Great Ideas in Computing

University of Toronto CSC196
Fall 2023

Week 6: October 16 - October 20 (2023)

Week 6 slides

Announcements:

- I have posted Assignment 2 which is due October 27, 9AM.
- The first quiz will take place Friday, October 20 during the tutorial class.
- I will comment on the nature of the quiz
- Aniket tells me that most students (who attended the tutorial) would prefer that we do a somewhat less technical topic and I will move on a discussion of search engines before going on to complexity theory.
- On October 31, our second guest Ashkay Srinivasan, will lead a discussion on "modern cryptography".

This weeks agenda

- The outline for showing the undecideability of natural problems; putting together universal Turing machines, reductions, encoding Turing computations.
- Diagonalization and proving the undecideability of the halting problem.
- Start Search Engines if there is time.

Outline for showing the undecideability of natural problems.

There are a number of ingrediants we will need to show that a natural question, such as the Entscheidungsproblem, is undecideable.

Here are the ingrediants we will need to show and we can show them in any order. Since we will eventually need to show that the halting problem is undecideable, we will soon sketch a proof using a diagonalization argument.

- Recall the definition of a specific Turing machine (TM) and a halting Turing machine computation.
- Viewing a halting specific TM and a TMe computation as a string of symbols.
- Encoding a halting Turing machine computation as a statement in predicate logic. The undecideability of the halting problem then implies the undecideability of predicate logic (i.e. is a statement in predicate logic true of not).
- Show that the halting problem is undecidedable

Formalization of a Turing machine

- Formally, a Turing machine algorithm is described by the following function $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
  Q is a *finite* set of *states*. Γ is a finite set of symbols
  (e.g., $\Gamma = \{\#, 0, 1, a, b, \ldots\}$ and perhaps $\Sigma = \{0, 1\}$)
- Note: Each δ function is the definition of a single Turing machine; that is, each δ function is the statement of an algorithm.
- We can assume there is a halting state q_{halt} such that the machine halts if it enters state q_{halt}. There is also an initial state q_0.
- We view a Turing machine P as computing a function $f_P : \Sigma^* \rightarrow \Sigma^*$ where $\Sigma \subseteq \Gamma$ where $y = f(x)$ is the string that remains if (and when) the machine halts. There can be other conventions as to interpreting the resulting output y.
- Note that the model is precisely defined as is the concept of a computation step. A *configuration* of a TM is specified by the contents of the tape, the state, and the position of the tape head. A computation of a TM is a sequence of configurations, starting with an initial configuration.
- For decision problems, we can have YES and NO halting states.

Formalizing a TM computation

We observe that since Q and Γ are finite, every TM can be represented by a finite length string and therefore we can enumerate $\mathcal{M}_1, \mathcal{M}_2 \ldots$, the infinite but *countable* set of a Turing machines. Is that clear?

Formalizing a TM computation

We observe that since Q and Γ are finite, every TM can be represented by a finite length string and therefore we can enumerate $\mathcal{M}_1, \mathcal{M}_2 \ldots$, the infinite but *countable* set of a Turing machines. Is that clear?
A halting computation is a composition of Turing machine *configurations* $C_1, C_2, \ldots, C_t$ such that C_1 is the initial configuration, C_{i+1} is the configuration that follows from executing one step of the Turing machine when it is in configuration C_i and C_t is in a halting state.

In the transformation reducing $\mathcal{L}_3$ to $\mathcal{L}_2$, we used the fact that a Turing machine can simulate the computation of another Turing machine. This is what Turing called a *universal Turing machine* (UTM). In modern terms, a UTM is an interpreter.

A universal Turing machine (UTM) $\mathcal{U}$ is a T.M. such that

$$\mathcal{U}(< \mathcal{M} >, w) = \mathcal{M}(w)$$

That is, $\mathcal{U}$ simulates exactly what $\mathcal{M}$ does on input w. Turing showed how to design a UTM.

What does this have to do with predicate logic?

