Language Modelling

N-Grams

İçerik

- Probabilistic Models
- Probabilistic Language Models
- Markov Assumption
- Bigram model
- N-gram models
- Estimating bigram probabilities
- Evaluation and Perplexity

Bir Cümlenin Olasılığı

- “Bir cümlenin olasılığı” ???
  - P(Eve gidiyorum) >P(Evde gidiyorum)

- Eğer bir dilde bir cümle hiç yer almıyorsa ???

Dil Modeli

- Cümle (w = w1 . . . wn) olasılığı nedir?
  - Veri
  - Model
- Farklı modeller ? = farklı kestirimler
- Nasıl daha doğru olasılık değerleri kestirilebilir?

Maximum-Likelihood Estimation

- Max-Likelihood Estimation model:
  - Likelihood: $P(D|\theta)$
  - Belirgin D (data) için, θ nın farklı seçimleri farklı $P(D|\theta)$ olasılıklar ile sonuçlanır.
  - maximum likelihood: ???

Örnek

- Bozuk paranın atılması deneyi, θ = (θH, θT)
  - Gözlenen veri : D = HTTTHTHTTT
- **Model 1:** θ = (0.5, 0.5)
  - Likelihood of Model 1:

$$P(\text{HTTTHTHTTT}|\theta) = (0.5)^3 \cdot (0.5)^7 = 0.00097$$

- **Model2:** θ = (0.3, 0.7).
  - Likelihood of Model 2

$$P(\text{HTTTHTHTTT}|\theta) = (0.3)^3 \cdot (0.7)^7 = 0.00222$$

The problem with Maximum Likelihood Estimation

- MLE varsayımı:
  - Henüz oluşmamış olaylar hiçbir zaman oluşmaz.
  - (P=0)

- Bu durumu düzeltmenin yolu:
  - ???

Probabilistic Language Models

- Amaç: Bir cümleye bir olasılık değeri atayabilmek

- Machine translation (Makine çevirisi),
  - P(**high** winds tonite) > P(**large** winds tonite)
- Spell correction (Yazım Düzeltme)
  - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
- Speech Recognition (Konuşma Tanıma)
  - P(I saw a van) >> P(eyes awe of an)
- Özetleme, soru-cevaplama, …

Probabilistic Language Modeling

- **Amaç:**
  - Bir cümlede kelimelerin dizilişlerine göre bir olasılık değeri hesaplayabilmek
    - $P(W) = P(w_1,w_2,w_3,w_4,w_5...w_n)$
  - Yeni eklenecek olan bir kelimenin olasılığını hesaplayabilmek
    - $P(w_5|w_1,w_2,w_3,w_4)$

P(W) Hesaplanması

- Olasılık değeri nasıl hesaplanacak:
  - P(its, water, is, so, transparent, that)
- Estimate: P(S = w1 . . . wn)
- Lets assume: S="the cat slept quietly"
  - joint probability over the words in S:
  - P(W1 = the, W2 = cat, W3 = slept, . . . W4 = quietly).

- The joint probability, P(X, Y) = P(Y |X)P(X)
  - P(the, cat, slept, quietly) =
    P(quietly|the, cat, slept)P(the, cat, slept) =
    P(quietly|the, cat, slept)P(slept|the, cat)P(the, cat) =
    P(quietly|the, cat, slept)P(slept|the, cat)P(cat|the)P(the)

- **Varsayım: Olasılığın zincir kuralı varsayımı**

Zincir Kuralı

- Koşullu olasılık tanımı:
  - $p(B|A) = P(A,B)/P(A)$
  - $P(A,B) = P(A)P(B|A)$
  - $P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$

- Zincir Kuralı (Chain Rule):
  - $P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$

Zincir kuralı ile cümlenin olasılık değerinin hesaplanması

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_1 w_2 \ldots w_{i-1})$$

P("I want to eat Chinese food") =

P(I|Start) × P(want|I) × P(to|I want)

× P(eat|I want to) × P(Chinese|I want to eat) × P(food|I want to eat Chinese)

…

- ???

$$P(\text{so} \mid \text{I want to eat Chinese food}) = \frac{Count(\text{I want to eat Chinese food so})}{Count(\text{I want to eat Chinese food})}$$

- Problem: Bu olasılık değerini hesaplamak için yeterince veri gözlememiz olanaksızdır.

