natural language

n-gram character

model evaluation

natural language processing

formal languages- characterized by a set of rules called grammar

natural languages- cant be characterized as definitive set of rules

we build approximate language models instead of deriving rules from natural languages

n-gram character models

n-gram is a sequence of written symbols of length n

1-gram=unigram
2-gram=bigram
3-gram=trigram

a simple language model for written text is a probability distribution of n-grams

an n-gram character model is a model of probability distribution of n-letter sequences

an n-gram model is defined as a markov chain of order n-1

we can define the probability of a sequence of characters P(c1:N) under the trigram model by first factoring with chain rule, and then using the Markov assumption

the training corpus provides only an estimate of the true probability distribution

smoothing is the process of adjusting the probability of low-frequency counts

the backoff approa

start by estim gram co

for any par sequence th low or zero c back off to gram

So many possible models - unigram, bigram, t
different values of λ - which model should we
We can evaluate the model with cross-validati

- Step 1: Randomly split the corpus into a training corp
- Step 2: Determine the parameters of the model from t
- Step 3: Evaluate the model on the validation corpus, a
  - i.e. Calculate how correct all the predicted prob
- Step 4: Repeat step 1 to 3 a few times, say 10 times.
- Step 5: Average the 10 accuracies - this will be the ac

- First build a trigram character model of each candidate language L
  - $P(c_i \mid c_{i-2:i-1}, L)$
  - For each L, the model is built by counting the trigrams in a corpus of that language
    - This gives P(Text | Language), e.g. P("Hello World"| English)

We know P(Text | Language), how can we get the most probable language?

$$\ell^* = \underset{\ell}{\operatorname{argmax}}\ P(\ell \mid c_{1:N})$$
$$= \underset{\ell}{\operatorname{argmax}}\ P(\ell) P(c_{1:N} \mid \ell) \quad \text{Bayes' Rule: P(A | B) = P(A) * P (B | A) / P(B)}$$
$$= \underset{\ell}{\operatorname{argmax}}\ P(\ell) \prod_{i=1}^{N} P(c_i \mid c_{i-2:i-1}, \ell) \quad \text{Markov chain of order 2}$$

What happens to l* if any of $P(c_i \mid c_{i-2:i-1}, L)$ is 0?

the trigram models for each language can be learned form language corpus, but how to calculate p(l)

we may already have some estimates of p(l)

the exact number we select for these priors is not critical

- Probability estimate is given by

$$\hat{P}(c_i|c_{i-2:i-1}) = \lambda_3 P(c_i|c_{i-2:i-1}) +$$

- The parameters λ_i can be fixed or they can be tra

pageRank algorithm

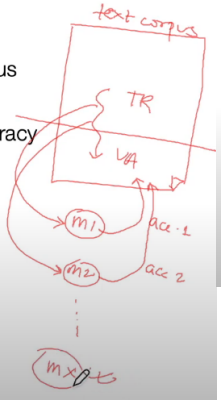
to solve the tyranny of TF scores problem, the PageRank algorithm is designed

weight links from high-quality sites more heavily

a high-quality sites is the one that is linked-to by other high-quality site

pageRank can be computed by an iterative procedure: start will all pages having PR(p)=1, and iterate the algorithm, updating ranks until they converge

rigram, interpolated smoothing with
choose?
on:
us and a validation corpus
the training data
and obtain average accuracy
abilities are.
ccuracy of the model



language modeling approach

Language Modeling approach (approach 1)

- Define one n-gram model for P(Message | Spam) by training on spam folder and one model for P(Message | Ham) by training on the inbox (basically, train two models)
- Classify a new message using Bayes' rule:
  - P(c) is estimated by simply counting total number of spam and ham messages

$$\underset{c \in \{spam, ham\}}{\operatorname{argmax}} \; P(c \mid message) = \underset{c \in \{spam, ham\}}{\operatorname{argmax}} \; P(message \mid c)\, P(c)$$

Measuring acc
- Say that the
- The sum of
- For a trigram
small numbe
- If we have a
underflow ca

we can describ
probability o
sequence us
perplexity

Machine Learning Approach (approach 2)

