Multimedia Information Retrieval and Computer Vision

Information retrieval is used in many aspects: email search, search toolbars such as Spotlight, product search such as Netflix, and others.
The interface for these systems is hands and keyboard, but there are systems that use more sophisticated tools, such as conversational search (e.g. Siri). We also have image search, audio search, medical search, legal search, etc.
Information retrieval is a field concerned with the structure, analysis, organization, storage, searching, and retrieval of information. Store, search, and retrieve are the main three activities.
Gerald Salton is the father of information retrieval and provided this definition in 1968.
In 1977 a new definition was given, the complexity of IR arises from the impossibility of describing the content of a document, or the intent of requests, precisely, or unambiguously.
The search problem is still open.
As a general definition, IR is the retrieval of relevant (useful) information from data sources that were not originally intended for access, e.g., unstructured data.
Most often, text (news articles, scientific papers, etc.), images, videos, audio, ...

IR is the science of search engines.
To address the information needs of users we need:

- effectiveness: get the right information to a user
- efficient: get the information to a user quickly

Many new technologies came from search engine technical problems: MapReduce, Distributed databases, data center technologies/protocols...
The main focus of IR was text and docs, while now we move towards images, audio, etc.
Documents, actually, may have a little structure, and some organization, such as title, author, date or intro, pre-chorus, refrain, and post-chorus for a song...

Documents are different from database records.
IR deals with unstructured information while a DB is a collection of related data. By data, we mean something known that can be recorded and that has implicit meaning.
Database records are typically made of well-defined attributes, easy to compare fields with well-defined semantics to queries in order to find matches, while the text is more difficult.

Text Ranking

Given a piece of text (keywords, question, news article,...), rank other pieces of text (documents, queries, passages,...) ordered by their content-based similarity of the first document w.r.t. to other documents.
Comparing the query text to the document text and determining what is a good match is the core issue of IR.
We need to compare the query text to the document text and determine what is a good match. This is the core issue of IR.
The exact matching of words is not enough, and we need to consider that there are many different ways to write the same thing in a natural language like English.

The main differences are the following:

	Databases	Information Retrieval
Unit	Structured data Clear semantics based on a formal model	Mostly unstructured data Free text with some organisation
Queries	Formally define queries Unambiguous	Vague and imprecise information needs
Results	Correct in a formal sense Closed-world assumption	Sometimes relevant, often not Open-world assumption
Match	Exact match No ranking	Best match Ranked
User Interaction	One-shot queries No interaction	Interaction is fundamental

Close-world assumption: all we need to know is contained in the DB.
Open-world assumption: we know nothing.
The interaction with the user is fundamental in IR, we do not just need a one-shot interaction.

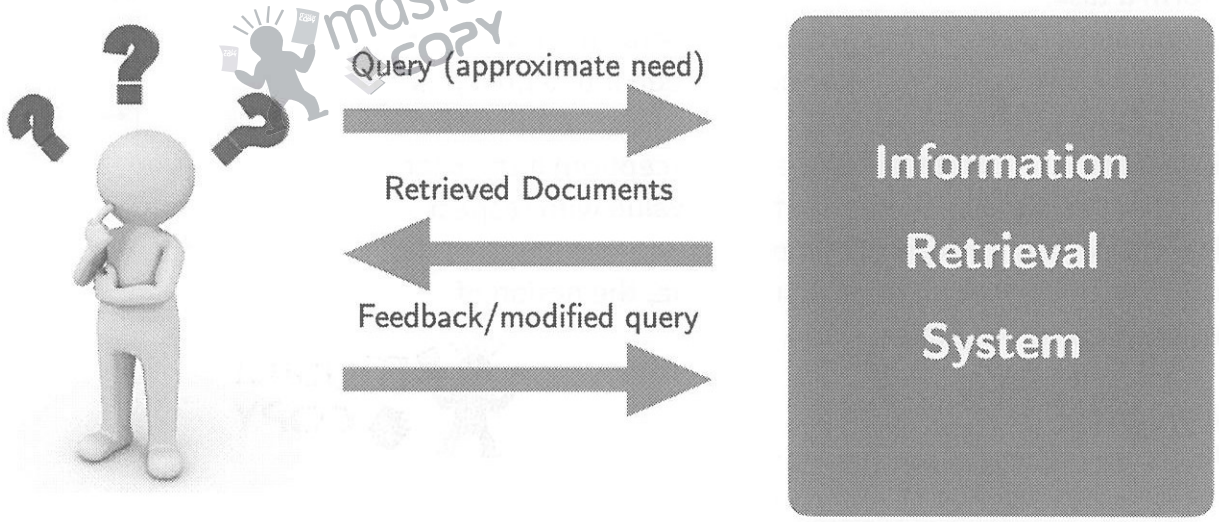
We have **imprecision**, and no clear semantics, in IR.
Most algorithms in computer science have a correct answer. In contrast, a heuristic tries to guess something close to the right answer.
There is one only possible correct answer in problems such as 'Sort list of integers', 'find the max', etc.
Now, let's consider different problems, such as 'Find the document most relevant to the query "hippos in the zoo"'.
IR techniques are essentially heuristic because we do not know the right answer.
There is no correct answer.