Markov Varsayımı

- Markov'un Basitlik varsayımı

$$P(\text{so} \mid \text{I want to eat Chinese food}) \approx P(\text{so} \mid \text{food})$$

$$P(\text{so} \mid \text{I want to eat Chinese food}) \approx P(\text{so} \mid \text{Chinese food})$$

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i \mid w_{i-k} \ldots w_{i-1})$$

- Yani:

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-k} \ldots w_{i-1})$$

Unigram model

- $$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i)$$

Uni-Gram	This	Is	Big	Data	AI	Book

https://devopedia.org/n-gram-model

Bigram model

-

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-1})$$

Uni-Gram	This	Is	Big	Data	AI	Book
Bi-Gram	This is	Is Big	Big Data	Data AI	AI Book	

https://devopedia.org/n-gram-model

N-gram models

- trigrams, 4-grams, 5-grams
- N değeri nasıl seçilmeli ???

  “The computer which I had just put into the machine room on the fifth floor crashed.”

- N-gram modelleri ile istenilen uzaklıkta bağımlılıklar yaratılabilir.

https://devopedia.org/n-gram-model

Bigram olasılıklarını kestirmek

- Maximum Likelihood Estimate:

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

Örnek

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

$$P(\texttt{I} \mid \texttt{<s>}) = \frac{2}{3} = .67 \qquad P(\texttt{Sam} \mid \texttt{<s>}) = \frac{1}{3} = .33 \qquad P(\texttt{am} \mid \texttt{I}) = \frac{2}{3} = .67$$

$$P(\texttt{</s>} \mid \texttt{Sam}) = \frac{1}{2} = 0.5 \qquad P(\texttt{Sam} \mid \texttt{am}) = \frac{1}{2} = .5 \qquad P(\texttt{do} \mid \texttt{I}) = \frac{1}{3} = .33$$

Örnek: Cümle olasılığının bigram kestirimi

$$
\begin{aligned}
&P(\texttt{<s> i want english food </s>}) \\
&\quad = P(\texttt{i}|\texttt{<s>})P(\texttt{want}|\texttt{i})P(\texttt{english}|\texttt{want}) \\
&\qquad\qquad P(\texttt{food}|\texttt{english})P(\texttt{</s>}|\texttt{food}) \\
&\quad = .25 \times .33 \times .0011 \times 0.5 \times 0.68 \\
&\quad = = .000031
\end{aligned}
$$

N-gram Kestirimlerinin Değerlendirilmesi

- Kurulan Dil Modeli ne kadar geçerli?
- Modelin parametreleri bir eğitim verisi ile eğitilir.
- Modelin performansı daha önceden makineye gösterilmeyen bir test verisi ile test edilir.
  - Test kümesi eğitim kümesinden farklı olmak zorundadır.
  - Modelin geçerliliği için bir değerlendirme metriğine ihtiyaç vardır.

...

- **Extrinsic evaluation**
  - İki ayrı A ve B modeli kur
    - Kaç tane yanlış yazılmış olan kelime düzeltilebildi,
    - Kaç kelime doğru şekilde çevrilebildi
  - Bu değerlendirme çok zaman harcar

- **Intrinsic evaluation:**
  - Perplexity (PP)
  - Test kümesinin kelime sayısı ile normalize edilmiş ters olasılık değeridir.

