Project 1: Mars Attacks!?: Reconstructing Sentences

2016/1/27
CSC 200H: Undergraduate Problem Seminar
Prof. Lane A. Hemaspaandra

Due: February 29, 2016, 2:00.00000PM (please note that carefully so you don’t get a zero or sharply lowered grade after having done lots of work!!!); also, fyi, that the first intermediate workshop on this will be 2/8 and the second intermediate workshop on this will be 2/22 and the conference days on this will be 2/29 and 3/2 and the Battle of the Unscramblers will be on 3/2.

1 Project Description

The strangely ungrammatically and suspiciously in-alphabetical-order-word-by-word agency, ANS (Agency National Security), has been put in charge of analyzing an ongoing sequence of encrypted burst transmissions, seemingly all coming from Mars. (The ANS’s experts on Mars have long known that Mars is inhabited by sentient creatures, and those experts believe that those creatives are obsessively interested in the English language, and use it extensively for communication, sometimes communicating their own thoughts, but sometimes communicating various tidbits the Martians themselves had picked up from their extensive monitoring/reading-from afar—their technology is apparently very advanced, as is their cryptography—of TV, radio, books, music, computers, and a variety of other sources.

Each of the burst transmissions seems to have two information components. The first components have been successfully decrypted. Each seems to be a collection of tokens, formed by taking a sentence (such as “I like computer science.”) or a sentence-like object (such as “Wow oh wow!”) and breaking it apart by splitting it at each space, leaving just a multiset of maximal contiguous nonspace character sequences (let us call each a “word” though it is a slightly strange use of the term). But these sequences seem to usually be out of order—often, they just seem to be sorted lexicographically, although not always.

The ANS believes that the second component of each burst contains the permutation to use to put those words back into their original order (perhaps seeded in some way regarding the occurring multiset of letters in use, thus perhaps making it harder to decrypt). In fact, the ANS’s best cryptographers have been completely unable to break the encryption regarding the second components.

Even worse, the ANS has been unable to on its own to reorder the first components. For example, one first component consisted of these five words—“We” “harm.” “mean” “no” “you”—and has completely stumped the ANS (said one expert, “the number of possibilities is exponential in the number of words!”), though the ANS is very worried about the word “harm” and fears that a Martian attack is imminent!

Now, the ANS knows that powerful research on reordering scrambled sentences was done at the University of Rochester, in the year 2010. This research of course was highly secret, and the work was never released to the public. Obviously, that has not kept the ANS from knowing of it, since the ANS knows far, far more about all things in the electronic world than people realize. The ANS also knows, though, that that research paper was done under a clause—almost identical to the one in the Spring 2016 CSC200H’s syllabus—that allows the paper to be made available to
future CSC200H students to use (with proper citation being made, of course!) in their own related research.

So the ANS has approached President Seligman, and asked him to assign to this reordering problem—currently the ANS’s top priority—his most powerful resource: the students of CSC200H. President Seligman has agreed (purely out of patriotism, and having nothing to do with the ANS’s comment to him that it knew exactly where the $16,997,104 deposit he made into a Cayman Islands bank account came from and that the planned $17,000,000 new Data-Science/Computer-Science Building will instead be replaced by a $2,896 circus tent), and indeed President Seligman has promised the ANS that the entire CSC200H class, separately in teams (following all the rules of the syllabus including those on limits on contact between teams), will work on this reordering problem.

So here is the task of each group on Project 1. You are to write a program that take a first component, and outputs a reordered version. Your program’s goal is to output the best possible reordered version, although of course your program must run reasonably quickly also. “Best possible” is of course quite a murky notion. Since it is believed that most of the scrambled sentences were grammatical, almost all first components probably unscramble to grammatical sentences. But even within that, some are probably much more plausible than others. And of course, the words have meaning, and that itself might make some unscramblings a better choice than others... although having your computer program take advantage of that could be utterly insanely difficult and will you a Turing Award or two (but President Seligman has assured the ANS, “Those CSC200H do ‘utterly insanely difficult’ each day before breakfast.”). For example, if a first component (apparently written by a Martian fan of the poet e e cummings) was this collection of words—“ran” “driver” “red-light” “the” “the”—then “the driver ran the red-light” is more likely than “the red-light ran the driver” is. Again, that comment/preference itself is using real-world knowledge, and so might be utterly crazy-hard/insanely difficult to inch toward (although let me mention in passing that the course TAs all have knowledge representation as one of their top areas of interest; perhaps on the start-up day we’ll have each say a few words, trying to—not solve the problem for you—point you towards potentially useful resources). Even just unscrambling first components into things that are grammatical might be very, very hard, and into things that are grammatical and not ludicrous might be much harder still.

