Improving Search Engines using Multi-Word Indicies

Hatem Nassrat
CSCI 6403
December 2008

Introduction

- Multi-Word Index
- To Find
  - Advantages (Better Accuracy)
  - Disadvantages (Speed, Disk Space)

Multi Words

Memory

M = 2
N = 4

Database

Multi Words

Memory

~~AA (0,1)~~
AB (0,2)
AD (0,3)

M = 2
N = 4

Database

Multi Words

Memory

AB (0,2)
AD (0,3)

M = 2
N = 4

Database

Multi Words

M = 2
N = 4

Memory

AB (0,2)
AD (0,3)
AB (1,2)
AD (1,3)
AE (1,4)

Database

Multi Words

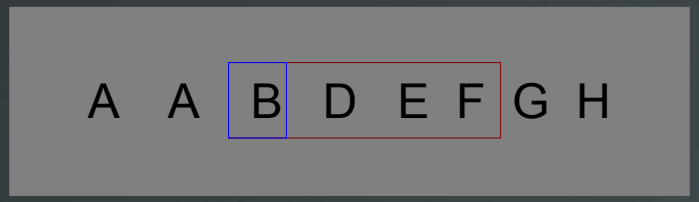
M = 2
N = 4

Memory

~~AB (0,2)~~
~~AD (0,3)~~
AB (1,2)
AD (1,3)
AE (1,4)

Database

Multi Words



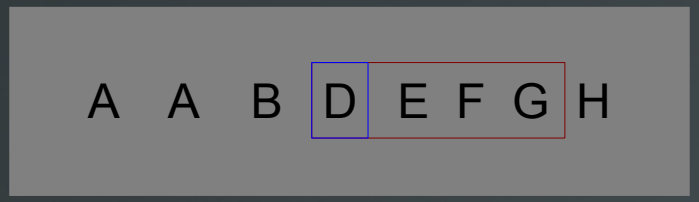
M = 2
N = 4

Memory

AB (1,2)
AD (1,3)
AE (1,4)
BD (2,3)
BE (2,4)
BF (2,5)

Database

Multi Words



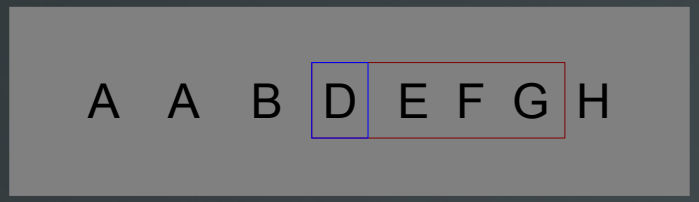
M = 2
N = 4

Memory

AB (1,2)
AD (1,3)
AE (1,4)
BD (2,3)
BE (2,4)
BF (2,5)

Database

Multi Words



M = 2
N = 4

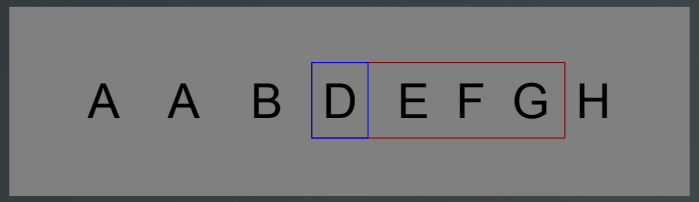
Memory

AD (1,3)
AE (1,4)
BD (2,3)
BE (2,4)
BF (2,5)

Database

AB (1,2)

Multi Words



M = 2
N = 4

Memory

AD (1,3)
AE (1,4)
BD (2,3)
BE (2,4)
BF (2,5)
DE (3,4)
DF (3,5)
DG (3,6)

Database

AB (1,2)

Multi Words

M = 2
N = 4

Memory

AE (1,4)
BE (2,4)
BF (2,5)
DE (3,4)
DF (3,5)
DG (3,6)

Database

AB (1,2)
AD (1,3)
BD (2,3)

Method

- Reuters 21578
- Base + Augmented System
- Inverted Files (TC B-Tree)
- Application in Vector Space
- Retreival – TFIDF
- MW: using M=4, N=10

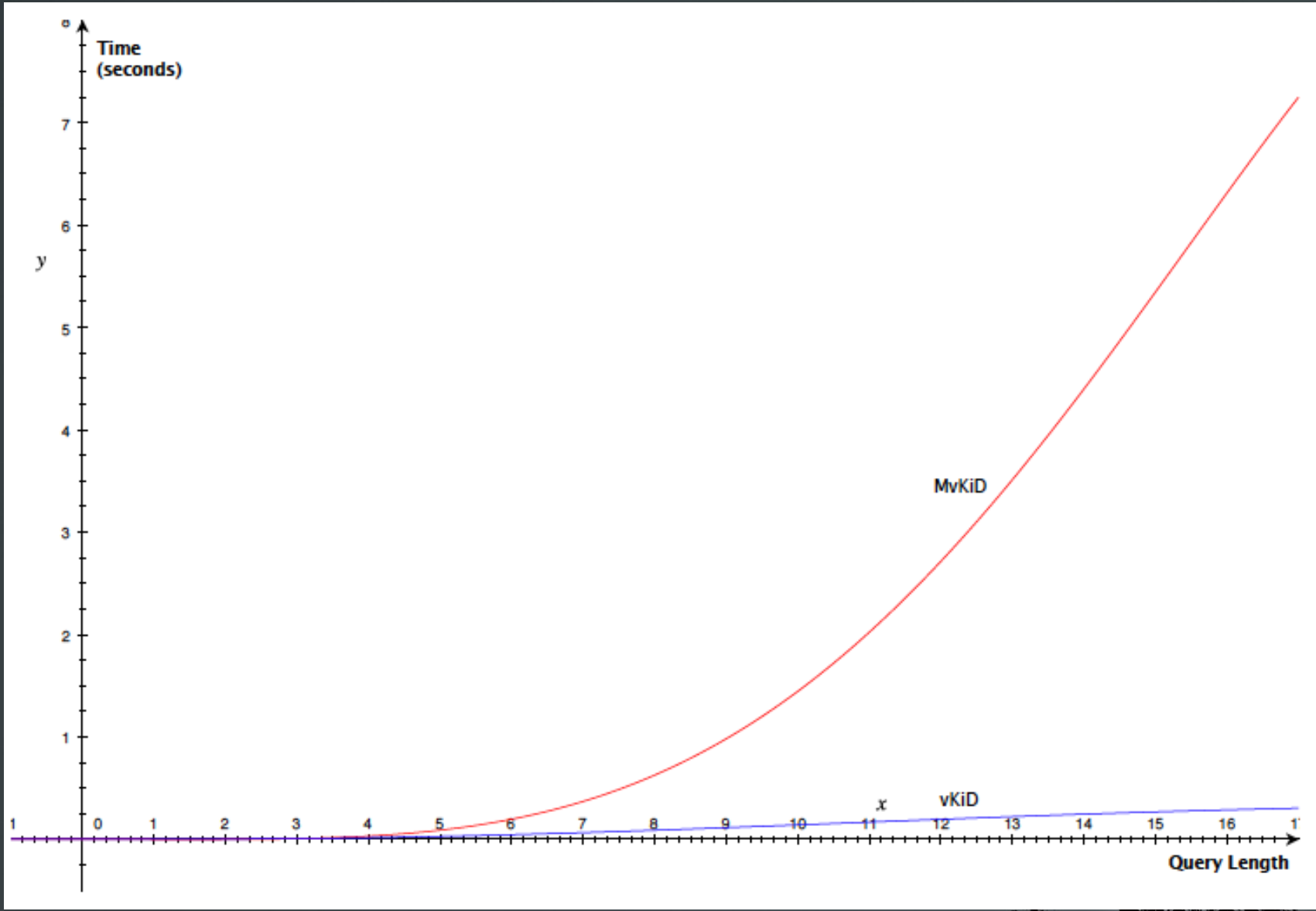
$$1 + \sum_{i=1}^{M} \binom{N-1}{i}$$

$$(1 + \sum_{i=1}^{M} \binom{N-1}{i}) \times (X + Y)$$

Results - Size

- Single
  - 38,067 entries
  - 9.7 MB
- Multi-Word
  - 15,178,734 entries
  - 38,067 , 2,615,008 , 8,726,517 , 3,799,142
  - 281 MB

Query Speed



Evaluation

crude china	coffee colombia	interest uk
earn italy	grain china	acq uk
cocoa usa	yen japan	corn usa
rapeseed japan	carcass usa	interest usa
oilseed china	wheat ussr	trade japan