Perplexity

- Bir test kümesi W={w1,w2,…,wN} için:

$$\begin{aligned} \text{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}} \end{aligned}$$

- Zincir kuralı ile:

$$\text{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \ldots w_{i-1})}}$$

- Eğer W'nin perplexity değerini bigram modeli ile hesaplarsak:

$$\text{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_{i-1})}}$$

Perplexity

- **Düşük perplexity=Daha iyi model**

- Örnek:
  - Training 38 million words,
  - Test 1.5 million words, WallStreetJournal

	Unigram	Bigram	Trigram
Perplexity	962	170	109

- Aynı kelime uzayını kullandığı sürece PP iki ayrı dil için karşılaştırıcı kabul edilir.

Zeros

- Training set:
  … denied the allegations
  … denied the reports
  … denied the claims
  … denied the request

  P(“offer” | denied the) = 0
- Test set
  … denied the offer
  … denied the loan

Laplace smoothing (Add-one estimation)

- MLE estimate: $P_{MLE}(w_i \mid w_{i-1}) = \dfrac{c(w_{i-1}, w_i)}{c(w_{i-1})}$
- Add-1 estimate: $P_{Add-1}(w_i \mid w_{i-1}) = \dfrac{c(w_{i-1}, w_i)+1}{c(w_{i-1})+V}$

$P(w_n|w_{n-1}) = C(w_{n-1}w_n)/C(w_{n-1})$

	I	want	to	eat	Chinese	food	lunch
I	8	1087	0	13	0	0	0
want	3	0	786	0	6	8	6
to	3	0	10	860	3	0	12
eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
food	19	0	17	0	0	0	0
lunch	4	0	0	0	0	1	0

	I	want	to	eat	Chinese	food	lunch
I	.0023	.32	0	.0038	0	0	0
want	.0025	0	.65	0	.0049	.0066	.0049
to	.00092	0	.0031	.26	.00092	0	.0037
eat	0	0	.0021	0	.020	.0021	.055
Chinese	.0094	0	0	0	0	.56	.0047
food	.013	0	.011	0	0	0	0
lunch	.0087	0	0	0	0	.0022	0

$P'(w_n|w_{n-1}) = [C(w_{n-1}w_n)+1]/[C(w_{n-1})+V]$

	I	want	to	eat	Chinese	food	lunch
I	9	1088	1	14	1	1	1
want	4	1	787	1	7	9	7
to	4	1	11	861	4	1	13
eat	1	1	3	1	20	3	53
Chinese	3	1	1	1	1	121	2
food	20	1	18	1	1	1	1
lunch	5	1	1	1	1	2	1

	I	want	to	eat	Chinese	food	lunch
I	.0018	.22	.00020	.0028	.00020	.00020	.00020
want	.0014	.00035	.28	.00035	.0025	.0032	.0025
to	.00082	.00021	.0023	.18	.00082	.00021	.0027
eat	.00039	.00039	.0012	.00039	.0078	.0012	.021
Chinese	.0016	.00055	.00055	.00055	.00055	.066	.0011
food	.0064	.00032	.0058	.00032	.00032	.00032	.00032
lunch	.0024	.00048	.00048	.00048	.00048	.00096	.00048

https://slideplayer.biz.tr/slide/17466195/

Vector Space Model

- Her doküman
  - **“bag” of words** temsilindedir.
  - Dokümandaki terim sırası yada pozisyonu önemsizdir.
- Bir doküman ayrı terimlerin kümesi olarak tanımlanır.
- Terim: kelime
  - Terim bir dil sözlüğündeki kelime olmasından çok ağırlıklandırılmış bir kelimedir.

Retrieval System

- **D:** collection of documents.
- ***V** = {t1, t2, ...,t|V|} dokümanlarda geçen tüm terimler kümesi*
- ***|V|**: terim kümesi büyüklüğü, (V deki terim sayısı)*
- ***wij** > 0 her bir ti terimini dj ϵ D dokümanı ile eşleştirir.*
- dj dokümanında yer almayan bir terim için***, wij = 0.***
- Her doküman bir terim vektörü (***term vector)ile temsil edilir.***
  - **d*j = (w1j, w2j, …, w|V|j),***
- Her bir *wij terim ti **ϵ** V ve **dj ile ilişkilidir ve terimin doküman için önem derecesini verir.***
- Bu vektör gösterimi ile doküman kümesi basitçe bir relational table (yada matris) olarak gösterilir.
- Farklı Information Retrieval modellerinde, *wij farklı şekillerde hesaplanır.*