But doing as well as possible on this unscrambling task is a goal of each group on this project. So, big-picture, your task (each group’s task, that is) is to write (following the rules of the course) a research paper exploring this problem. You of course will have written a program (or perhaps multiple ones, that you then compared) to do the unscrambling. And your paper will be largely about your program(s): How it works, what techniques it uses, why you used those techniques and how they fit together, what techniques you tried and decided were not wise to use, what cases it handles well and what its weaknesses are (and how you would address/remove those if you had more time), and so on.

Ideally, your paper will be written in \LaTeX, since is flexible and powerful and the norm for scientific papers (and you can include pictures/images/etc.; see the book “The \LaTeX Graphics Companion” for how); however, if you really must, you may also write it in some other system (such as—sigh—Word or any other decent system). (Note: I will put up on the course web site one of my own \LaTeX papers, as source, so that you can use it as a \LaTeX template, if you wish.)

Don’t despair if you find the problem extremely hard or nearly or outright impossible. It probably is impressively hard—but that doesn’t mean you won’t be able to learn things that address it, or special cases of it, or that do far, far, far better than just outputting back the words
in the order they were given to you.

Basically, poke around, do your best to achieve as strong performance as possible. We’ll discuss the flavor of this in class a bit on the starting date. And of course, as always, we’ll before the presentation day have two intermediate days (February 8, and February 22) during which you share, present, defend, discuss, etc. your ideas/progress/etc.

Your group’s hand-in must be sent in using the rules in the course information document (so one group member will be sending in, all zipped together in a single email to csc200staff at cs.rochester.edu that is cc’ed openly to every group member including the person doing the emailing, your paper, and the slides you’ll be using in your 2/29 or 3/2 talk, and probably your code and executable version and sample runs (but most likely what we will read and grade most closely—or perhaps solely—is your paper, though we also might for example run your program on some examples and see how it performs); that email must arrive by 2:00.000000PM—no extra seconds—on 2/29, even if your group is one of the ones speaking on 3/3, since if it is late the extremely harsh lateness penalties from the course information document will be applied, strictly; and do read the hand-in rules in the course information document as there are lots of things you must do, e.g., including also a good pdf version of your slides if your slides are in PowerPoint).

In what order will the groups go? On both intermediate workshop days and at the final conference presentation day pair, we’ll go in alphabetical order by house name (on the conference day-pair, if we have three groups, then Houses Lannister and Stark will go on Day 1, and on Day 2 we’ll have Targaryen and the Battle of the Unscramblers). (Note that your group will come in to each of your intermediate workshops and your final conference day with a slide presentation, timed to take (this assumes that there will be three groups) for the workshop ones 19 minutes (plus there will probably be about 2 minutes for questions at the end of each talk) and for the conference ones 30 minutes (plus there will probably be about 4 minutes for questions at the end of each talk), that your group will present to the class. And, again, as per the course information document, you’ll also email those slides to “csc200staff” by 2PM on the day of your intermediate talk for intermediate talks but by 2PM on 2/29, along with your paper/etc., for your conference talk. And of course, at each of the workshops and the conference talks that you give, as per the course information document, do make sure to bring into class four printouts—2-up and 2-sided is ok, and of course each stapled together—so that the TAs and I can have those in hand during your talk.)

Good luck, and do well. The ANS and President Seligman have put the fate of Planet Earth and the honor of UR in your hands!

2 Very Important Additional Information

The 2010 Paper It is very important that on this project, and on all things, credit be given to earlier work, sources, etc. So let me stress that be basic problem that this project is addressing—unscrambling sentences—is not due to me. Rather, both the idea of that challenge and attacks/programs as to solving it were a Project 2 of a group in the Spring 2010 CSC 200/200H class. (This is the first time, ever, in this course that I’ve made a project be based on something by a previous year’s students... but their problem indeed is sooooooo very interesting and so deeply challenging and open-ended that it should make a very interesting project.) That group’s paper is dated April 21, 2010, is entitled “Sentence Unscrambler,” and is by Omar Mustardo, Chiranjeevi Raghunath, Darcey Riley, and Michael Schermerhorn. I don’t want to put the paper itself on our
course web site, as that would make it available to the entire Internet, but I will make the paper
available to you—the course’s students—by putting it as a document in our course’s Blackboard
area.