Precision & Recall

Small pre-tagged result set					
Query	P@1	P@10	P@50	MAP	Recall
china crude	0.0%	20.0%	4.0%	8.7%	100.0%
china oilseed	0.0%	10.0%	4.0%	7.0%	100.0%
cocoa usa	0.0%	0.0%	4.0%	4.4%	90.0%
earn italy	0.0%	10.0%	2.0%	2.6%	62.5%
japan rapeseed	100.0%	60.0%	12.0%	41.6%	100.0%
Medium pre-tagged result set					
Query	P@1	P@10	P@50	MAP	Recall
carcass usa	0.0%	0.0%	0.0%	0.1%	4.5%
china grain	100.0%	90.0%	36.0%	40.5%	100.0%
coffee colombia	100.0%	100.0%	62.0%	91.0%	100.0%
japan yen	0.0%	20.0%	6.0%	9.7%	100.0%
ussr wheat	100.0%	70.0%	50.0%	51.0%	100.0%
Large pre-tagged result set					
Query	P@1	P@10	P@50	MAP	Recall
acq uk	0.0%	0.0%	8.0%	1.7%	28.9%
corn usa	0.0%	0.0%	2.0%	31.4%	94.2%
interest uk	0.0%	20.0%	18.0%	14.3%	94.1%
interest usa	0.0%	0.0%	0.0%	0.8%	37.7%
japan trade	100.0%	50.0%	26.0%	29.5%	98.8%

Table 1: Precision and Recall for the base system

Precision & Recall

Small pre-tagged result set					
Query	P@1	P@10	P@50	MAP	Recall
china crude	0.0%	20.0%	4.0%	8.9%	100.0%
china oilseed	0.0%	10.0%	4.0%	7.8%	100.0%
cocoa usa	0.0%	0.0%	2.0%	4.0%	90.0%
earn italy	0.0%	10.0%	2.0%	2.7%	62.5%
japan rapeseed	100.0%	60.0%	12.0%	52.5%	100.0%
Medium pre-tagged result set					
Query	P@1	P@10	P@50	MAP	Recall
carcass usa	0.0%	0.0%	2.0%	0.1%	4.5%
china grain	100.0%	90.0%	36.0%	38.8%	100.0%
coffee colombia	100.0%	90.0%	62.0%	84.3%	100.0%
japan yen	0.0%	10.0%	12.0%	15.1%	100.0%
ussr wheat	100.0%	90.0%	48.0%	53.7%	100.0%
Large pre-tagged result set					
Query	P@1	P@10	P@50	MAP	Recall
acq uk	0.0%	0.0%	8.0%	1.7%	28.9%
corn usa	100.0%	60.0%	40.0%	39.7%	94.2%
interest uk	0.0%	30.0%	26.0%	17.1%	94.1%
interest usa	0.0%	0.0%	0.0%	0.8%	37.7%
japan trade	100.0%	40.0%	40.0%	38.0%	98.8%

Table 2: Precision and Recall for the Phrase system

Precision & Recall

Avg(Multi-Word) - Avg(Single-Word)

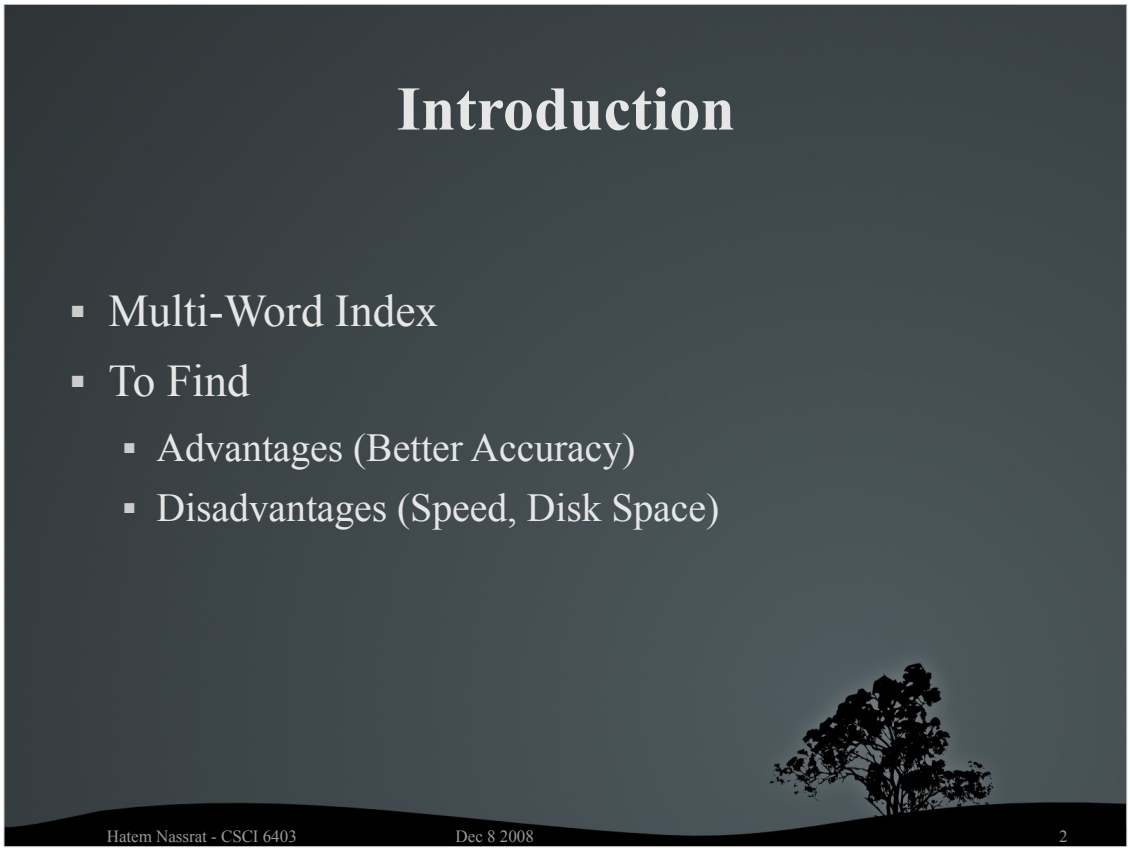
map	p@1	p@10	p@50	recall
Small pre-tagged result set				
2.3%	0.0%	0.0%	-0.4%	0.0%
Medium pre-tagged result set				
-0.1%	0.0%	0.0%	1.2%	0.0%
Larger pre-tagged result set				
3.9%	20.0%	12.0%	12.0%	0.0%

Table 3: Average Precision and Recall Increase

Questions ?

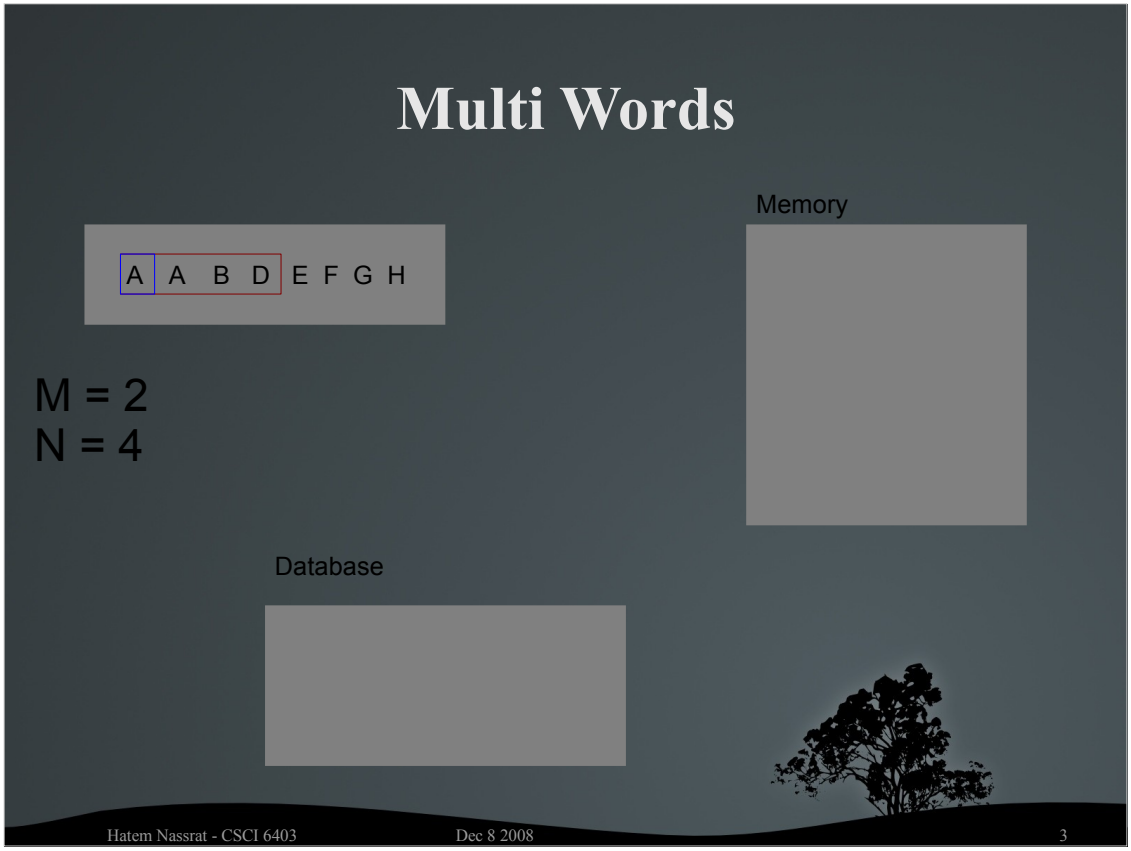
Improving Search Engines using Multi-Word Indicies