Vector Space Model

- Bu model en iyi bilinen ve en yaygın olarak kullanılan IR modelidir.
- ***Document Representation:***
  - Bu modelde bir doküman ağırlık vektörü olarak temsil edilir.
  - Her terimin ağırlıkları TF veya TF-IDF şemasının değişik versiyonları ile hesaplanır.
  - Dj dokümanındaki terim ti ‘nin ağırlığı *wij* {0, 1} aralığındadır.

Document Representation

- **Term Frequency (TF) Scheme:** dj dokümanındaki ti teriminin ağırlığı terimin dokümanda kaç kere geçtiğinin sayısıdır.(denoted by *fij)*.
- Dezavantaj: Terimler bu şekilde modele dahil edilirse ayırt edici olmayabilir.

Document Representation

Word	cf	df
try	10422	8760
insurance	10440	3997

► **Figure 6.7** Collection frequency (cf) and document frequency (df) behave differently, as in this example from the Reuters collection.

- Eğer bir terim çok sayıda dokümanda görülürse, bu terim yüksek olasılıkla çok önemli değildir yada ayırt edici değildir.
- ti teriminin **inverse document frequency** (*idfi*) değeri:

$$idf_i = \log \frac{N}{df_i}.$$

term	df_t	idf_t
car	18,165	1.65
auto	6723	2.08
insurance	19,241	1.62
best	25,235	1.5

► **Figure 6.8** Example of idf values. Here we give the idf's of terms with various frequencies in the Reuters collection of 806,791 documents.

Document Representation: **TF-IDF Weighting**

- **TF-IDF Scheme:**
  - TF: **term frequency**
  - IDF: **inverse document frequency**.
- *N: toplam doküman sayısı*
- *dfi:* ti teriminin en az 1 kere geçtiği doküman sayısı
- *fij:* ti teriminin dj dokümanındaki frekansı
- **normalized term frequency** (*tfij*) (ti teriminin dj dokümanındaki): $tf_{ij} = \frac{f_{ij}}{\max\{f_{1j}, f_{2j}, \ldots, f_{|V|j}\}}$,
- Yada Euclidean normalizasyon
- En son TF-IDF terim ağırlığı şu şekilde hesaplanır:

$$w_{ij} = tf_{ij} \times idf_i \qquad \mathrm{w}_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \times \log_{10}(N / \mathrm{df}_t)$$

- Final Ranking:

$$\mathrm{Score}(q, d) = \sum_{t \in q} \mathrm{tf\text{-}idf}_{t,d}.$$

Tf-idf Örnek

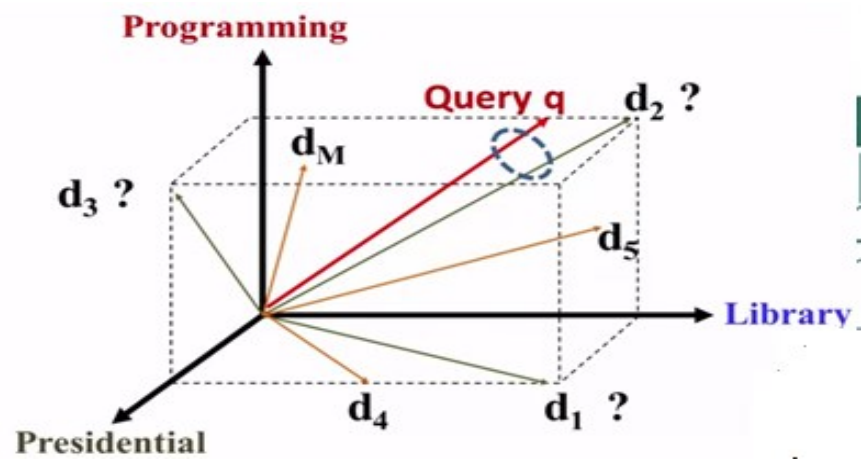
- Document: “car insurance auto insurance”
- Query: “best car insurance”