You certainly should openly cite it for the general flavor of this problem, unless you find another
even earlier source (and of course, on all papers, combing the literature for all relevant work
and then discussing it is important, so please make sure to do that here), but even if you find
earlier source(s) to discuss/cite, you’ll in any case want to mention also the 2010 earlier work,
and discuss what it does, and how your approach differs. A right way to cite the paper could be
(with whatever italics/names-or-initials-and-ordering-of-first-and-last-names/quotation marks/etc.
match your bibliography system’s approach; using a .bib file and BibTEX is a dream-case approach
to bibliographies in LaTeX) something like this: Omar Mustardo, Chiranjeevi Raghunath, Darcey
Riley, and Michael Schermerhorn, Sentence Unscrambler, unpublished manuscript, April 21, 2010.

I should mention that your task is not exactly the same as theirs. In our version of the problem,
you have some additional signals coming in, due to the punctuation being “attached” to the words
it is contiguous with. For example, the sentence “I came, I saw, I conquered.” would have these—
“came,” “conquered,” “I” “I” “I” “saw”—as its multiset of words. And the sentence “Dogs like
cats.”—would have these—“cats.” “Dogs” “like”—as its multiset of words. In our version, you get
the words with their case intact and that might also be helpful (in the 2010 paper’s examples, the
sentences always start with ‘I’ so I can’t tell if, when another word is starting the sentence, they
would in the input preserve its upper-caseness-due-to-starting-a-sentence; but in our setting, we do
preserve that (because that is how Martians roll))

A two final comments about the 2010 paper. First, I’d urge each group to, before reading
the 2010 paper, try to yourself think of as many approaches as you can to the problem. You
might think of some wonderful approaches that, if you had first read their paper, you would not
have (as reading a paper often focuses on the approach the paper took). Second, of course
do read that paper. You’ll find all sorts of ideas and progress from a group of tremendously
bright students. I should mention in passing that they actually in their initial plan had other
cool ideas that I think did not make it into their final paper/program, e.g., using the case of
pronouns—I/he/she vs. me/him/her to help with the unscrambling. As you’ll see in their paper—
see the second paragraph of their introduction section—they “make no attempt to ensure semantic
correctness.” They immediately mention that their particular approach often gives “sentences that
are close to semantically correct”), but on the other hand their conclusion section mentions that
“in most cases, the sentences [that the sixth, best version of our algorithm] forms are not complete
or coherent English.”

Standardized Input Format. Standardized Output Format, Program Name  Your main
program’s executable (which should be able to be run on the URCS-UG Linux machines, though
you can develop it elsewhere) should have as its name your group’s name (but all lower-case). For
every example, if you are House Stark, your program’s executable would be called stark. And it should
take its input from stdin and write its output (and just its output) to stdout. So it could be run
(in a shell on Linux) as follows, if one wanted to run it on a prepared set of inputs:

    stark < examples.input > examples.output.

As to input format, let us adopt precisely the following format, which will allow multiple inputs
to be in the same file. Each first component will be listed as a set of “words” one per line, and then
after that set of words there will be a line that is a string of five = signs. (We will assume that “======” is such a taboo string on Mars that it never, ever appears in an actual communication.)

We’ll also assume that the communications don’t contain kinky things such as $\alpha^3 \Pi$ or en dashes or em dashes or so on. We will also assume that the quote and double-quote marks are not the strange curly ones one sometimes gets, but rather are the plain ASCII ones that one gets by using the double-quote (the opening and closing versions of that are identical) and open-quote and close-quote that are right on the keyboard in Linux-land. We’ll assume that the line-enders are the Linux norm (LF). And when the end-of-file (so control-D if one is running it interactively) is reached the program exits. So, for example, the following is a valid input file that captures communications of the three sentences/sentence-like items (well, $\LaTeX$ is distorting some of the single-quotations marks in what you see in the pdf version, but in the sample inputs file the TAs will prepare, you’ll get an actual real, live file). The sentences:

Jane said, "I love the word ‘the’.
Did the polynomial-time algorithm run quickly?
It’s the end of the world as we know it.