What makes a document relevant?
Does it contain all query terms? many times? query terms close together?
There is no way to understand if they compute the correct answer in a mathematical way.
All of these are heuristics. They are not guaranteed to get a correct, relevant document for all users.
We want to use IR systems to fill the gap, we need to look for the information need that must be exact and not vague.
E.g. what to do this weekend in Pisa? (vague)
what is the head of SOE? (exact)
Where to search is not easy to understand.
Retrieval means we need to find something we are looking for.
We can distinguish different kinds of search:

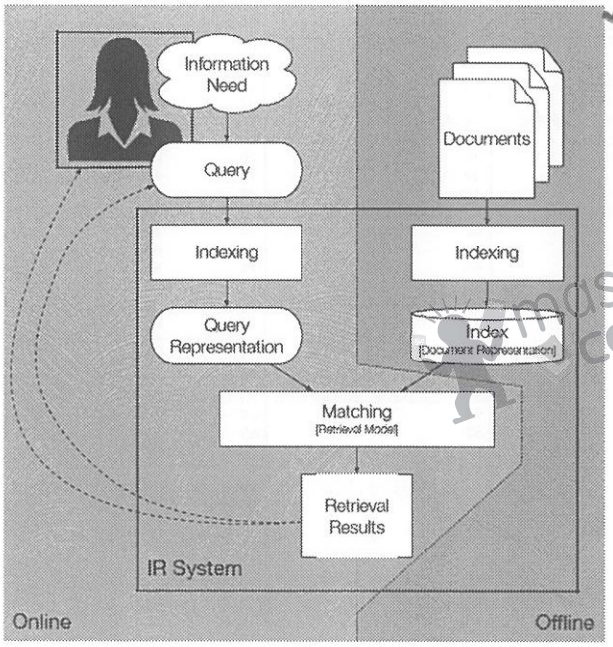
- Ad-hoc search: find documents about a certain topic
- Known item search: find the Unipi homepage
- Answer seeking: what is the capital of Italy?
- Exploratory search: who makes good wines in Pisa?
- Decision making: best places to stay in Pisa
- Expert search: who knows about x in my organization?

The retrieval scenario is the following.
The user provides a question, we give the answer, and then the user provides an assessment (is the answer good?).
Asking a good question can be as hard as computing the answer.
The objective of the IR engineer is to automate this process.
We need to create an IR system that can:

- Infer what a user wants
- Fetch relevant documents as fast as possible
- Present them to the user in a way that they understand



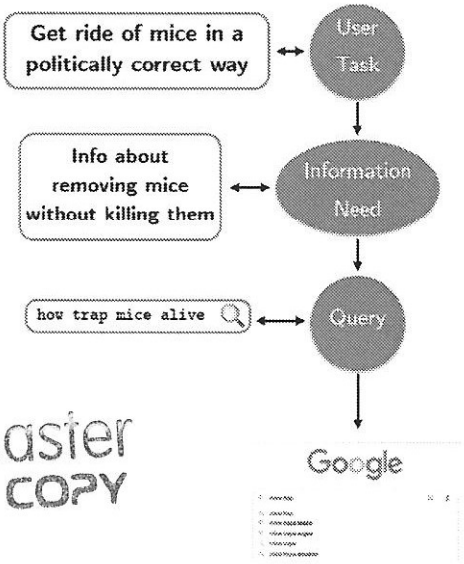
Basic Retrieval
The Y Diagram is the following:



Offline: activity for which we have a reasonable amount of time before interacting with the user
Online: interacting with the user.

