UNIT–V
Natural Language processing

Language models

- Formal languages like Java or Python have precisely defined language models.

- A language can be defined as a set of strings;
  "print (2+2)" is a legal program in Python.
  Whereas "2) + (2 print" is not.

→ They are specified by a set of rules called grammar.

→ Formal lang also have rules that define meaning or semantics of a program.

→ P(S = words) – what is probability that a random sentence would be words.

→ Natural languages are also ambiguous.

→ Natural languages are difficult to deal with b'coz they are very large & constantly changing.

N-gram Character models

- A written text is composed of characters – letters, digits, punctuation & spaces in english.

- A model of probability distribution of n-letter sequences is called an n-gram model.

→ ~~An~~ n-gram model is defined as a Markov chain of order n−1. i.e., ~~Mo~~ Markov chain the probability of character c_i depends on immediately preceding characters, not on any other characters.

Scanned by CamScanner

So in a trigram model (Markov chain of order 2)
we have

$$P(c_i / c_{1:i-1}) = P(c_i | c_{i-2:i-1}).$$

→ - Probability of a sequence of characters $P(c_{1:N})$ under trigram model by 1st factoring with the chain rule & then using Markov assumption.

→ $$P(c_{1:N}) = \prod_{i=1}^{N} P(c_i | c_{1:i-1}) = \prod_{i=1}^{N} P(c_i | c_{i-2:i-1}).$$

→ n-gram character models are well suited in language identification.

→ One approach to lang identification is to 1st build a trigram character model of each candidate language $P(c_i / c_{i-2:i-1}, l)$. l is ranges over languages.

→ for each l the model is built by counting trigrams in a corpus of that language.
↓ body of text.

→ That gives a model of P(Text/language)

but we want to select most probable language given the text

we apply Bayes' rule followed by Markov assumption to get most probable lang.

$$l^* = \underset{l}{\operatorname{argmax}} P(l | c_{1:N})$$
$$= \underset{l}{\operatorname{argmax}} P(l) P(c_{1:N} | l) = \underset{l}{\operatorname{argmax}} P(l) \prod_{i=1}^{N} P(c_i | c_{i-2:i-1})$$

Scanned by CamScanner

Smoothing n-gram models

→ Major Complication of n-gram models is that training corpus provides only an estimation of true probability distribution.

'th' is common so we get good estimate
but 'ht' is uncommon – no dictionary words start with ht.

→ Should we assign P("_th") = 0.

→ then "Program issues an httprequest" would have an English probability of '0'.

→ So, problem in generalization: we want our language models to generalize well to texts they haven't seen yet.

→ So, we will adjust our language model so that sequences that have a count of zero in training corpus will be assigned a small non zero probability.

→ So, process of adjusting the probability of low-frequency counts is called Smoothing.

→ In lack of further information, if a random Boolean variable X has been false in all n observations so far then estimate for P(X = true) should be 1/(n+2)

i.e., he assumes with 2 more trials i.e., T or F.

→ Laplace Smoothing is a step in right direction, but performs relatively Poorly.

Scanned by CamScanner

→ Better approach is a backoff model in which we Start by estimating n-gram counts, but for any particular Sequence that has a low count, we back off to (n−1)-grams.

→ Linear interpolation Smoothing is a backoff model that combines, trigram, bigram & unigram models by linear interpolation.

→ It defines probability estimate as

$$\boxed{\hat{P}(c_i \mid c_{i-2:i-1}) = \lambda_3 P(c_i \mid c_{i-2:i-1}) + \lambda_2 P(c_i \mid c_{i-1}) + \lambda_1 P(c_i) \\ \lambda_3 + \lambda_2 + \lambda_1 = 1.}$$

→ Parameter values λ_i can be fixed, or they can be trained with an expectation-maximization algorithm.

→ $\boxed{P(c_1 \mid c_{1:0}) = P(c_1).}$

Model evaluation

- Perplexity — describing the probability of a Sequence.

$$\text{Perplexity}(c_{1:N}) = P(c_{1:N})^{-\frac{1}{N}}.$$

N-gram word models

<UNK> adding it to vocabulary.

unigram:

Bigram:

Trigram:

Scanned by CamScanner

Information Retrieval

→ Information Retrieval is the task of finding documents that are relevant to a user's need for information.

→ It can be characterized by

1. A Corpus of documents:-
Each System must decide what it wants to treat as a document: a paragraph, a page, or a multipage text.

2. Queries Posed in a Querylanguage.
   - A Query Specifies what the ~~user~~ user wants to know.
   - query language can be just a list of words. such as [AI book]; ["AI book"].

3. A Result Set :- Subset of documents that IR System judges to ~~the~~ be relevant of the query.

4. A presentation of result set:-
   - as a ranked list of document titles or as complex as a rotating color map of result set.

Boolean keyword model

- Each word in document collection is treated as a Boolean feature i.e., true of a document if word occurs in document & false if it doesnot.

- This model is simple to explain & implement.
- disadvantages. 1) degree of relevance of a document is a single bit, so there is no

IR Refinements

→ Assume that we have created an index of N documents look up TF(q_i, d_j),
guidance as to how to order the relevant documents for presentation.

2. Boolean expressions are unfamiliar to users who are not programmers or logicians.

3. It can be hard to formulate an appropriate query, even for a skilled user.

IR Scoring functions

- A Scoring function takes a document & a query & returns a numeric score.
- Most relevant documents have highest scores.
- In BM25 f^n, the score is a linear weighted combination of scores for each of words that make up query.

Three factors affect the weight of a query term:

① frequency with which a query term appears in a document.

② inverse document frequency of term, or IDF.
word 'in' appears in almost every document, so, it has a high document frequency & a low inverse document frequency.

