Computational Linguistics and Human Language Technology

(class presentation for ling0001, 11/14/2022)

These slides are selected from some previous presentations,
which are linked in the background reading panel for this lecture:

Centre Cournot, Paris, 5/21/2015

PREM2019, University of Puerto Rico, 5/3/2019

NLPCC2019, Dunhuang, China, 10/13/2019

The context:

The past 50 years
have seen enormous quantitative changes
in the efficiency and reproducibility
of speech and language research,
thanks to advances in digital technology.

The near future will bring even larger changes –
not only quantitative changes in productivity and scale ,
but also qualitative changes in the nature of our research,
enabled by new (semi-)automatic methods.

New sources of data
and new methods of automated analysis
are opening up vast new territories of linguistic research.

We can easily acquire and manage new sources of linguistic data
that are several orders of magnitude bigger than old ones.

Because new methods can do old tasks several orders of magnitude more efficiently,
it's increasingly easy to explore these new datasets in old ways.

We can also easily experiment with completely new approaches to analysis and modeling.

And these new methodologies are rapidly spreading
into all the fields that study speech, language, and communicative interaction,
from poetics, sociology, and politics to psychology and neuroscience.

Unfortunately,
biomedical and psychological research practices
are (for the most part) 20-30 years behind the times
in ways that seriously harm research.

So let's take a brief side trip to explain what I mean by that,
based on a presentation to a workshop on
"Statistical Challenges in Assessing and Fostering the Reproducibility of Scientific Results"

Committee on Applied and Theoretical Statistics (CATS),
Board on Mathematical Sciences and their Applications,
National Academy of Sciences

February 26-27, 2015

What is “Human Language Technology”?

The term HLT was coined by DARPA program managers in the 1990s

Inputs might be
- audio or video that includes people communicating
- texts
- complex semi-structured collections of recordings and texts
- data structures representing (perhaps evolving) understanding

Outputs might be
- computer-created audio/video streams
- texts
- data structures representing (perhaps evolving) understanding

Relevant technological capabilities include speech recognition, machine translation, information retrieval and information extraction, summarization, question answering, optical character recognition, speaker identification, language identification, sentiment analysis, etc.

Today, HLT is more and more widely used.

There are three secrets to its success:

1. Cheap fast digital hardware
2. Ubiquitous digital networking
3. A research management technique
   developed in the 1980s
   and applied increasingly widely since then

The first two driving forces are obvious,
but the third involves some semi-obscure intellectual history.

First, a series of HLT failures:

In the 1960s and 1970s, there were many projects focused on machine translation, natural-language interaction with databases, speech recognition, and speech synthesis.

Based on human-coded rules or constraints,
these generally worked to some extent – but they were

- limited in scope,
- brittle under even modest changes in context,
- expensive to create and port,
- and generally disappointing in performance anyhow.

Many influential people concluded
that these endeavors were hopeless.

A leader in pushing this narrative was John Pierce.

Pierce supervised the team that built the first transistor,
and oversaw development
of the first communications satellite.

So his opinion carried considerable weight.


ASSOCIATES

John Pierce, “Whither Speech Recognition?”, JASA 1969:

“… a general phonetic typewriter is simply impossible unless the typewriter has an intelligence and a knowledge of language comparable to those of a native speaker of English.”

“Most recognizers behave, not like scientists, but like mad inventors or untrustworthy engineers. The typical recognizer gets it into his head that he can solve ‘the problem.’ The basis for this is either individual inspiration (the ‘mad inventor’ source of knowledge) or acceptance of untested rules, schemes, or information (the untrustworthy engineer approach). . . .”

“The typical recognizer ... builds or programs an elaborate system that either does very little or flops in an obscure way. A lot of money and time are spent. **No simple, clear, sure knowledge is gained.** The work has been an experience, not an experiment.”

Tell us what you really think, John . . .

"We are safe in asserting that speech recognition is attractive to money. The attraction is perhaps similar to the attraction of schemes for turning water into gasoline, extracting gold from the sea, curing cancer, or going to the moon. One doesn't attract thoughtlessly given dollars by means of schemes for cutting the cost of soap by 10%.
To sell suckers, one uses deceit and offers glamor."