And a file that captures them is (note: each of the 3 here is internally sorted in ascending ASCII-sorted order, but you should not assume that that always is the case on inputs):

"I
Jane
‘the’.
love
said
the
word
=====
Did
algorithm
polynomial-time
quickly?
run
the
=====
It’s
as
end
it.
know
of
the
the
we
world
=====

5
And your algorithm should write the output for each first component on a line (ending with LF), followed by 5 equal signs. So a program might on the above input give the following output (yes, it perhaps didn’t do the world’s best job on the third sentence’s decoding):

Jane said, "I love the word ‘the’." =====

Did the polynomial-time algorithm run quickly? =====

It’s the world of the we as end know it. =====

**Examples from the TAs and from You, and the Battle of the Unscramblers**  The TAs will make up a file ta.input (and will also provide you with ta.output, which will in the above-mentioned format give the sentences from which they created the input file). That will give you some examples to work on, But you also of course will want to make your own sample sets as you work on this project. Since you are not supposed to share programs and data files with other groups, *it is cheating to share your sample sets with other groups.* Please see the syllabus for what types of interactions with members of other groups are and are not allowed (outside of the Workshop class sessions, which themselves are places for open presenting of one’s best work, and for questioning other groups).

(This is not a traditional “training set” and then “test set” kind of project, and so you should not assume that doing well on the “ta.input” set assures that your program will do well on whatever input set, if any, the TAs test your program on after it is handed in. They of course are unlikely to test it on ta.input, since you might have just hardwired in perfect answers to all the items in that!)

We will in class have a brief Battle of the Unscramblers, namely, as the second part of Conference Day 2. For that, each group should, as part of the archive that is their project turn-in email, send in a file pair groupname-battle-examples.input and groupname-battle-examples.output, containing at most three examples. You’ll probably want to send in examples that your program does well on (perhaps displaying something clever and broad your program does) but that you suspect the other programs might find a real challenge. (Using a complex sentence and hardwiring the answer into your program is considered bad form here—try to give examples where your program for some interesting reason does well on a challenging example!). We’ll then at the Battle run all the programs on those examples (the input files) and perhaps also on some examples the TAs or I have made.

**Using Inanimate Things**  It is legal to draw on freely available inanimate resources. So you cannot, while the project is in progress, post questions on this to stackexchange or indeed ask (either directly or electronically) people outside your group for advice/help/information. But if you find some external software tool/resource (e.g., perhaps Moby Part-of-Speech, or WordNet, or a powerful off-the-shelf English parser, or having your program—either in preprocessing or on-the-fly-as-it-runs—draw on the power of Google) that you think would be valuable to your program, you can use/draw on that—of course, only within any constraints that that tool/resource puts on people who use/draw on it, and of course your paper would absolutely have to openly mention it (you can’t use powerful external tidbits or tools and pretend they are made by you, or even use them but not mention/credit them—that would be an Academic Honesty violation).
(Non)Assumptions  It is likely that most of the examples—perhaps all—will be in US English, but it certainly is not impossible that some might be in British English. Note that those two differ in some interesting ways. For example what in US English would be written as:

    I love pretty words, e.g., "color."

might in British English be written as (the period’s moved location and the removed comma are not typos; and even the two periods in “e.g.” sometimes are omitted in British English, though not in this example):

    I love pretty words, e.g. ‘colour’.

You may not assume that in any given input file all the sentences are from the same source, or are from the same source in the same order they are there (except with the words of each sentence scrambled), or even that more than one sentence in the file is from any given source. Of course, it might be the case that sometimes some of those might be the case, and perhaps if you detect that you could take advantage of that. But I don’t off-hand expect that the TAs’ example file will be so kind as to just take some example and shuffle its sentences one at a time and make that your sample file. (Side comment: That would be a very nice variant of this project—trying to do this project for the case where one’s expected inputs each really will be a book or paper, with all the sentences in the same order as they are there, except with each being word-shuffled. Clearly, one could attempt to use that in interesting ways. But that case and how to attack it are not what this project is studying.)