The simplicity of a TM make it possible to describe each individual step of a TM so that we can encode the condition that $\forall i[C_{i+1}$ is the next configuration after $C_i]$.

Suppose that one possible step of the TM is $\delta(q, s) = (q', s', R)$ for some $q, q' \in Q, s, s' \in \Gamma$. Then We can express the following condition: if the TM is in configuration C_i, where the TM tape head is at tape square j and is reading symbol s, then configuration C_{i+1} must indicate that the symbol in tape square j is s', the tape head is now at tape square $j + 1$, and the state is s'. Similarly if $\delta(q, s) = (q', s', L)$ then the tape head in C_{i+1} will be reading square $j - 1$.

We also need to express that in initial configuration C_1, the input $\sigma = \sigma_1, \sigma_2, \ldots \sigma_n$ is written on the first first n squares, the tape head is reading tape square 1, and the TM is in some initial state q_1.

And we need to say that $\exists t$: the state in C_t is a halting state.

But how to we account for our saying that each TM has an infinite tape?

It is convenient in definiting our TM formalism, that we have a special symbol $\perp \in \Gamma$ such that in C_1, $\forall j < 1 \vee j > n$ [tape square j has the symbol $\perp$] to indicate that a blank square exists at this tape square (i.e. nothing has yet been writen on this tape square j).

We will insist that in our formalism, that $\delta(q, \perp) = (q', s', D)$ where $s' \neq \perp$ and $D \in \{L, R\}$.

That is, whenever the TM is reading the $\perp$ symbol, it knows that it has reached a tape square not seen before and hence we want to be sure that the square will forever after not have symbol $\perp$ in this tape square.

Diagonalization

You may have learned in high school why using the diagonalization method it can be proved that the set of real numbers say in $(0, 1)$ is *uncountable* while the set of rationals in $(0, 1)$ is *countable*.

The idea is that since the rationals are countable we can list them and that lets us say that $r_i = \text{ the } i^{th}$ rational number in $(0, 1)$ (in some agreed upon listing of these rationals). Suppose as a binary fraction $r_i = .r_i(1)r_i(2)r_I(3)\ldots r_i(m_i)$ for some m_i and $r_i(j) = 0$ for all $j > m_i$.

We can the create a non-rational number $x = .s_1\ s_2\ s_3\ldots$ where $s_i = 1 - r_i(i)$; that is, we change the i^{th} bit in the i^{th} number which means that x could not be in this list that contains all the rational numbers.

Diagonalization and the halting problem

We will just sketch why the halting problem is undecidable. This is a proof by contradiction assuming that the halting problem is decideable.

Using diagonalization, we can show that the halting problem is undecidable. In particular, we will show that the decision problem for $\mathcal{L} = \{\mathcal{M} \text{ halts on input } < \mathcal{M} >\}$ is undecideable. Recall that $< \mathcal{M} >$ denotes the string representation of the TM $\mathcal{M}$. This is not how we stated the halting problem. Why is this sufficient for showing that the halting problem is undecideable?

The set of Turing machines is a countable set so we can let $\mathcal{M}_i$ denote the i^{th} TM. Consider the following function $f(i) = \mathit{YES}$ if $\mathcal{M}_i(i) = \mathit{NO}$ or $\mathcal{M}_i(i)$ does not halt, and $f(i) = \text{NO}$ otherwise.

If the halting problem were decideable, then using a UTM and the claimed decidability of the halting problem, f would be a computable function but that would be a contradiction since f is defined to be different than every TM $\mathcal{M}_j$.

Finishing the undecideability of the halting problem

Cantor's [1891] diagonal argument (showing that the reals are uncoutable) precedes Turing. What Turing showed is that we can encode whether or not a Turing machine $\mathcal{M}$ accepts input w (i.e. the halting problem) by a statement in predicate logic. This is the argument we sketched.