"It is clear that glamor and any deceit in the field of speech recognition blind the takers of funds as much as they blind the givers of funds.
Thus, we may pity workers whom we cannot respect."

But in 1985, DARPA restarted HLT support

Some smart program managers had an idea.

They designed a speech recognition research program that

- protects against "glamour and deceit"
  because there is a well-defined, objective evaluation metric
  applied by a neutral agent (NIST)
  on shared data sets; and
- and ensures that "simple, clear, sure knowledge is gained"
  because participants must reveal their methods
  to the sponsor and to one another
  at the time that the evaluation results are presented

Needed: Published data and well-defined metrics

David Pallett, "Performance Assessment of Automatic Speech Recognizers”, *J. of Research of the National Bureau of Standards*, 1985:

Definitive tests to fully characterize automatic speech recognizer or system performance cannot be specified at present. However, it is possible to design and conduct performance assessment tests that make use of widely available speech data bases, use test procedures similar to those used by others, and that are well documented. These tests provide valuable benchmark data and informative, though limited, predictive power. **By contrast, tests that make use of speech data bases that are not made available to others and for which the test procedures and results are poorly documented provide little objective information on system performance.**

“Common Task” structure

- A detailed task definition and “evaluation plan”
  developed in consultation with researchers
  and published as the first step in the project.
- Automatic evaluation software
  written and maintained by NIST
  and published at the start of the project.
- **Shared data:**
  Training and “dev(elopment) test” data
  is published at start of project;
  “eval(uation) test” data is withheld
  for periodic public evaluations

Not everyone liked it

Many Piercians were skeptical:

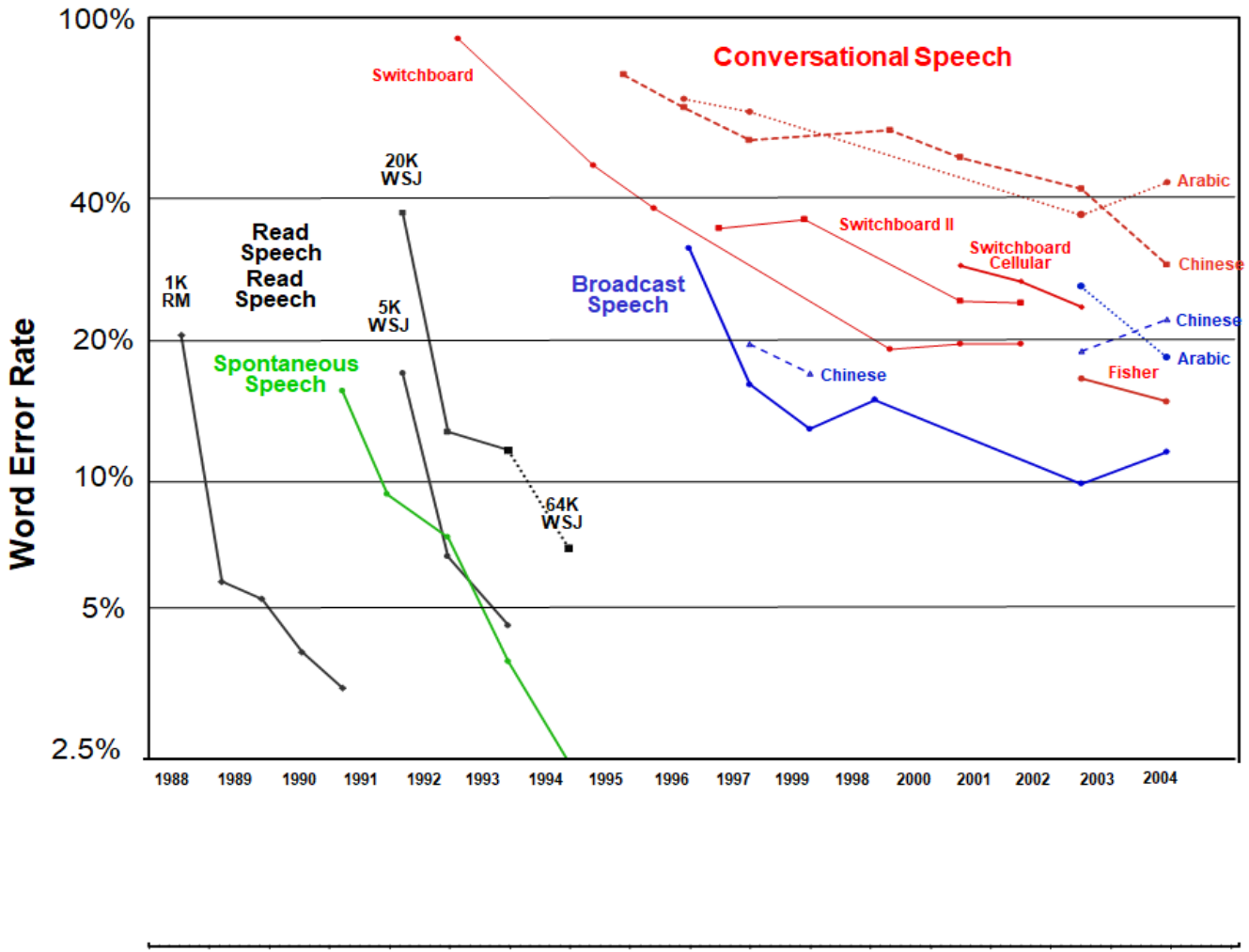
"You can't turn water into gasoline,
no matter what you measure."