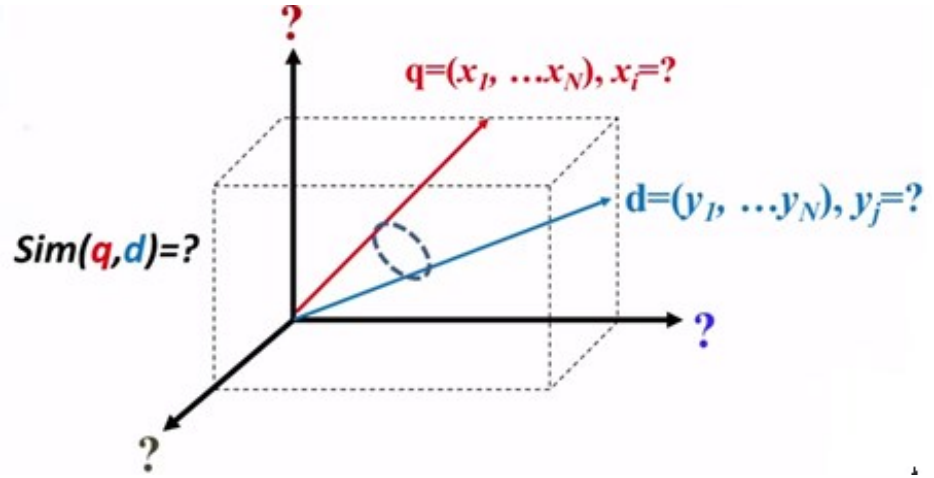
Example 6.4: We now consider the query best car insurance on a fictitious collection with $N = 1{,}000{,}000$ documents where the document frequencies of auto, best, car and insurance are respectively 5000, 50000, 10000 and 1000.

term	query				document			product
	tf	df	idf	$w_{t,q}$	tf	wf	$w_{t,d}$	
auto	0	5000	2.3	0	1	1	0.41	0
best	1	50000	1.3	1.3	0	0	0	0
car	1	10000	2.0	2.0	1	1	0.41	0.82
insurance	1	1000	3.0	3.0	2	2	0.82	2.46

In this example the weight of a term in the query is simply the idf (and zero for a term not in the query, such as auto); this is reflected in the column header $w_{t,q}$ (the entry for auto is zero because the query does not contain the termauto). For documents, we use tf weighting with no use of idf but with Euclidean normalization. The former is shown under the column headed wf, while the latter is shown under the column headed $w_{t,d}$. Invoking (6.9) now gives a net score of $0 + 0 + 0.82 + 2.46 = 3.28$.

Document Similarity: Use angle instead of distance

- 



Document Retrieval and Relevance Ranking

- **cosine similarity**: sorgu vektörü q ve doküman vektörü d*j* *arasındaki açının kosinüs değeri*,

$$cosine(\mathbf{d}_j, \mathbf{q}) = \frac{\langle \mathbf{d}_j \bullet \mathbf{q} \rangle}{\| \mathbf{d}_j \| \times \| \mathbf{q} \|} = \frac{\sum_{i=1}^{|V|} w_{ij} \times w_{iq}}{\sqrt{\sum_{i=1}^{|V|} w_{ij}^2} \times \sqrt{\sum_{i=1}^{|V|} w_{iq}^2}}.$$

- Ranking işlemi dokümanların benzerlik derecelerine göre yapılır.