Hatem Nassrat
CSCI 6403
December 2008



- Reasearch Q: Can we improve Search Engines
- Attempt extending the traditional single word indices with Multi-Word Indices

- The aim here is to gain a more accurate SEngine
- We also aim to reduce the apparent disadvantages
  - Slower Speeds
  - Higher Disk Usage due to larger indices



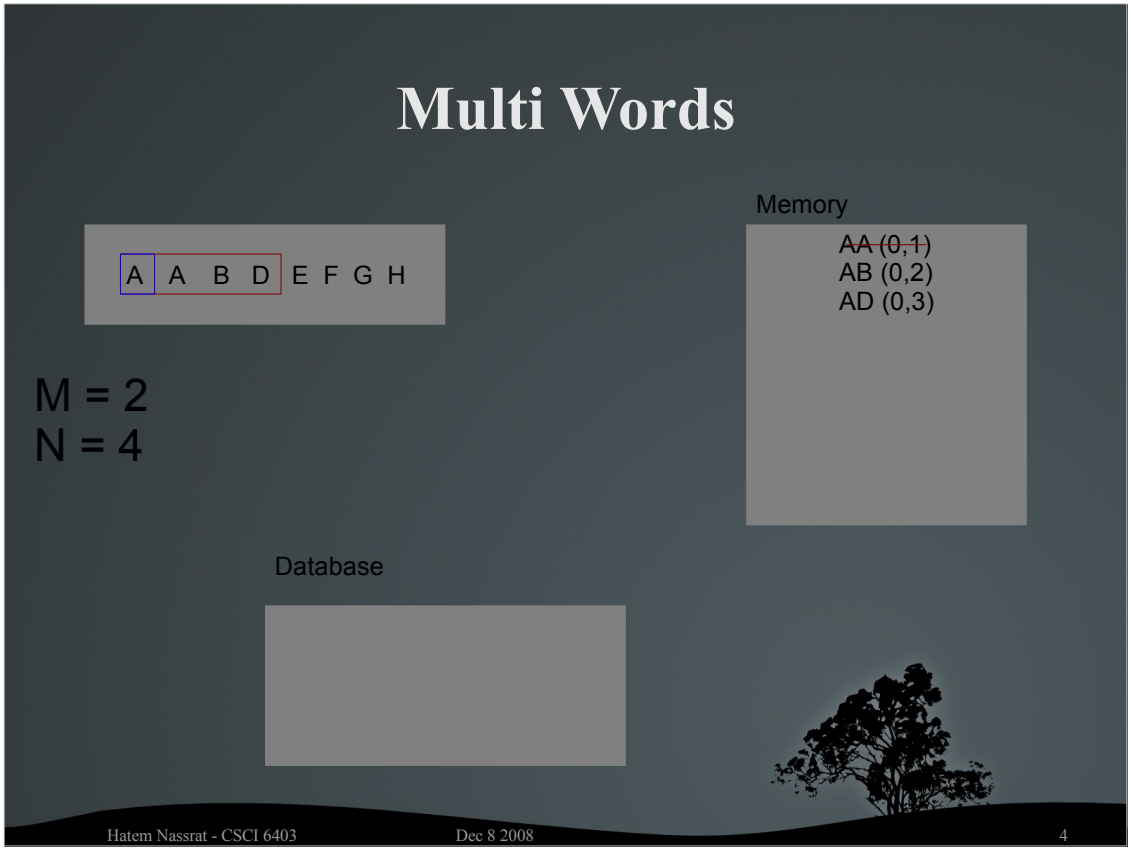
There are many little tricks used to generate the multi-words and avoid repitition. With this example I will try to show some of them.

In this example, we:

Generating Multi-Words of length M
Window of Size N

In this example M = 2, N = 4

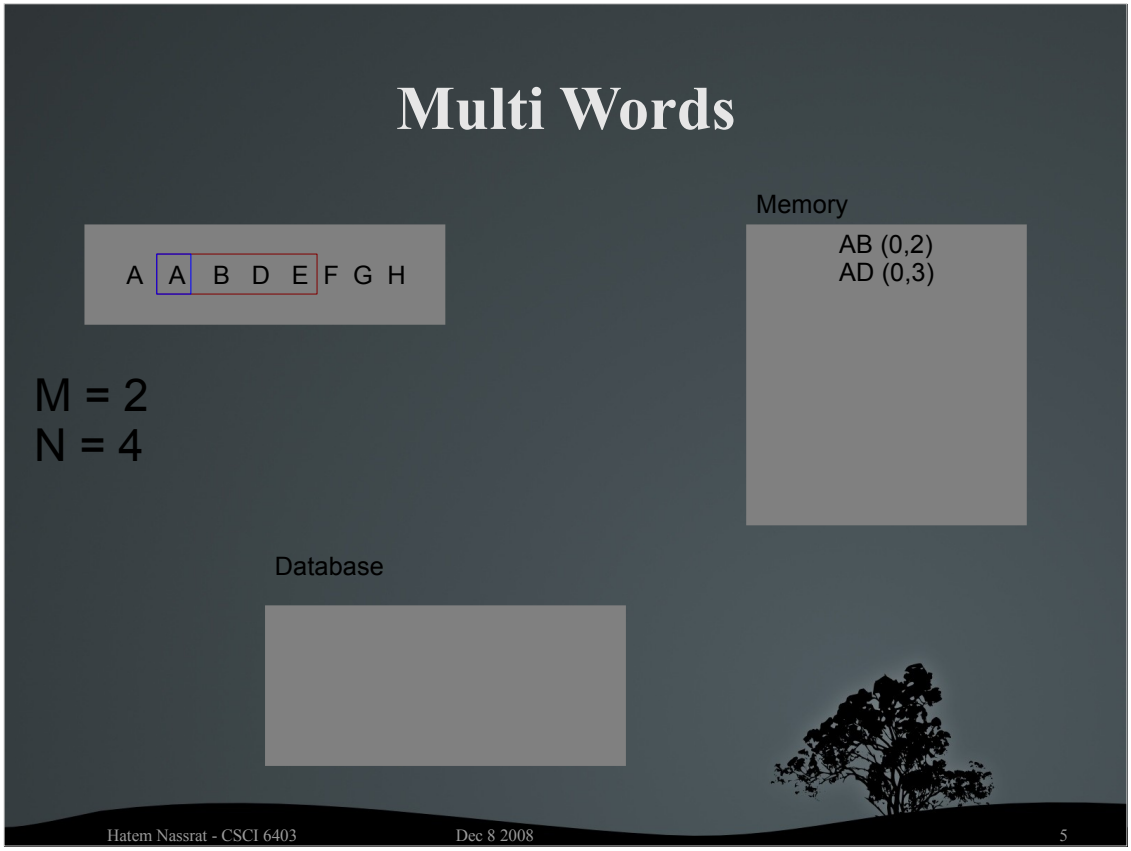
Using the first word, A, in the window we generate all the 2 length Multi-Words.



AA, AB, AD are created and are kept in memory. We also store the indices of the words that make up each Mult-Word.