Build a representation of a collection of documents on top of which we can run algorithms
Index: representation of document collection, data structure important for IR.
It has to be built, and this has to be done by the indexer
On top of it, we perform the matching of user queries in order to find relevant information.
We need to perform operations on the query in order to be executed on the index.
The matching component is the one responsible to perform matching operations.
The online part goes on and on forever.

The information need is the lack of knowledge needed to perform a task.
E.g. Information required to execute a task for the user's job
The user translates her/his information need into a query, i.e. a set of keywords.
A document is relevant (relevance is a perception) if the user perceives it as containing information of value with respect to her/his personal information need.
Because the information need is subjective, the notion of relevance is also subjective.



master COPY

Text Processing
Different kinds of encodings options can be chosen.
The **ASCII** encoding maps characters to 7-bit integers, using the range from 0 to 127 to represent 94 printing characters, 33 control characters, and the space.
Since a byte is usually used to store a character, the eighth bit of the byte is filled with a 0.

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

ASCII does not provide a way to encode characters from other alphabets, such as Cyrillic or Greek.
Text is not always written in latin characters, so we use other encodings.
Unicode is a standard for representing written symbols, but not an encoding.

Unicode contains the alphabets of almost all known languages, as diverse as Japanese, Chinese, Greek, Cyrillic, Canadian Aboriginal, and Arabic.
A character is defined as the smallest component of a written language that has semantic value.
The Unicode Standard defines a codespace, a set of numerical values ranging from 0 through 10FFFF.
The number assigned to a character is called a code point.
A code point is denoted by "U+" following by a hexadecimal number from 4 to 8 digits long.
Most of the code points in use are 4 digits long.
For example, U+03C6 is the code point for the Greek character ϕ.
Unicode is not a character encoding scheme. Unicode code points are just numeric values assigned to characters. Code points are not representations of characters as sequences of bytes.
For example, the code point U+0C36 is not a sequence of the bytes 0x0C and 0x36.
There are several encoding schemes that can represent Unicode, including UTF-8, UTF-16, and UTF-32.
In **UTF-8** characters are encoded with anywhere from 1 to 6 bytes, depending on the code point.
In other words, the number of bytes varies with the character.
In UTF-8, all ASCII characters are encoded within the 7 least significant bits of a byte whose most significant bit is 0.
Files and strings that contain only 7-bit ASCII characters have the same encoding under both ASCII and UTF-8. UTF-8 is backward compatible w.r.t. ASCII.

All Unicode characters larger than U+007F are encoded as a sequence of two or more bytes, each of which has the most significant bit set.
This means that no ASCII byte can appear as part of any other character, because ASCII characters are the only characters whose leading bit is 0.
The first byte of a multibyte sequence that represents a non-ASCII character is always in the range 0xC0 to 0xFD and it indicates how many bytes follow for this character.
We say how long the encoding is.
One of 110xxxxx, 1110xxxx, 11110xxx, 111110xx, and 1111110x, where the x's may be 0's or 1's.
The number of 1-bits following the first 1-bit up until the next 0-bit is the number of bytes in the rest of the sequence.
All further bytes in a multibyte sequence start with the two bits 10 and are in the range 0x80 to 0xBF.
UTF-8 sequences must be of the following forms in binary, where the x's represent the bits from the code point, with the leftmost x-bit being its most significant bit:

```
0xxxxxxx
110xxxxx  10xxxxxx
1110xxxx  10xxxxxx  10xxxxxx
11110xxx  10xxxxxx  10xxxxxx  10xxxxxx
111110xx  10xxxxxx  10xxxxxx  10xxxxxx  10xxxxxx
1111110x  10xxxxxx  10xxxxxx  10xxxxxx  10xxxxxx  10xxxxxx
```

The first row is the standard ASCII representation.
If we have two bytes it is going to start with 110, and so on.