside an entity
- `O`: Outside an entity

We train a Machine Learning model on a variety of text features to accomplish this. We'll see how to do this in the next session.

Word	Label	Tag
American	B	ORG
Airlines	I	ORG
a	O	–
unit	O	–
of	O	–
AMR	B	ORG
Corp.	I	ORG
immediately	O	–
matched	O	–
the	O	–
move	O	–
spokesman	O	–
Tim	B	PERS
Wagner	I	PERS
said	O	–

Features for Sequence Tagging

Feature Type	Explanation
Lexical Items	The token to be labeled
Stemmed Lexical Items	Stemmed version of the token
Shape	The orthographic pattern of the word (e.g. case)
Character Affixes	Character-level affixes of the target and surrounding words
Part of Speech	Part of speech of the word
Syntactic Chunk Labels	Base-phrase chunk label
Gazetteer or name list	Presence of the word in one or more named entity lists
Predictive Token(s)	Presence of predictive words in surrounding text
Bag of words/ngrams	Words and/or ngrams in the surrounding text

Word Shape

In English, the shape feature is one of the most predictive of entity names.

It is particularly useful for identifying businesses and products like Yahoo!, eBay, or iMac.

Shape is also a strong predictor of certain technical terms, such as gene names.

Shape	Example
Lower	cummings
Capitalized	Washington
All caps	IRA
Mixed case	eBay
Capitalized character with period	H.
Ends in digit	A9
Contains hyphen	H-P

NER Pipeline

A full production pipeline for NER will typically combine a few approaches:

1. First, use high-precision rules to tag unambiguous entities:
    - Use hand-tailored regular expressions, e.g. for dates and times.
    - Or write entity parsers for particular web sites, such as infoboxes on Wikipedia.
2. Search for substring matches of previously-detected names on the same page, using probabilistic string-matching metrics.
3. Consult application-specific name lists to identify likely name entity mentions from the given domain.
4. Apply **sequence tagging** using the tags from 1-3 as well as additional features, to find entities missed by the rule-based systems.

Wrapping Up

Named Entity Recognition is an important source of features for IR.

A very large fraction of queries contain named entities, so recognizing them as such and finding documents which mention the same entities is very important.

We may also want to treat the named entity as a single token, instead of as individual words (e.g. “New York Times).

Next, we’ll see how to perform NER using sequence tagging.

NER with Hidden Markov Models

Document Understanding, session 9

Context-based NER

Rule-based NER systems will inevitably miss some entities.

- All lexicons are incomplete, because new names are continually invented.

- Pattern matching doesn't work for every entity type, and is at odds with the creativity put into writing.

Statistical NER techniques instead identify entities using the terms in and around them. Today, we'll look at identifying entities using a **Hidden Markov Model**.

HMM Tagging

A **Hidden Markov Model** (HMM) is a ML model for labeling a sequence of objects based on the assumption that a given label only depends on a small number of previous items in the sequence.

Items are tagged in sequence, and use decisions made for previous items to inform the decision for the next item.

The entity labels can't be directly observed, so they are "hidden" from us.

Sentence With Tags

O	B-ETH	O	O	O	B-LOC	I-LOC
The	Phoenicians	came	from	the	Red	Sea.

Sequence Tagging

$$P(t_i | w_i = \text{“Sea”}, w_{i-1} = \text{“Red”}, t_{i-1} = \text{“B-LOC”})$$