Many researchers were disgruntled:

"It's like being in first grade again --
you're told exactly what to do,
and then you're tested over and over ."

But it worked.

Hill-climbing in DARPA Speech-To-Text programs:



Why did it work?

1. The obvious: it allowed funding to start
   (because the projects were glamour-and-deceit-proof)

   and to continue
   (because funders could measure progress over time)

Why did it work?

2. Less obvious: it allowed project-internal hill climbing
   - because the evaluation metrics were automatic
   - and the evaluation code was public

   This obvious way of working was a new idea to many!
   … and researchers who had objected to be tested twice a year began testing themselves every hour…

Why did it work?

3. Even less obvious: it created a culture
   (because researchers shared methods and results
   on shared data with a common metric)

Participation in this culture became so valuable
that many research groups joined without funding

What else it did

The *common task method* created a positive feedback loop.

When everyone's program has to interpret the same ambiguous evidence,
ambiguity resolution becomes a sort of gambling game,
which rewards the use of statistical methods,
and led to the flowering of "machine learning".

Given the nature of speech and language,
statistical methods need the largest possible training set,
which reinforces the value of shared data.

Iterated train-and-test cycles on this gambling game are addictive;
they create "simple, clear, sure knowledge",
which motivates participation in the common-task culture.

The “Common Task Method”

. . . has become the standard research paradigm
in experimental computational science:

- Published training and testing data
- Well-defined evaluation metrics
- Techniques to avoid over-fitting
  (managerial as well as statistical)

Domain: ***Algorithmic analysis of the natural world.***

Over the past 35 years, variants of this method
have been applied to many other problems:

machine translation, speaker identification, language identification, parsing, sense disambiguation, information retrieval, information extraction, summarization, question answering, OCR, sentiment analysis, image analysis, video analysis, … , etc.

The general experience:

1. Error rates decline by a fixed percentage each year,
   to an asymptote depending on task and data quality
2. Progress usually comes from many small improvements;
   improvement by 1% is a reason to break out the champagne.
3. Shared data plays a crucial role – and is re-used in unexpected ways.
4. Glamour and deceit have mostly been avoided.

Four lessons from that experience:

1. Learning is better than programming;
2. Global optimization of gradient local decisions is crucial;
3. Top-down and bottom-up knowledge must be combined;
4. Metrics on shared benchmarks matter.

Versions of the “common task” model are now routinely used,
outside of any sponsored projects,
by both academic and industrial researchers –
and similar “challenges” are routinely created
by technical societies and ad hoc groups.

But there are several important areas that are lagging far behind,
both in technology and in methodology:

- Clinical applications
- Educational applications
- Legal applications

A cultural change in those fields is long overdue!

Recent history:

Transition from statistical modeling of sophisticated features
to “deep learning” models of simpler inputs and outputs.