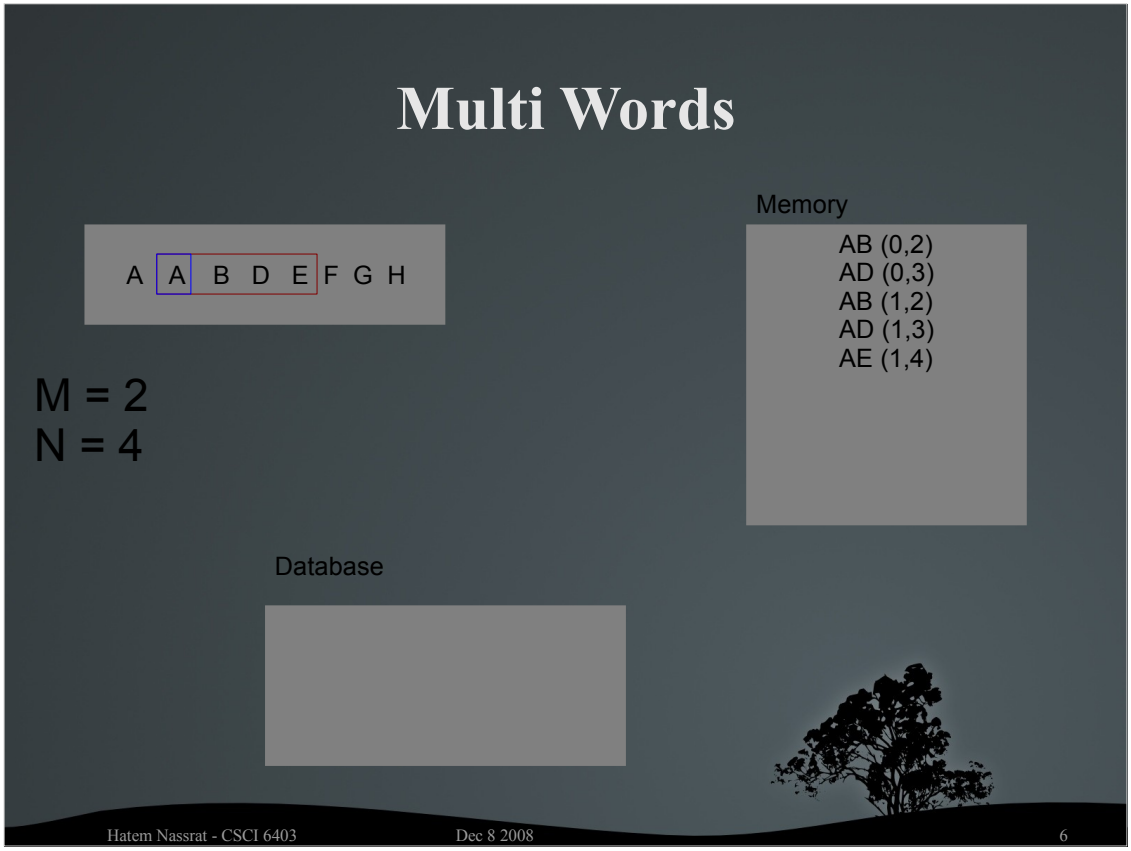
Since the word A was combined with the same word A, that Multi-Word instance is ignored.

The reason we keep them in memory will be aparent shortly



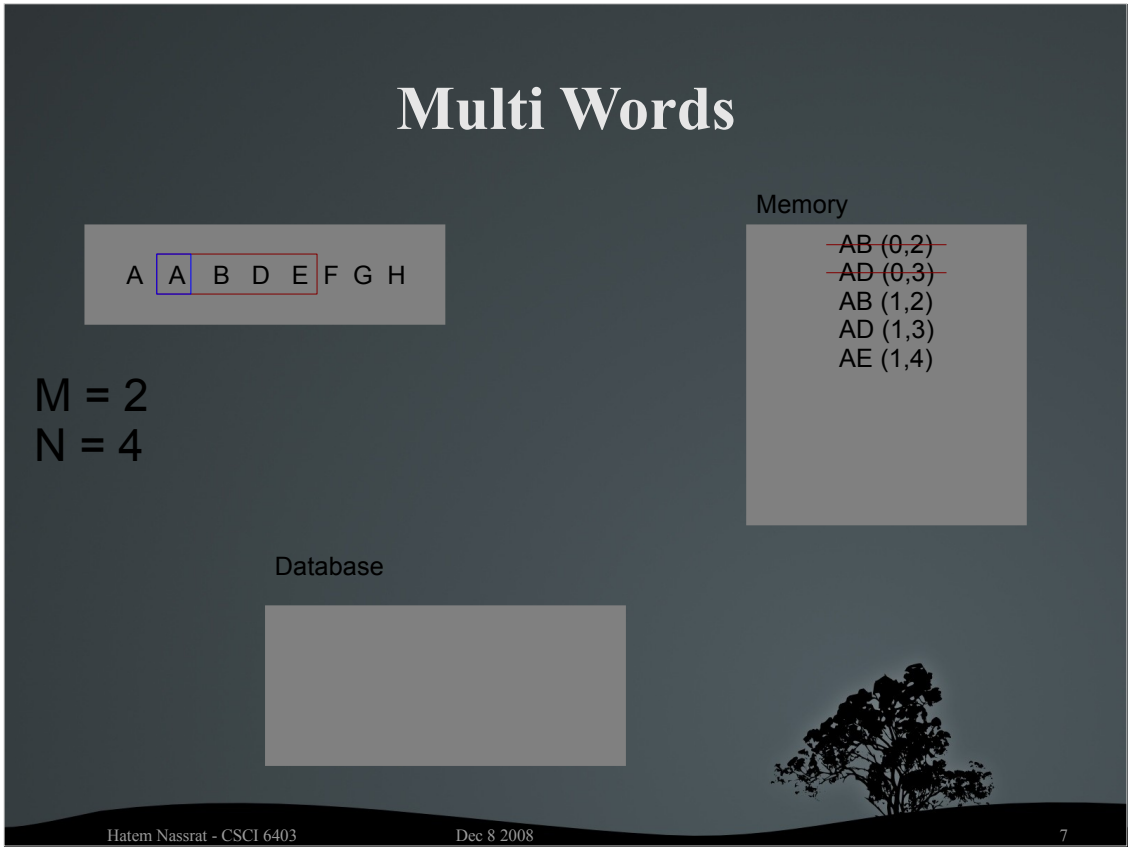
We then move the Window, thus adding word E to the window and removing the first word A.

We then generate the Multi-Words from the window elements.



Getting AB, AD, AE

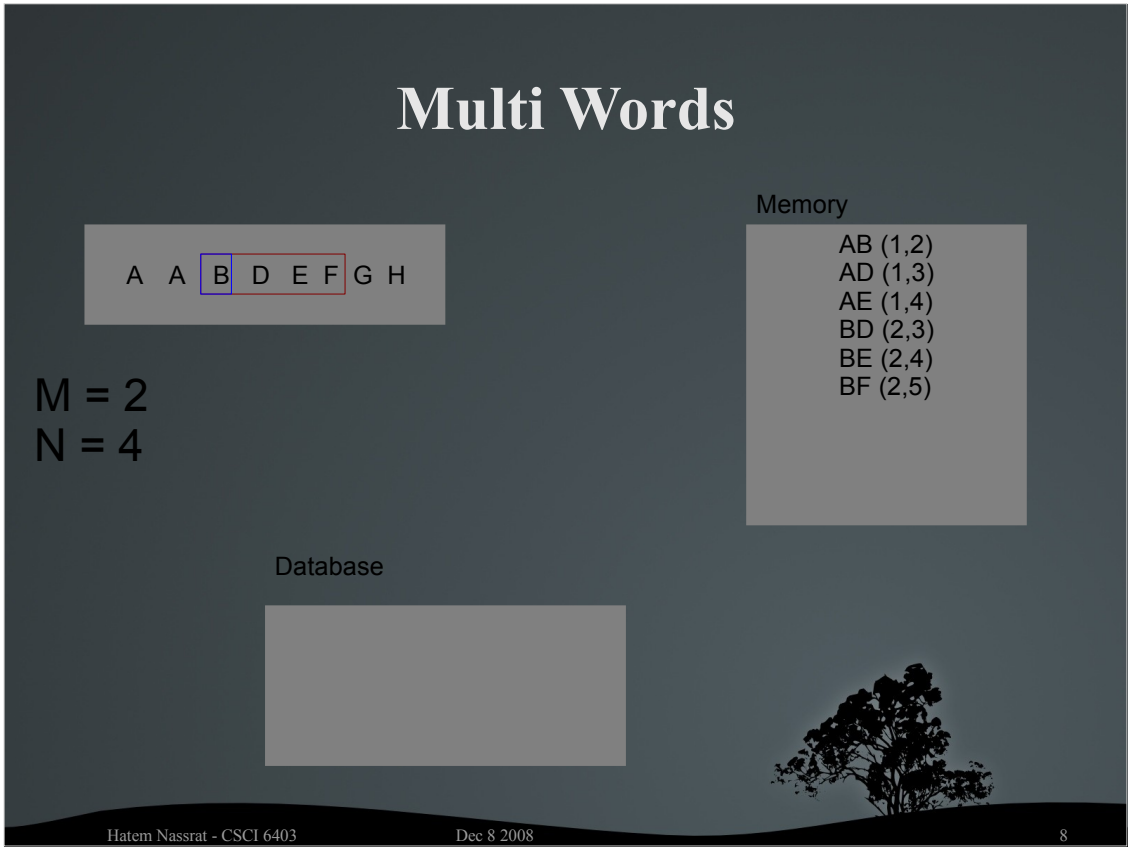
Since AB abd AD were already available in the memory datastructure, we have to do some pruning.