That encoding is showing that we can reduce (in fact transform) a TM computation into a statement in predicate calculus. So if the Entscheidungsproblem were decideable, then the halting problem would be decideable. This was one of the great ideas in Turing's 1936 seminal paper ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

We recall Hilbert's 10th problem: Can we decide if a multivariate polynomial with integer coefficients has an integer root.
It wasn't until 1970 that Matiyasevich proved that this problem is undecideable. (This follows a line of partial results by Davis, Putnam and Robinson spanning 21 years building on the Church-Turing precise definition for what it means to be decideable.)

New topic: search engines

I think of search engines as a great idea in the sense of being a "killer application" and also leading to interesting computational issues that have energized the field of computing.

In doing so I am mostly talking about search engines as they are mainly used today in terms of searching web documents. That is, search engines that take queries (usually in the form of key words or phrases) and produce a *ranked list* of documents.

I am mostly going to talk about search engines independent of the importance (and necessity) of having large pools of fast machines, high speed communication and massive storage.

We ended Monday's class asking if large language models will make search engines obsolete? (Previously we asked if large language models will make Wikipedia obsolete and the consensus was NO.

Search engines intro continued

That is, I am mostly going to talk about search engines in terms of their functionality and the basic computational ideas that make them work (so) well. This is another example (like deep neural nets) of a great idea where greatness depended on new technology. Todays quality search engines simply could not exist say using the technology of the 1960s and 70s.

It is also an example where its greatness may also be an inhibitor for thinking about how to "significantly" move beyond the *current norm of key word based search*.

Of course, in some sense we have moved beyond just key word search in that one can now input an image and ask the search engine to find examples like that image. And in addition to ML being used for image recognition, ML is now playing a more algorithmic role in the quality the key word search.

But still from a functional point of view, while the quality of search has greatly improved, we are still basically doing what we did since say the mid 1990s.

A little search engine history

Search engines are part of the topic of "information retrieval" once the domain of library science. Computerized information retrieval has been an application idea since the start of modern computing.

On the web page there is a link to a prophetic July, 1945 Atlantic article "As We May Think" by Vannevar Bush where he envisions something quite close in many respects to the modern web and hyperlinked documents.

The article begins with the following: "Consider a future device . . . in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory."

That is, some kind of semi-automated information retrieval has been explicitly thought about for almost 80 years.

Some quotes from the Vannevar Bush article

There are a lot of anachronisms (in terms of what the underlying technology will be, gender roles) in this article but more important there are many insightful ideas about the future of accessing information. Here are some quotes from that article.

"Much needs to occur, however, between the collection of data and observations, the extraction of parallel material from the existing record, and the final insertion of new material into the general body of the common record. For mature thought there is no mechanical substitute. But creative thought and essentially repetitive thought are very different things. For the latter there are, and may be, powerful mechanical aids."
Note: Bush is then clearly drawing his line here between human intelligence (and say creating new knowledge) vs retrieving existing knowledge.

More quotes from Bush's article in the Atlantic

"Our ineptitude in getting at the record is largely caused by the artificiality of systems of indexing. ... The human mind does not work that way. It operates by association. With one item in its grasp, it snaps instantly to the next that is suggested by the association of thoughts, in accordance with some intricate web of trails carried by the cells of the brain."

"Man cannot hope fully to duplicate this mental process artificially, but he certainly ought to be able to learn from it." In minor ways he may even improve; e.g., for his records have relative permanency. The first idea, however, to be drawn from the analogy concerns selection. **Selection by association, rather than indexing**, may yet be mechanized."

"Wholly new forms of encyclopedias will appear, ready made with a mesh of associative trails running through them, ready to be dropped into the **memex** and there amplified." Think now of hyperlinks.

"Consider a future device for individual use, which is a sort of mechanized private file and library. It needs a name, and, to coin one at random, memex will do."