Markov Models

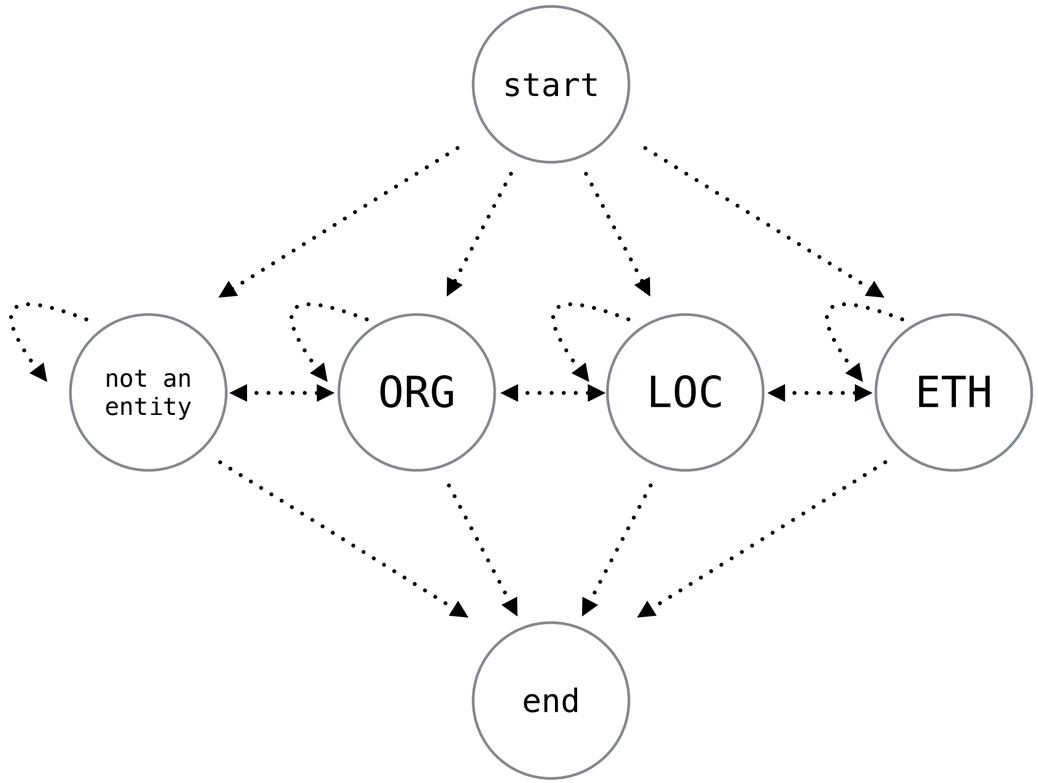
A HMM describes a process as a series of states, each with some probability distribution over the vocabulary.

When we want to assign a tag to some word w_i in a sentence, we only consider:

- The properties of w_i
- The properties and tags assigned to w_{i-1} through w_{i-k} for some small constant k

We assume that for words before w_{i-k} or after w_i have no information about the tag for w_i, mainly because this simplifies computation.

State diagram for NER tagging



* Any entity type state can transition to any other.
Some arrows omitted for clarity.

Forward-Backward Algorithm

HMMs are commonly trained using a dynamic programming technique called the **Forward-Backward algorithm**. This algorithm has three steps:

1. Forward step: Move through the sequence in increasing order, calculating $P(t_i | w_1, \ldots, w_i)$.

2. Backward step: Move backward through the sequence, calculating $P(t_i | w_{i+1}, \ldots, w_n)$.

3. Smoothing step: Smooth together the two probabilities to calculate $P(t_i | w_1, \ldots, w_n)$.

Forward Step

We need to calculate the potential of observing tag t_j given word w_j and the prior tag t_{j-1}. Denote this as:

$$\psi(t_j, t_{j-1}, j) := Pr(t_j|t_{j-1})Pr(w_j|t_j)$$

The potential of observing the whole sequence of tags up to tag m is:

$$\psi(t_1, \ldots, t_m) = \prod_{j=1}^{m} \psi(t_j, t_{j-1}, j) = Pr(w_1, \ldots, w_m, t_1, \ldots, t_m)$$

In the forward step, we compute this one position at a time:

$$\forall t \in \mathcal{T} : a(1, t) \leftarrow \psi(\langle start \rangle, t, 1)$$

$$\forall t \in \mathcal{T}, j \in \{2 \ldots m\} : a(j, t) \leftarrow \sum_{t' \in \mathcal{T}} a(j-1, t') \times \psi(t', t, j)$$

Backward Step

In the backward step, we calculate the potential of the latter portion of the sequence being preceded by the tag at position j.

The algorithm uses the same potential function ψ from the forward step:

$$\forall t \in \mathcal{T} : \beta(m, t) \leftarrow 1$$

$$\forall t \in \mathcal{T}, j \in \{1 \ldots m-1\} : \beta(j, t) \leftarrow \sum_{t' \in \mathcal{T}} \beta(j+1, t') \times \psi(t', t, j+1)$$

Smoothing Step

In the final step, we combine a and β to produce the probabilities we care about:

The normalizing constant to turn the potentials into probabilities:

$$Z := \sum_{t \in \mathcal{T}} a(m, t)$$

The potential of observing word w_j with tag a:

$$\forall j \in \{1, \ldots, m\}, t \in \mathcal{T} : \mu(j, a) \leftarrow a(j, a) \times \beta(j, a)$$
$$= \sum_{t_1, \ldots, t_m : t_j = a} \psi(t_1, \ldots, t_m)$$