Document Retrieval and Relevance Ranking

- *dj: doküman*
- *q:sorgu*
- *ti: terim*
- *fij: ti teriminin dj dokümanındaki frenkansı*
- *fiq: ti teriminin q sorgu dokümanındaki frenkansı*
- *N: toplam doküman sayısı*
- *dfi: ti terimini içeren doküman sayısı*
- *dlj: dj dokümanının uzunluğu (*in bytes)
- *avdl: doküman koleksiyonundaki ortalama doküman uzunluğu*

Document Retrieval and Relevance Ranking

- **Okapi relevance score (**of a document *dj* for a query *q* is)

$$okapi(d_j, q) = \sum_{t_i \in q, d_j} \ln \frac{N - df_i + 0.5}{df_i + 0.5} \times \frac{(k_1 + 1) f_{ij}}{k_1 (1 - b + b \frac{dl_j}{avdl}) + f_{ij}} \times \frac{(k_2 + 1) f_{iq}}{k_2 + f_{iq}},$$

- where *k*1 (between 1.0-2.0), *b* (usually 0.75) and *k*2 (between 1-1000) are parameters.

Document Retrieval and Relevance Ranking

Diğer bir yöntem:**pivoted normalization weighting (*pnw)***

$$pnw(d_j, q) = \sum_{t_i \in q, d_j} \frac{1 + \ln(1 + \ln(f_{ij}))}{(1-s) + s\frac{dl_j}{avdl}} \times f_{iq} \times \ln\frac{N+1}{df_i}$$

Burada s sabit bir parametredir ve genellikle 0.2 seçilir. Uygulamaya göre farklılık gösterebilir.

Evaluating IR Systems

- IR ve Web search modellerinde dokümanın ilgili yada ilgisiz olması gibi bir sonuca ulaşılmaz. Bunun yerine dokümanlar arasında ilgiliden ilgisize bir ranking skorlaması yapılır ve kullanıcıya sunulur.
- D:doküman veritabanı, toplam doküman sayısı:N, q:kullanıcı sorgusu
- Retrieval algoritması öncelikle D'deki tüm dokümanlar için uygunluk (relevance) skorları hesaplar. Ranking vektörü kurulur:

- Vektördeki her doküman için precision ve recall değerleri hesaplayabiliriz.

$$R_q : \ < \mathbf{d}_1^q, \mathbf{d}_2^q, \ldots, \mathbf{d}_N^q >,$$

Evaluation Measures

- **Recall** at rank position i or document $\mathbf{d}_i^q$ (denoted by $r(i)$) is the fraction of relevant documents from $\mathbf{d}_1^q$ to $\mathbf{d}_i^q$ in R_q. Let the number of relevant documents from $\mathbf{d}_1^q$ to $\mathbf{d}_i^q$ in R_q be s_i ($\leq |D_q|$) ($|D_q|$ is the size of D_q). Then,

$$r(i) = \frac{s_i}{|D_q|}. \tag{17}$$

Precision at rank position i or document $\mathbf{d}_i^q$ (denoted by $p(i)$) is the fraction of documents from $\mathbf{d}_1^q$ to $\mathbf{d}_i^q$ in R_q that are relevant:

$$p(i) = \frac{s_i}{i} \tag{18}$$

Example

- *D: 20 doküman*
- q:sorgu, biliyoruz ki 8 doküman q sorgusu ile uyumlu (relevant)
- Bir retrieval algoritması sıralamayı(ranking) çıkarır.
  - 1: highest rank, 20: lowest rank.
- "+" :relevant document
- "-": irrelevant document
- Kolon 3: precision ($p(i)$)
- Kolon 4: recall ($r(i)$) değerlerini verir.