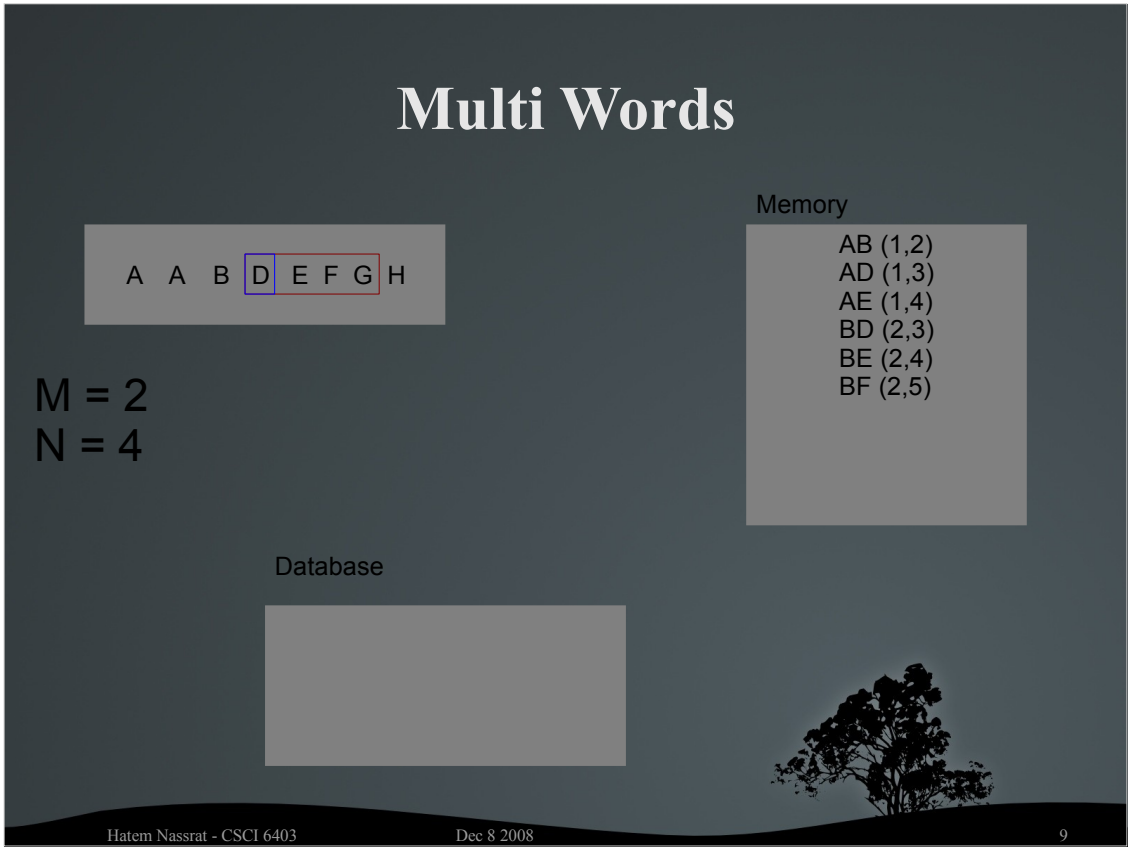
Which removes the First instance AB and AD.

The reasoning here is made by looking at the disnaces between the words.
For example the first instance AB appeared with a distance of 2, While he second had an index distance of 1.

We then preoceed to move the window and generate words.



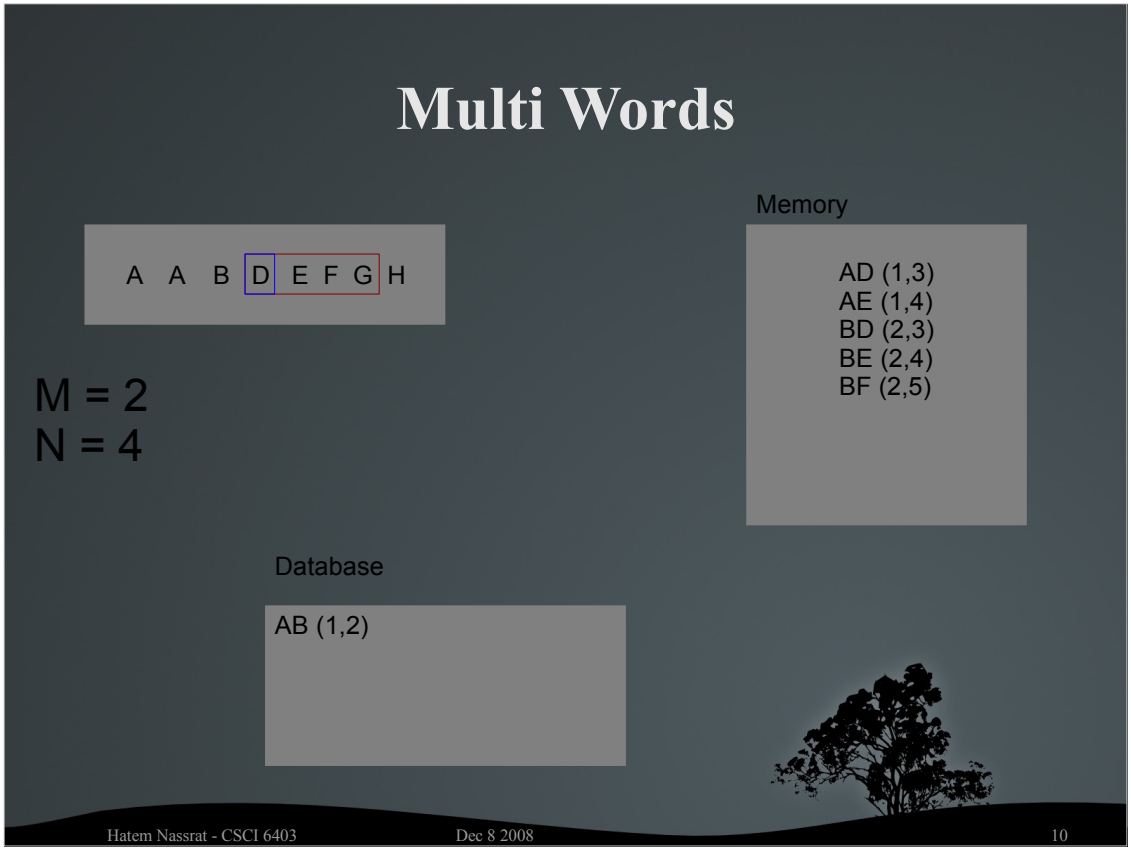
And we move the window again ...



Well now we see that the first words A, A, B have completely left the window.

From the in memory datastructure we notice that a subset of those words joined to make a full Multi-word. Namely the first instance AB.

So that gets saved as a full Multi-Keyword.