The debate as to the nature of information retrieval

In the 1960's and 70', there was a "debate" (albeit not widely discussed outside of those interested in information retrieval) between those who felt that information retrieval (IR) (i.e. finding documents to satisfy an "information need") was a subfield of AI (and more specifically natural language understanding) verses those who thought it could be best realized by more well established combinatorial, algebraic and statistical ideas. Bush seems to have already settled his views well before this debate is taking place

That is, one constituency felt that we needed to be able to "understand" what a document was saying (and what people were requesting) so as to find relevant documents.

The other constituency felt that the claims of many AI researchers were not at all feasible and that again a more statistical/algebraic/combinatorial approach (devoid of any real "intelligence") would produce better results.

The debate continued

I had a course (1967) in IR from Gerald Salton, who (according to Wikipedia) was "perhaps the leading computer scientist working in the field of information retrieval during his time". His group at Cornell developed the SMART Information Retrieval System".

I am not a great historian but I believe the vector space model (which we will discuss) was his idea. Salton was a proponent of the statistical/algebraic/combinatorial approach. I think that he always felt that AI was over-hyped.

So who won the debate?

The debate continued

I had a course (1967) in IR from Gerald Salton, who (according to Wikipedia) was "perhaps the leading computer scientist working in the field of information retrieval during his time". His group at Cornell developed the SMART Information Retrieval System".

I am not a great historian but I believe the vector space model (which we will discuss) was his idea. Salton was a proponent of the statistical/algebraic/combinatorial approach. I think that he always felt that AI was over-hyped.

So who won the debate?

As of today, it is clear that the approach of the constituency represented by Salton has turned out to be the basis for the way we currently do search in the internet.

The debate continued

I had a course (1967) in IR from Gerald Salton, who (according to Wikipedia) was "perhaps the leading computer scientist working in the field of information retrieval during his time". His group at Cornell developed the SMART Information Retrieval System".

I am not a great historian but I believe the vector space model (which we will discuss) was his idea. Salton was a proponent of the statistical/algebraic/combinatorial approach. I think that he always felt that AI was over-hyped.

So who won the debate?

As of today, it is clear that the approach of the constituency represented by Salton has turned out to be the basis for the way we currently do search in the internet. However, search engines today do incorporate ML into their retrieval algorithms. And I think large language models are doing more (e.g., composing text) than Vannevar Bush or Salton envisioned.

Key word search

At a very very general level, we can think of current search as the following process:

1. A user converts an "information need" into a query (i.e. a set of key words)
2. The search engine is then an algorithm for the mapping: query $\times$ {collection of documents} $\rightarrow$ <ranked list of "relevant documents">.
3. Upon receiving highly ranked documents, the user may choose to refine the query.
4. This process continues until the user is either satisfied or gives up. How often do you have to refine your queries? How often do you abandon a search?

As we discuss the ideas behind key word search in search engines, it should be noted that there are many specific ideas and engine specfic details that go into making a search engine successful (in terms of the quality, speed, and coverage) and these ideas and details are kept reasonably confidential. Why?

Why the secrecy?

There are two main reasons for not disclosing specific search engine ideas and details:

- Not suprisingly, these ideas are trade secrets that give a company an edge
- Perhaps less obvious, knowing exactly how a compnay does its searches allows one to easily spam documents so as to raise their ranking (and hence lower the quality of the ranking).

So please be advised that what I am discussing is just the high level ideas and not the specifics say being utilized by Google, Yahoo or Microsoft.

It clearly took significant progress in technology (i.e., the speed and memory capilities of large numbers of distributed machines) to make key word search as successful as it is today. Equally important, many significant algorithmic ideas plus extensive and ongoing experience with user requests has been necessary for search engine success.

However, collecting information from user interactions is, of course, an important privacy issue.

The challenge of real time information retrieval

In addition to algorithmic ideas used to improve search quality (i.e., precision, recall), commercial search engines are dealing with enormous collections of sites/documents and must return responses in what appears to be "real time".

Estimates of the size of the web vary. One site (WorldWideWebSize.com) provdes daily reports on the size of the web: That site reported "The Indexed Web contains at least 5.42 billion pages (Sunday, 04 October, 2020)" but as of October 14, 2021, it reports "The Indexed Web contains at least 4.81 billion pages". Did the size really decline?