Rank i	+/–	$p(i)$	$r(i)$
1	+	1/1 = 100%	1/8 = 13%
2	+	2/2 = 100%	2/8 = 25%
3	+	3/3 = 100%	3/8 = 38%
4	–	3/4 = 75%	3/8 = 38%
5	+	4/5 = 80%	4/8 = 50%
6	–	4/6 = 67%	4/8 = 50%
7	+	5/7 = 71%	5/8 = 63%
8	–	5/8 = 63%	5/8 = 63%
9	+	6/9 = 67%	6/8 = 75%
10	+	7/10 = 70%	7/8 = 88%
11	–	7/11 = 63%	7/8 = 88%
12	–	7/12 = 58%	7/8 = 88%
13	+	8/13 = 62%	8/8 = 100%
14	–	8/14 = 57%	8/8 = 100%
15	–	8/15 = 53%	8/8 = 100%
16	–	8/16 = 50%	8/8 = 100%
17	–	8/17 = 53%	8/8 = 100%
18	–	8/18 = 44%	8/8 = 100%
19	–	8/19 = 42%	8/8 = 100%
20	–	8/20 = 40%	8/8 = 100%

Evaluation Measures

- **Average Precision:** Sometimes we want a single precision to compare different retrieval algorithms on a query **q**. An average precision (*p*avg) can be computed based on the precision at each relevant document in the ranking,

$$p_{avg} = \frac{\sum_{d_i^q \in D_q} p(i)}{|D_q|}.$$

$$p_{avg} = \frac{100\% + 100\% + 100\% + 80\% + 71\% + 67\% + 70\% + 62\%}{8} = 81\%.$$

Evaluation Measures

- **Comparing Different Algorithms:** farklı algoritmalara ait sonuçları karşılaştırmak için precision ve recall eğrilerini aynı figürde göstermek iyi bir yoldur

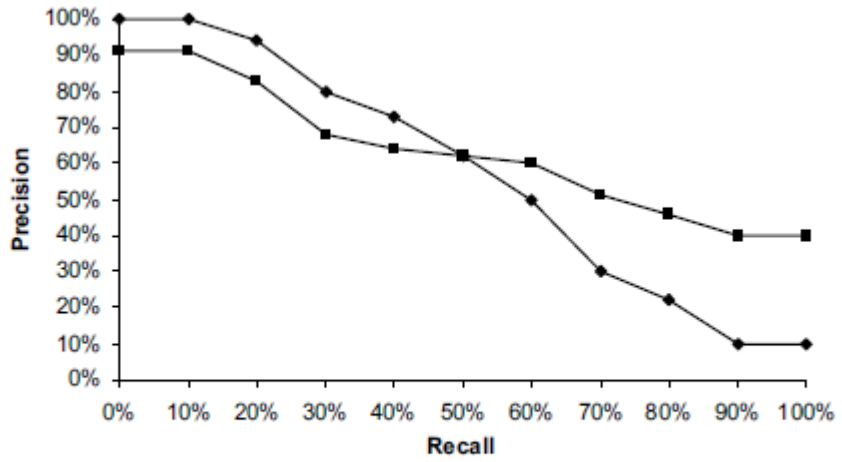


Fig. 6.5. Comparison of two retrieval algorithms based on their precision-recall curves

Evaluation Measures

- **Rank Precision: Örneğin bir** Web arama motoru için genellikle top 5, 10, 15, 20, 25 ve 30 adet dönen sayfa için precision değeri hesaplanır. Önceki örneğe göre: $p(5) = 80\%$, $p(10) = 70\%$, $p(15) = 53\%$, ve $p(20) = 40\%$.
- Search ranking'i değerlerindirmek için precision tek ölçüt değildir. Top ranked sayfaların niteliği de çok önemlidir.
- **F-score:** bir diğer sık kullanılan ölçüttür. Her i rank pozisyonuna ait F-score değerini hesaplayabiliriz.
- F-score: precision ve recall değerlerinin harmonik ortalamasıdır.

$$F(i) = \frac{2}{\frac{1}{r(i)} + \frac{1}{p(i)}} = \frac{2\,p(i)r(i)}{p(i) + r(i)}$$

Referanslar

- An Empirical Study of Smoothing Techniques for Language Modeling, Chen and Goodman, TR-10-98, 1998 (*lots of technical and empirical details*)
- https://lagunita.stanford.edu/c4x/Engineering/CS-224N/asset/slp4.pdf
- http://www.cstr.ed.ac.uk/downloads/publications/2000/gutkin_mphil.pdf