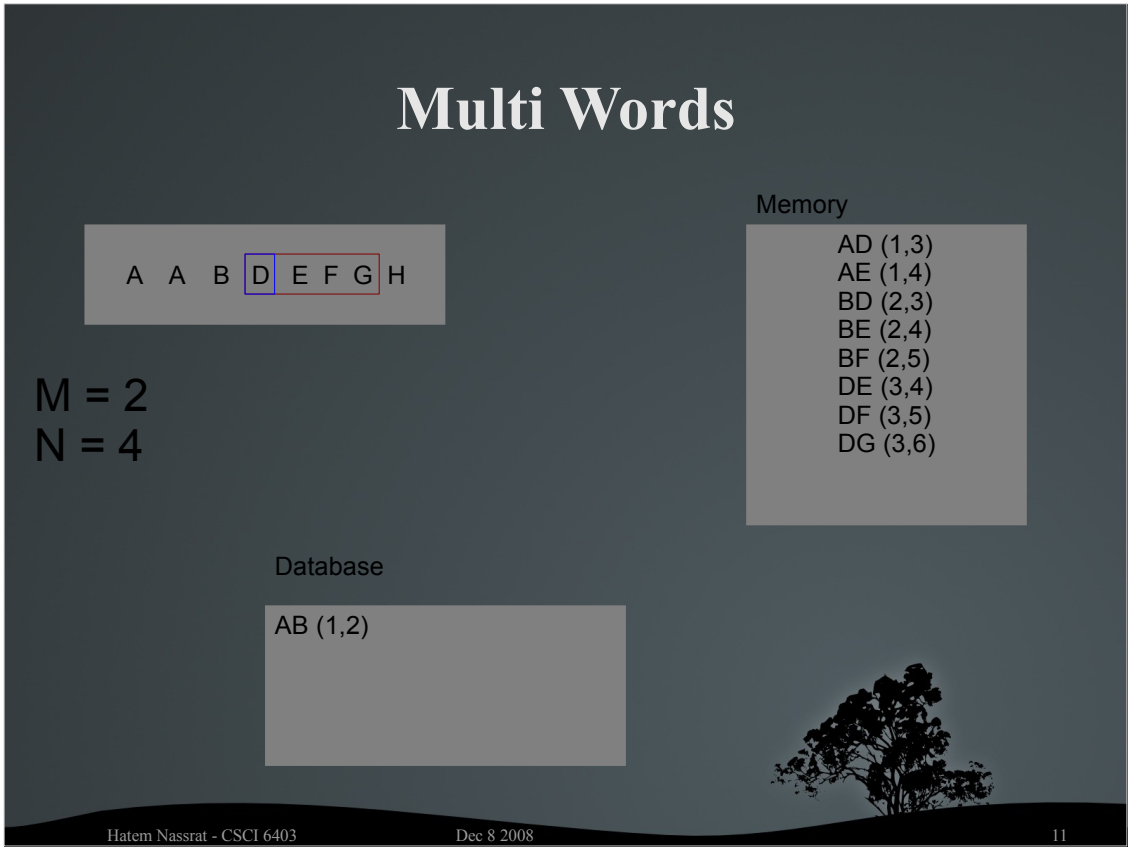


Well now we see that the first words A, A, B have completely left the window.

From the in memory datastructure we notice that a subset of those words joined to make a full Multi-word. Namely the first instance AB.

So that gets saved into the forward Index for this document.

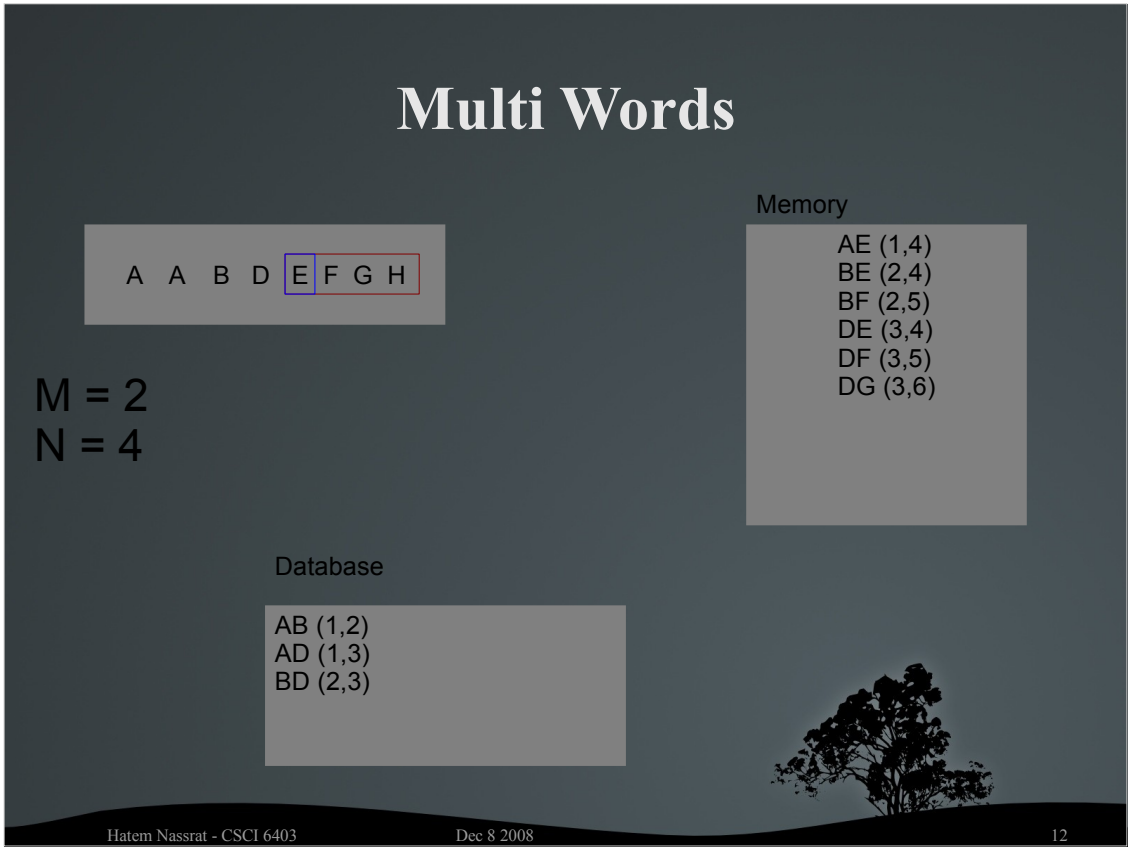
Moving the window again …



Well now we see that the first words A, A, B have completely left the window.

From the in memory datastructure we notice that a subset of those words joined to make a full Multi-word. Namely the first instance AB.

So that gets saved as a full Multi-Keyword.

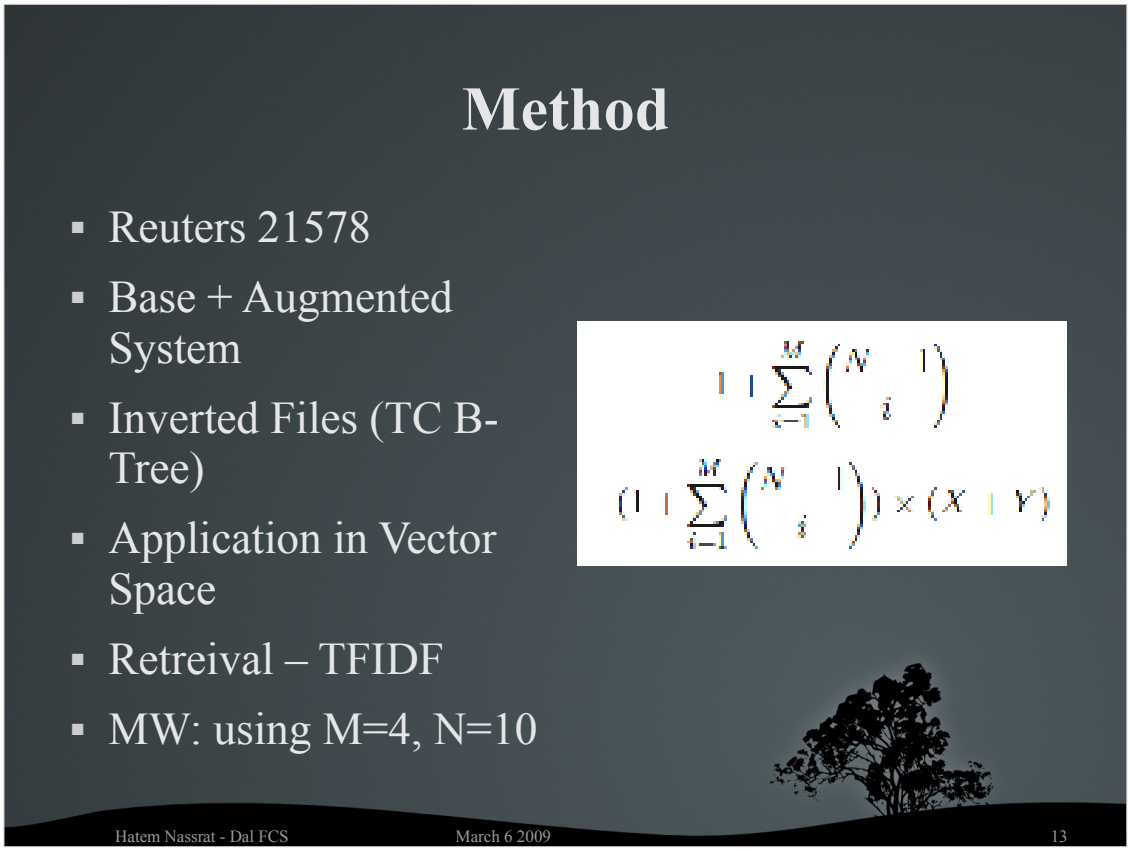


Will flush the next instances AD, and BD

And so on untill all windows of words in the document have been processed.