“Learning is better than programming” –

…but many aspects of early 2000s HLT systems were still “programmed”
via “feature engineering” at both ends
and many structural and algorithmic choices in the middle…

1. Top-down language models rely on combinations of characters
   into ”words” and “phrases”
   with pronunciations given by a dictionary
   and/or by letter-to-sound rules
2. Bottom-up acoustic models rely on sophisticated spectral analysis
3. In the middle, there are many structural and algorithmic choices

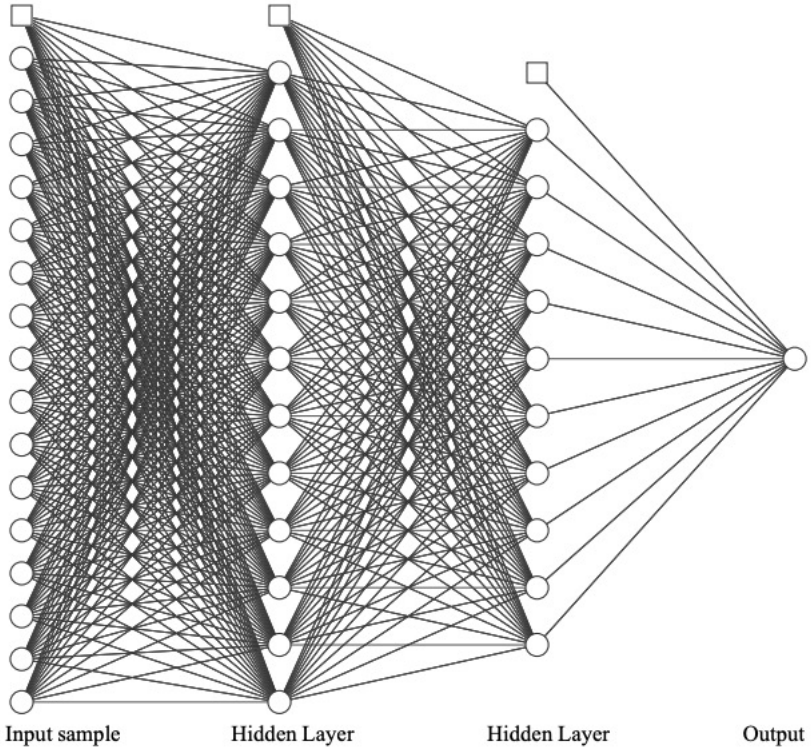
SO…

“Deep Learning” to the rescue!

F(x) = L(N(L(N(…L(x)))))

- where x is an arbitrary vector input
- L(x) is an affine function ax+b
- N(x) is a non-linear function applied to each vector element separately

…plus some other goodies around the edges…



This is a universal computing model,
in the sense that such a system
can be programmed to compute
any finite function.

And even better, general optimization techniques
can learn model parameters from training data.

(…in the limit, sort of, sometimes..)

So we can do away with “feature engineering”
and design a “sequence-to-sequence” model
whose inputs are audio waveform samples
and whose outputs are text characters.

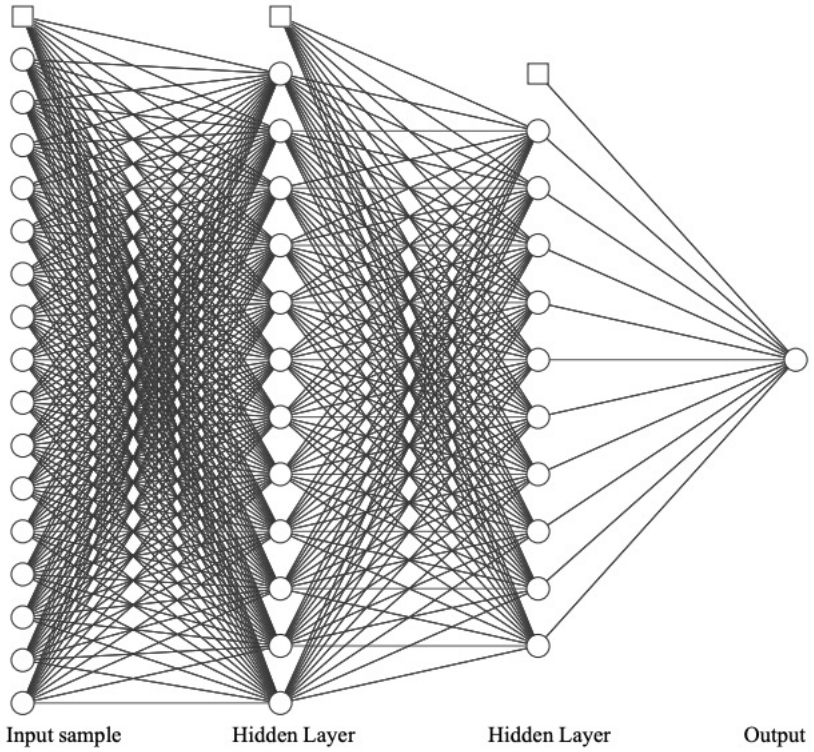
After all, spectral analysis is just a bunch of inner products,
so why not learn the basis functions and band definitions
rather than programming them?