Precision in a set of documents (for an information need) is defined as the fraction of documents that are relevant. In a ranked list we can say that precision means that the higher the rank of the document, the more likely it is to be relevant.

Recall in a set of documents is defined as the fraction of all relevant documents in the set. In a ranked list of retrieved documents (where there can be many thousands of relevant documents), we want the most relevant documents to occur earliest in the ranked list.

Do we want diversity in the documents retrieved?

We may (or may not) want the highest ranked documents to reflect some desired diversity.

For example, what if I provide the query "What did Donald Trump accomplish as US president"? Do I want just what is reported by people who like him? Or do I just want opinions that are negative? Or do I want a diversity of opinions? Do we want denials of the Holocaust to be presented in the name of diversity. What is a "legitimate" opinion vs a conspiracy theory devoid of facts?

Similarly, when considering the stock market, do I want some overall assessment, or do I want reports on different sectors of the market?

Even for a more classical and now perhaps a more mundane example, when I ask for recent information about "jaguars", do I mean the car, the animal or the NFL football team? I probably only want one of these. When I make my request clear, a search engine should avoid ambiguous meanings. Should a search engine use my previous history of requests to better identiffy the most relevant documents personalized for me?

The basic bag of words model

Suppose $\mathcal{C} = \{D_i\}$ is a collection of web documents (URLs).

- We can treat each document as a *bag of words*. Let's just say 200 words per document as some very rough average.
- Each query can also considered as a very small bag or words. Most queries are two or three words. One estimate is an average of 2.2 words per query. Slightly more than half use only 1-2 words.
- The most naive approach. Find all the documents that contain all the words in the query. As a naive first approach call these the "relevant documents".
- The most naive way to find all these (potentially) relevant documents would look at each document and check if all the query words occur.
- Even if all the documents were stored locally (which is not possible), what would be a rough estimate for the time to find all the relevant documents?

A quick calculation

You can do a quick calculation: compute $|\mathcal{C}| \cdot \frac{\text{number words}}{\text{document}} \cdot \frac{\text{number words}}{\text{query}}$ and then divide by $\frac{\text{number comparisons}}{\text{second}}$ to estimate the time for naively looking for documents that contain all the query terms.

Let's say that we have approximately 5 billion URLs, 200 words per document, 2 words per query which naively would result in $2 \cdot (10)^{12}$ comparisons. And let's say $(10)^7$ comparisons per second. Then a query would take $2 \cdot (10)^5 = 200,000$ seconds.

OK I might have some miscalculations but clearly this is *not* "real time" and not even feasible.

Making search feasible

One simple idea but but very useful idea is the following. When a search engine *crawls* the web to find documents, it indexes documents so that for each *term* (i.e., word and frequent 2 word and 3 word phrases it maintains a sorted list of documents that contain that term. We usually ignore common articles such as "the", "an, etc.

A term may also represent a number of strongly related terms. For example, a match for "cook" might be satisfied by "cooking" or "chef".

Making search feasible

One simple idea but but very useful idea is the following. When a search engine *crawls* the web to find documents, it indexes documents so that for each *term* (i.e., word and frequent 2 word and 3 word phrases it maintains a sorted list of documents that contain that term. We usually ignore common articles such as "the", "an, etc.

A term may also represent a number of strongly related terms. For example, a match for "cook" might be satisfied by "cooking" or "chef". You can get spelling suggestions, or maybe get a partial match, and sometimes be told that no documents match your query or there are no good matches but still get some suggested matches.

What can happen often is that there be too many documents matching the query terms. So as we already suggested we really need a ranked list of documents in which the "most relevant" documents are ranked highest.

The vector space model and ranking documents

Instead of simply matching for query terms, we want to account for the fact that the occurence of certain terms are more important for relevance.

Gerald Salton's idea was that a document (and a query) are represented by a vector of weighted counts of words/terms. Here are some ways to weight the occurences of terms in a document.