- Represent the message as a set of feature/value pairs and apply classification algorithm 'h' to the feature vector X
- N-grams can be input as features
  - For example, for a unigram model, each word is a feature
  - The feature value is the count of the word in the email
  - If there are 100,000 words in the language model, then the feature vector is 100,000 long
  - For a short email message almost all features will have a count zero

	Word 1	...	Word N	Spam
Email 1	2	...	8	0
Email 2	5	...	5	1

- With unigram models, the notion of the order of words is lost

f model
ch

linear interpolation smoothing

nating n-
unts

combines trigram, bigram, and unigram models using linear interpolation

rticular
at has a
ount, we
(n-1)-
s

with bigrams and trigrams the number of features is squared or cubed

it can be expensive to run algorithms on a very large feature vector, so often a process of feature selection is used

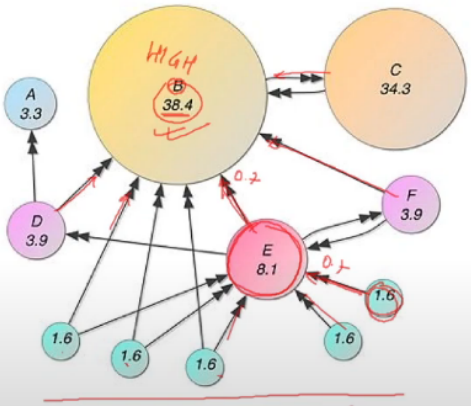
with feature selection, often large number of features can be reduced to as little as a few hundred features

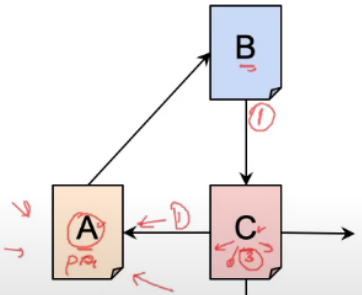
Three fact
- (1) Ter
- (2) Inve
- (3) Len

$$\lambda_2 P(c_i|c_{i-1}) + \lambda_1 P(c_i)$$
$$\lambda_3 + \lambda_2 + \lambda_1 = 1$$
ained.

Given the number of web pages N = 3, and the damping parameter
pages A, B, and C. Links between the pages are shown in the grap

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





curacy can be inconvenient

- model predicts the probability of P("abc"), P("abd"), P("abe"), etc.
- all these probabilities must be 1 (it is a multi-class classification problem)
- n model, if there are 1 million trigrams, the probability of each trigram (on average) is a
- er 1/one million
- 4-gram model, these probability values are very small numbers, and floating-point
- an become as issue

be the
of a
sing
y

Authors build a unigram, bigram, and trigram models over the words in the book and then randomly sampled sequences of words from the models

- Unigram: logical are as are confusion a may right tries agent goal the was . . .
- Bigram: systems are very similar computational approach would be represented . . .
- Trigram: planning and scheduling are integrated the success of naive bayes model is . . .
- Which sampling appears more close to AI language?

Perplexity

- Unigram model = 891
- Bigram model = 142
- Trigram model = 91

If so, why should we not build 4-gram or 5-gram word models?

IR is the task of finding documents that are relevant to a user's need for information

information retrieval scoring functions

if we have score for each document, then we can rank the documents by score and return results

scoring function takes a document and a query and returns a numeric score

tors affect the weight of each query term:

- m Frequency (TF) - the frequency with which the query term appears in a document
- For example, the documents that mention "farming" will appear more frequently
- erse Document Frequency Term (IDF)
- The word "in" appears in almost every document, so it has high document frequency, and thus ow inverse document frequency, and so it is not as important to the query as "farming" or "Kansas"
- gth of the document
- A million word document will mention all query words, but may not actually be about the query; nstead, a short document that mentions all the words is a much better candidate

r d = 0.7. Calculate the PageRank of the
h below.

$$PR(p) = \frac{1-d}{N} + d\sum_{i} \frac{PR(in_i)}{C(in_i)}$$

- PR(p) is the PageRank of page p
- N is the total number of pages in the corpus
- in_i are the pages that link in to p
- $C(in_i)$ is the count of the total number of out-links on page in_i
- d is a damping factor constant

$PR(A) = (1-0.7) + 0.7 \left\{ \frac{PR(B)}{1} \right\}$