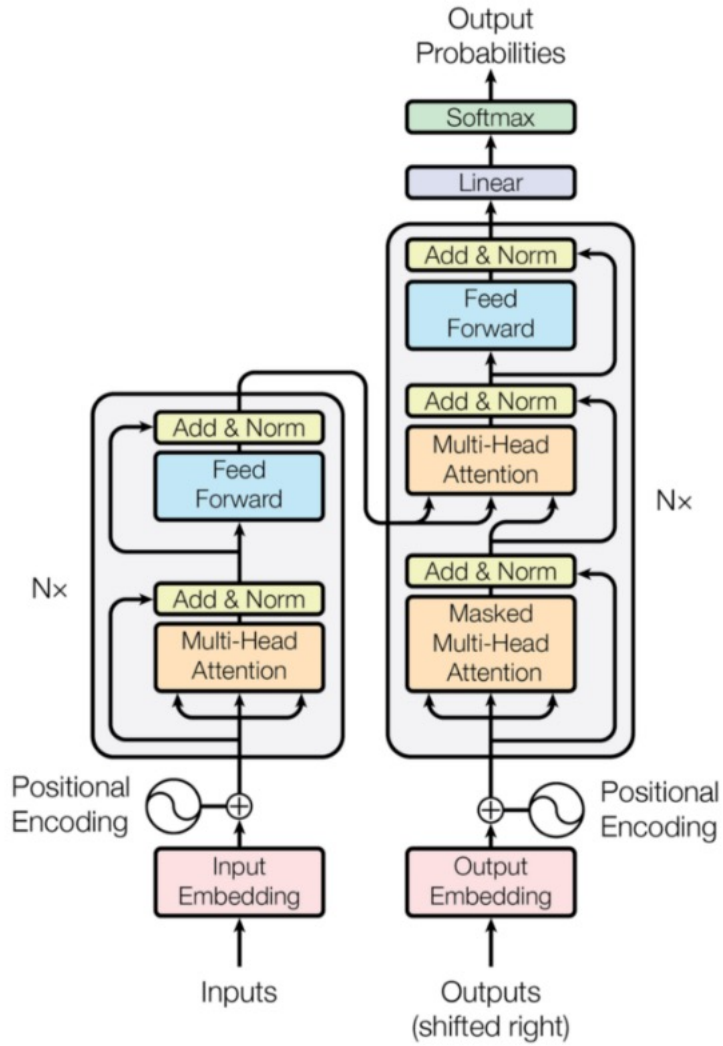
And a more complicated version of the same story
applies to text analysis/synthesis.

Deep Learning solutions do work better –

…but at a cost.

Deep Learning “programs” are increasingly complicated -- CNNs, RNNs, LSTMs, “transformers”:





Avoiding “feature engineering”

vastly increases the number of parameters that need to be learned

and the amount of training data and training time needed to learn them.

And the result is a “black box”,
connecting inputs and outputs
with no ability to justify or explain the mappings.

Which are sometimes bizarre –
see e.g. “Electric Sheep”.

So people are starting to ask…

Why should our systems have to re-learn everything --
logic, mathematics, physics,
acoustics, chemistry, dictionaries, etc. --
all over again
for every new problem?

And we're beginning to see a return to an old idea:
systems that have pieces of relevant science "baked in".

In other words, the old epistemological pendulum
is starting to swing back from empiricism towards rationalism.

Metaphor:
AI programming should be like video game programming.