1. Count the number of occurences of a query term in a document, and better yet normalize this count by the relative frequency of terms in "the corpus of documents". This normalized count is called *tf-idf* standing for term frequency-inverse document frequency. Terms that occur infrequenly throughout the corpus but appear frequently in a document should be weighted more. Wikipedia quotes a 2015 study that states "83 % of text based recommender systems in digital libraries use tf-idf".
2. Terms that appear in the title of the document or the title of a section heading should be given higher weights.
3. Terms that appear in the *anchor text* are important.

The vector space model continued

The above ideas for weighting terms are independent of the user queries. In contrast, we could also give higher weights to terms that relate to an individuals interests (say as learned by previous searches).

There can be many other ways to weight terms say by using machine learning techniques.

Now once we adopt this vector space representation, we can measure the similarity of a document and a query by say the cosine of these vectors.

An additional idea (in additional to the term similarity of the document and the query) is to expoint the "popularity" of a document. Popularity of a document in Google was done using *page rank* which is basically a random walk on the graph defined by the hyperlinks. This leads to a stationary distribution (i.e., an equilibirum) on the vertices (i.e., the relevant documents).

Some further comments on the history of search engines

Page rank was touted as an essential idea in the early days of Google search but not clear how much of a role it now plays.

At about the same time as page rank, Jon Kleinberg introduced another graph based popularity method called *hubs and auhtorities* which was used in IBMs search engine (which they never commercialized).

With regard to td-idf (now accepted as an important idea), I saw the following comment in a web post (Language Log)
https://languagelog.ldc.upenn.edu/nll/?p=27770)
"one of Marvin Minsky's students once told me that Minsky warned him 'If you're counting higher than one, you're doing it wrong'. Still, Salton's students (like Mike Lesk and Donna Harman) kept the flame alive."

Marvin Minsky is recognized as one of the pioneers of artificial intelligence.

Why is search so profitable?

Companies such as IBM and (initially) Microsoft did not try to commercialize search, not recognizing the profitability of search. Indeed, should one charge for information or should the business model be based on advertising? Or it possible that search would not be profitable?

We now know that search has turned out to be extremely profitable for companies based on advertising. The main way that Google and other comapnies sell advertising for search has spawned major research in algorithm design and auction theory. We will say more about autions, game theory and mechanism design.

We can view the process of assigning queries to advertisers (say wanting to display an *ad* as an *online biparitite graph matching problem*).

When a query arrives it needs to be assigned to one (or more, depending on how many advertising slots will be displayed) ads.

The "adwords" assignment problem

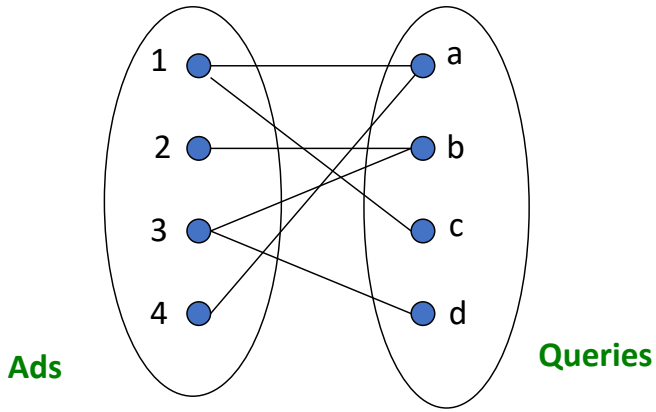

Figure: Figure taken from USC lecture notes by Rafael Ferreira da Silva

Each advertsiser may have a budget (say for a given day) and indicates for given queries (or keywords) what it is willing to pay for that query but never exceeeding its budget for all the queries assigned to that advertiser.

The search engine adjusts this advertiser *bid* for a query based on how well it thinks the ad matches the query and then decides whether or not to assign an advertising slot to an advertiser and the price paid by the advertiser (depending on the slot) for each click by search users for the ad.