...

So thats basically what these Multi-Words are.



To aid in our research, a prototype Vector Space Search Engine was implemented that utilized the Reuters 21578 dataset.

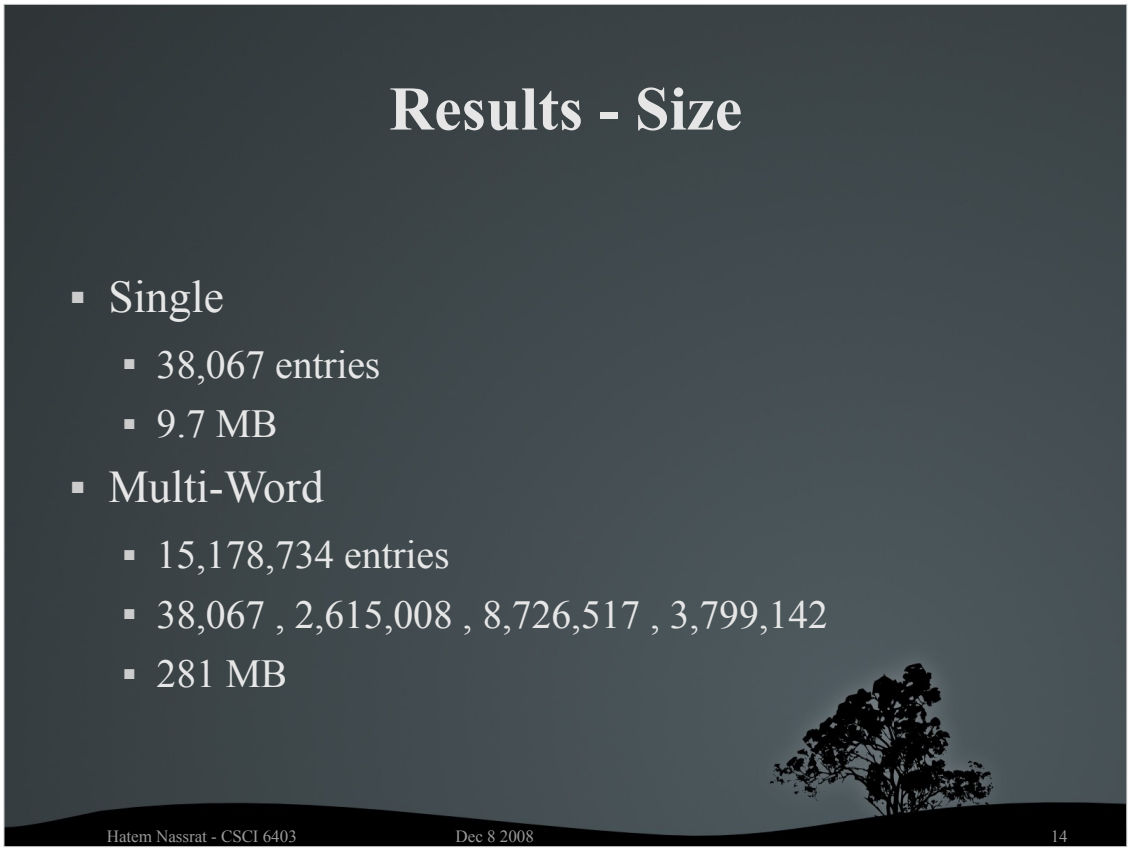
Two similar systems were created, the major difference being the indexing code for each. The first implements a traditional single word system, while the second implements the new Multi-word system.

TokyoCabinet was used for the Indices as it is the fastest available DBM clone. Similar to Berkley DB it had the B-Tree backend option which was utilized for both indices.

The Ranking function in both situations was the TFIDF.
M = 4 and N = 10

To formulize the worst case for the number of Multi-words of size M, per Window of size N. (Size 1, Size, 2 … M)

The second equation shows the calculation for a Document containing X words in the Title and Y words in the body.

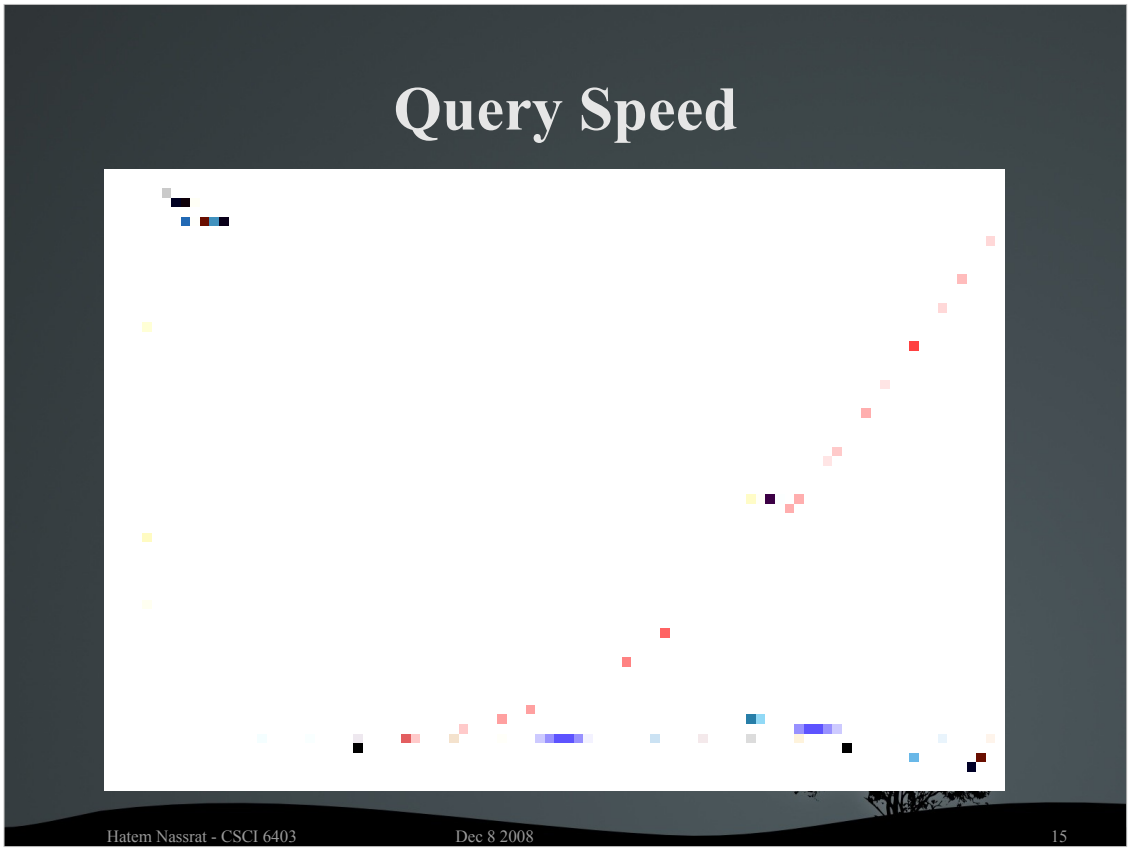


After generating both Indices, we noticed that the Single Keyword Index Contained 38K+ entries and was 9.7MB in size.

The Multi_word Index contained over 15M+ entries
The distributions was 38K Single, 2.6M Double, 8.7M Triple and 3.8M Quad.