The potential of transitioning from tag a to tag b given word w_j:

$$\forall j \in \{1, \ldots, m-1\}, a, b \in \mathcal{T} : \mu(j, a, b) \leftarrow a(j, a) \times \psi(a, b, j+1) \times \beta(j+1, b)$$
$$= \sum_{t_1, \ldots, t_m : t_j = a, t_{j+1} = b} \psi(t_1, \ldots, t_m)$$

Wrapping Up

Hidden Markov Models can be used for many other sequence tagging tasks: part of speech recognition, spell checking, and much more.

When used for entity recognition, they can help find entities that would be missed by rule-based systems.

The resulting entities are strong signals of page relevance, especially when the query text mentions the same entity.

Next, we'll talk about how to pull together these various types of features for IR ranking.

Features for Ranking

Document Understanding, session 10

The Importance of Features

Recall that a Machine Learning algorithm's job is to choose the most accurate function from a family of functions, as measured against some training data.

ML Model = Hypothesis Space
$P(rel = 1 \mid \mathcal{D}, \mathcal{Q}; \vartheta)$

ML techniques are very good, but aren't magic. They will only work their best if several things hold:

- If the model you choose is appropriate for the data – how good is the best function in the family?
- If the features you choose are *well correlated* with the value you're trying to predict.
- If the features provide information about *different aspects* of the value you're trying to predict.
- If you have *enough training data* to represent all the cases you're going to see in production.

Informative Features

Suppose you have already built a ranker using the following document features:

- The BM25 score against the query
- The document's PageRank
- The time the page was last updated

Which features are likely to substantially improve your ranker's performance?

Your Options:

1. A unigram language model score
2. The TF-IDF score against the query
3. The language the page is written in
4. The time the page was crawled
5. The probability the page expresses certain topics, from

Features for Ranking

Commercial web sites use hundreds of features for their rankers. Feature are generated for both documents and queries, and the document and query features are combined into a single row for the ranker.

- **Text match features** measure how well the *text* of the document and query match each other. TF-IDF, BM25, etc.
- **Topical match features** measure how well the *topics* of the document and query match each other. pLSA, LDA, etc.
- **Web graph features** are graph-theoretic properties of a page, such as PageRank, HITS, number of in-links, etc.
- **Document statistics** are basic structural information, such as the number of words in different types of tags, number of slashes in the URL, etc.

Features for Ranking

- **Document classifiers** are used to provide document categories: spam, language, news, adult content, page quality, etc.
- **Click data** gives the probability a user will click on a page if it's in the result list, or the probability of skipping it, dwell time, click count, etc.
- **External references** are supporting evidence from other sites, such as Facebook likes, Pinterest tags, etc. (Is this page trending right now?)
- **Time features** provide page freshness, date of first appearance on web, update frequency, etc.

Identifying important new features that predict relevance can make a large impact on the quality of a search engine. However, it is difficult to find new features that provide *unique* information about page relevance.

Query Features

Query features are also important for the ranking function:

- **Query statistics**, such as the number of terms, frequency of the query, and frequency of query terms.
- **Query click data**, such as click-through rate or how often users reformulate the query (and to what).
- **Result set features** are functions calculated over the top documents from some prior run of the query. This is useful, for instance, to indicate whether a query is for adult content.

Wrapping Up

Most modern search engines combine evidence from a variety of features to predict document relevance to a query.

Adding new features can dramatically improve query performance, but it's important that the features provide new information. Redundant features can confuse the ML algorithm and decrease performance.

There are likely many new families of features to be discovered for improving performance of various families of queries.

Module Wrap Up

Document Understanding, session 11

Document Understanding

Web pages contain a wealth of information about the queries they may be relevant to, if we know how to look for it.

There is rich structure to be found in HTML documents, in natural language text, and in user behavior.

http://en.wikipedia.org/wiki/Susan_Dumais

Document Understanding

When a useful set of features is identified, documents and queries can be converted into numeric vectors that provide this rich information to learning algorithms.

This allows us to leverage the best ML techniques for document ranking.

DocID	Body BM25	Title BM25	In-links	…	Rel?
1	1.23	1.2	3		1
2	1.2	1.4	12		1
3	17.3	13.2	2		0
4	10.55	0	207		0