③ Length of the document.

→ BM25 f^n takes all 3 of these into account.

→

Scanned by CamScanner

→ Assume that we have created an index of N documents in the corpus so that we can look up $TF(q_i, d_j)$, the count of no. of times word q_i appears in document d_j.

- we assume a table of document frequency counts, $DF(q_i)$, that gives no. of documents that contain the word q_i.

- Given a document d_j & a query consisting of words $q_{1:N}$, we have

$$\boxed{BM25(d_j, q_{1:N}) = \sum_{i=1}^{N} IDF(q_i) \cdot \frac{TF(q_i, d_j) \cdot (K+1)}{TF(q_i, d_j) + K \cdot \left(1 - b + b \cdot \frac{|d_j|}{L}\right)}}.$$

$|d_j|$ is length of document d_j in words.
L is average document length in corpus.

$$L = \sum_i |d_i| / N.$$

K & b can be tuned by cross validation.
$K = 2.0$ & $b = 0.75$ are typical values.

$$\boxed{IDF(q_i) = \log \frac{N - DF(q_i) + 0.5}{DF(q_i) + 0.5}}.$$

is inverse document frequency of word q_i

Rather than applying BM25 scoring f^n systems create an index ahead of time that lists, document that contain word, this is called hitlist for word.

Scanned by CamScanner

IR System evaluation

To know whether an IR System is performing well or not.

- System is given a set of queries & result sets are scored with respect to human relevance judgment.
- we use recall & precision.
- Imagine that an IR System has returned a result set for a single query ~~for which we~~

	In Resultset	Not in resultset
Relevant	30	20
Not Relevant.	10	40.

Precision:- measures the proportion of documents in resultset that are actually relevant.

Here Precision is $30/(30+10) = 0.75$.

false positive rate is $1 - 0.75 = 0.25$.

Recall:- measures the proportion of all relevant documents in collection that are in result set.

Here recall is $30/(30+20) = 0.60$.

false -ve rate is $1 - 0.60 = 0.40$.

In large document collection like www, recall is difficult to compute.

Scanned by CamScanner

IR Refinements

- Stemming
- Synonyms
- Metadata.

Page Rank Algorithms

Page Rank was invented to solve the problem of tyranny of TF scores.

Q.) if query is [IBM], home page of IBM 'ibm.com', is 1st result even if another page mentions 'IBM' more frequently?

idea is that ibm.com has many in-links, so it should be ranked higher: each inlink is a vote for quality of linked-to page.

→ if we consider only in-links, then webspammer may create a n/w of pages & ~~the~~ have them all pt to page of his choosing & increasing score of that page.

→ ∴ PageRank algorithm is designed to weight links from high-quality sites more heavily.

$$PR(P) = \frac{1-d}{N} + d \sum_i \frac{PR(in_i)}{C(in_i)}$$

$PR(P)$ → Page Rank of Page P.

N → total no. of Pages in corpus

in_i → Pages that link in to p.

$C(in_i)$ → Count of total no. of out-links on Page in_i

d is a damping factor.

Scanned by CamScanner

Q

— A.

It is

HITS Algorithm

→ Hyper link - Induced Topic Search algorithms, also known as "Hubs and Authorities" or HITS is another influential link analysis algorithm.

→ HITS differs from PageRank in several ways.

→ 1st it is a query-dependent measure: it rates pages w.r.t. to a query.

~~→ i.e, it must~~

→ HITS, first finds a set of pages that are relevant to query, by intersecting hit lists of query words & then adding pages in the link neighborhood of these pages - B.

→ Pages that link to or are linked from one of the pages in original relevant set.

→ Each page in this set is considered an authority on query to degree that other pages in relevant set point to it.

→ A page is considered a hub to degree that it points to other authoritative pages in relevant set.

→ With PageRank, we iterate a process that updates the authority score of a page to the sum of hub scores of pages that point to it,

→ hub score to be sum of authority scores of pages it points to.

→ We then normalize the scores & repeat k times ... will converge.

Scanned by CamScanner

Question Answering

- AskMSR System is a typical Web-based question-answering System. It is based on intuition that most questions will be answered many times on web.

-> So it is a problem in precision, not in recall.

-> we don't have to deal all different ways that an answer might be phrased, we only have to find one of them.

Information Extraction

- It is the process of acquiring knowledge by skimming a text and looking for occurences of a particular class of object & for relationships among objects.

- In a limited domain, it can be with high accuracy.
- As domain gets more general, more complex models & more complex learning techniques are necessary.

Here we see 6 different approaches to information extraction, in order of increasing complexity on several dimensions:
~~dime~~ deterministic to stochastic,
domain-specific to general,
hand-crafted to learned,
small-scale to large-scale.

Finite-State automata for information extraction

-> Simplest type of information extraction system is an attribute-based extraction system that assumes that the entire text refers to a single object & task is to extract attributes of that object.

Scanned by CamScanner

Extracting from text "IBM ThinkBook 970. Our price: $399.00".

Set of attributes {Manufacturer = IBM, Model = ThinkBook 970, Price = $399.00}.

→ We can address it by defining a template for each attribute.

→ template is defined by a finite state automaton, regular expression or regex.

Simplest example is attribute-based extraction systems

⇒ One step up from attribute-based extraction systems are relational extraction systems deal with multiple objects.

→ Relational based extraction system is FASTUS. which handles news stories about corporate mergers & acquisitions. It can read story.

→ Fastus consists of 5 stages:

Tokenization

Complex word handling preposition, conjuction

Basic group handling — noun groups & verb groups

Complex-phrase handling — o/p is placed in db template

Structure merging. — ~~merge~~ merges the structures

Scanned by CamScanner