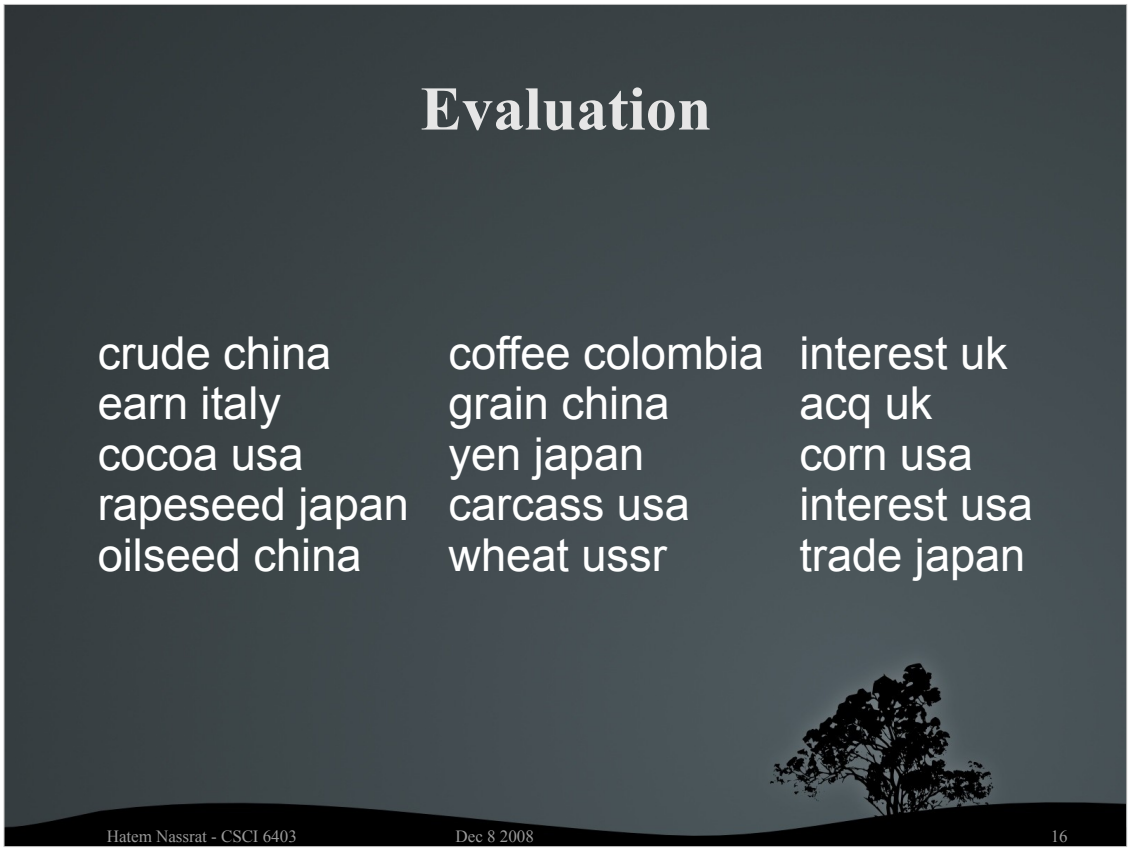
As we can see the number of Quad-Words was less than the number of Triple-Words, the reason being that the Apriori approach was utilized in generating the index. Thus we only look at Double-Words that contain a single word the appeard more the Some Threshold. Similarly Triple only if Double.

Threshold for the run being dicussed here is 50.



One of the questions here is Querying speed. This graph shows the time taken to process a query and return results between the two systems. The MvKid(Red) being the Multi-Word system.

Since most queries ont he web are in the proximity of two to three words we see that both systems have the same performance. The Multi-word starts being significantly slower at 5 word queries and larger.



To evaluate the system, the Reuters 21578 dataset contains Topic and Places keywords attached in the metadata of each document.

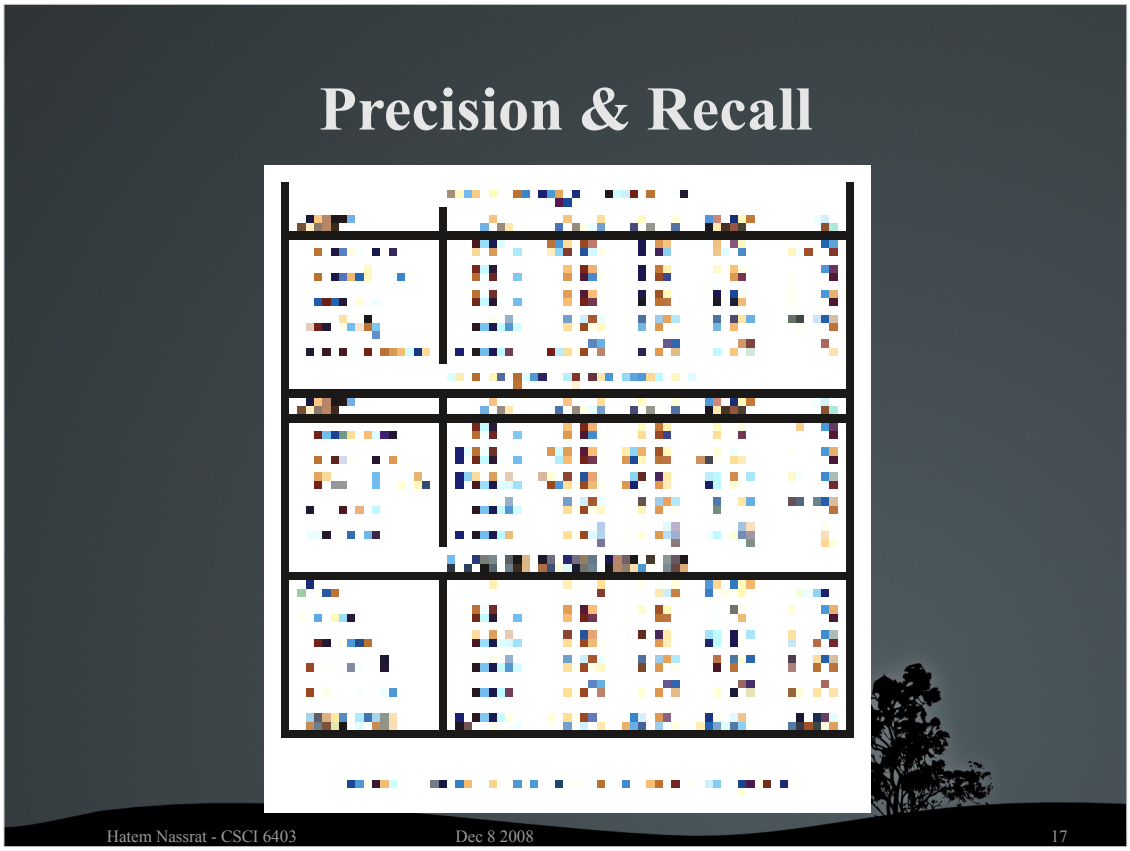
The tags were used to generate these pseudo queries.

The row on the right (crude china) were used in an average of 10 documents each.

For the middle set an avg of 40 documents contained per pseudo query. (30 – 50)

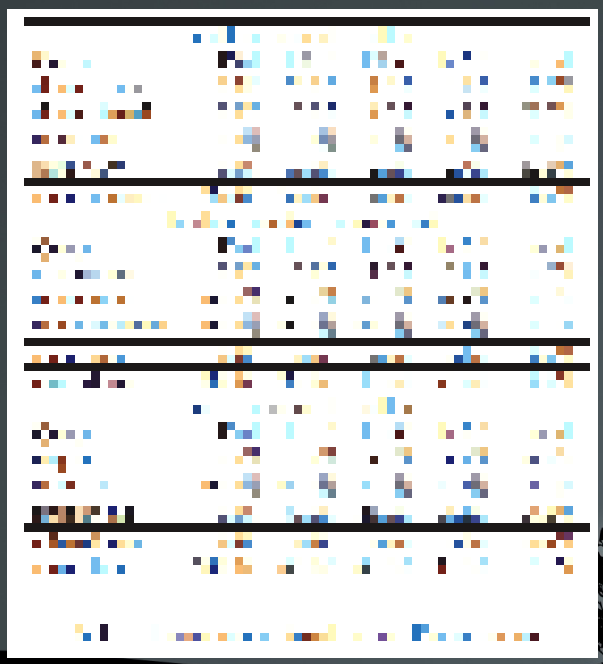
As for the last set an avg of 140 documents were available.

These documents acted as relevant sets for these Queries.



This large mess of numbers displays the precision and recall for the Single Word index S Engine.

Precision & Recall

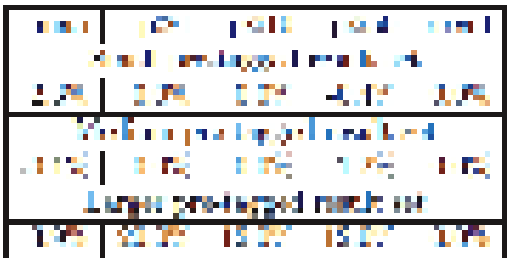


This table displays the precision and recall for the Multi-Word index system.

Noting here that the Vector Space Retreival Threshold for the multi-word system was turned down such that the recall levels are the same, so that we can easily compare the precision values at different intervals and also the MAP.

Precision & Recall

Avg(Multi-Word) - Avg(Single-Word)



To make more sense of the results, I decided to average each of the groups into one value.

This table displays the Average for each group of queries, for each of the systems, by subtracting the Avg returned for the single word system from the Multi-Word System.

As we see here, the significant increase was for the group of queries that had around 140 relevant documents in the dataset.

Questions ?