Introduction to NLP and Career Prospects

(Slides are borrowed from Chris Manning's Stanford CS 224 course)
February 2021

Fatemehsadat Mireshghallah
UC San Diego
fatemeh@ucsd.edu
...anyway, I could care less.

I think you mean you couldn't care less. Saying you could care less implies you care at least some amount.

I dunno.

We're these unbelievably complicated brains drifting through a void, trying in vain to connect with one another by blindly flinging words out into the darkness.

Every choice of phrasing and spelling and tone and timing carries countless signals and contexts and subtexts and more, and every listener interprets those signals in their own way. Language isn't a formal system. Language is glorious chaos.

You can never know for sure what any words will mean to anyone. All you can do is try to get better at guessing how your words affect people, so you can have a chance of finding the ones that will make them feel something like what you want them to feel. Everything else is pointless.

I assume you're giving me tips on how you interpret words because you want me to feel less alone. If so, then thank you. That means a lot.

But if you're just running my sentences past some mental checklist so you can show off how well you know it,

then I could care less.

https://xkcd.com/1576/  Randall Munroe CC BY NC 2.5
Applications of NLP

- Translation
- Error Detection
- Email Classification
- Disease Prediction
- Task Planning
- Fake News Detection
- Email Completion
- Sentiment Analysis
How do we represent the meaning of a word?

**Common solution:** Use e.g. WordNet, a thesaurus containing lists of *synonym sets* and *hypernyms* (“is a” relationships).

**e.g. synonym sets containing “good”:**

```python
from nltk.corpus import wordnet as wn
poses = { 'n': 'noun', 'v': 'verb', 's': 'adj (s)', 'a': 'adj', 'r': 'adv'
for synset in wn.synsets('good'):
    print("{}: {}").format(poses[synset.pos()],
"", ".join([l.name() for l in synset.lemmas()])))
```

```
noun: good
noun: good, goodness
noun: good, goodness
noun: commodity, trade_good, good
adj: good
adj (sat): full, good
adj: good
adj (sat): estimable, good, honorable, respectable
adj (sat): beneficial, good
adj (sat): good
adj (sat): good, just, upright
...
adverb: well, good
adverb: thoroughly, soundly, good
```

**e.g. hypernyms of “panda”:**

```python
from nltk.corpus import wordnet as wn
panda = wn.synset("panda.n.01")
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```
Problems with wordnet

• Great as a resource but missing nuance
  • e.g. “proficient” is listed as a synonym for “good”. This is only correct in some contexts.

• Missing new meanings of words
  • e.g., wicked, badass, nifty, wizard, genius, ninja, bombest
  • Impossible to keep up-to-date!

• Subjective

• Requires human labor to create and adapt
Word Embeddings

In traditional NLP, we regard words as discrete symbols: hotel, conference, motel – a localist representation

Means one 1, the rest 0s

Words can be represented by one-hot vectors:

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
hotel = [0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]

Vector dimension = number of words in vocabulary (e.g., 500,000)
Word Embeddings

**Example:** in web search, if user searches for “Seattle motel”, we would like to match documents containing “Seattle hotel”.

But:

```
motel = [0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
```

These two vectors are orthogonal.

There is no natural notion of **similarity** for one-hot vectors!
Word Embeddings

We will build a dense vector for each word, chosen so that it is similar to vectors of words that appear in similar contexts.

\[
\text{banking} = \begin{pmatrix}
0.286 \\
0.792 \\
-0.177 \\
-0.107 \\
0.109 \\
-0.542 \\
0.349 \\
0.271
\end{pmatrix}
\]

Note: word vectors are sometimes called word embeddings or word representations. They are a distributed representation.
Embedding Visualizations

\[
\text{expect} = \begin{pmatrix}
0.286 \\
0.792 \\
-0.177 \\
-0.107 \\
0.109 \\
-0.542 \\
0.349 \\
0.271 \\
0.487
\end{pmatrix}
\]
Word Embeddings

- Start with random word vectors
- Iterate through each word in the whole corpus
- Try to predict surrounding words using word vectors: 
  \[
  P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}
  \]

- **Learning**: Update vectors so they can predict actual surrounding words better
- Doing no more than this, this algorithm learns word vectors that capture well word similarity and meaningful directions in a wordspace!
Embedding Visualization: Gender
Embedding Visualization: Superlatives
Doctor: No heart, cognitive issues

But Trump needs to reduce his cholesterol, lose weight

By JILL COLVIN
Mutilated body washes up on Rio beach to be used for Olympics beach volleyball
Dependency Parsing

Dependency syntax postulates that syntactic structure consists of relations between lexical items, normally binary asymmetric relations ("arrows") called dependencies.
Language Modeling

- Introduce a new NLP task
  - Language Modeling

- Introduce a new family of neural networks
  - Recurrent Neural Networks (RNNs)
Language Modeling is the task of predicting what word comes next.

the students opened their ________

More formally: given a sequence of words $x^{(1)}, x^{(2)}, \ldots, x^{(t)}$, compute the probability distribution of the next word $x^{(t+1)}$:

$$P(x^{(t+1)} | x^{(t)}, \ldots, x^{(1)})$$

where $x^{(t+1)}$ can be any word in the vocabulary $V = \{w_1, \ldots, w_{|V|}\}$

A system that does this is called a Language Model.
You can also think of a Language Model as a system that assigns probability to a piece of text.

For example, if we have some text $x^{(1)}, \ldots, x^{(T)}$, then the probability of this text (according to the Language Model) is:

$$P(x^{(1)}, \ldots, x^{(T)}) = P(x^{(1)}) \times P(x^{(2)} | x^{(1)}) \times \cdots \times P(x^{(T)} | x^{(T-1)}, \ldots, x^{(1)})$$

$$= \prod_{t=1}^{T} P(x^{(t)} | x^{(t-1)}, \ldots, x^{(1)})$$

This is what our LM provides
Language Modeling

I'll meet you at the cafe, airport, or office.
Language Modeling

what is the weather
what is the meaning of life
what is the dark web
what is the xfl
what is the doomsday clock
what is the weather today
what is the keto diet
what is the american dream
what is the speed of light
what is the bill of rights
Language Modeling: N-grams

*the students opened their _____*

- **Question**: How to learn a Language Model?
- **Answer** (pre- Deep Learning): learn a *n*-gram Language Model!

- **Definition**: A *n*-gram is a chunk of *n* consecutive words.
  - unigrams: “the”, “students”, “opened”, “their”
  - bigrams: “the students”, “students opened”, “opened their”
  - trigrams: “the students opened”, “students opened their”
  - 4-grams: “the students opened their”

- **Idea**: Collect statistics about how frequent different *n*-grams are, and use these to predict next word.
Language Modeling: N-grams

Sparsity Problem 1

**Problem:** What if “students opened their $w$” never occurred in data? Then $w$ has probability 0!

**Partial Solution:** Add small $\delta$ to the count for every $w \in V$. This is called smoothing.

$$P(w|\text{students opened their}) = \frac{\text{count(students opened their $w$)}}{\text{count(students opened their)}}$$

Sparsity Problem 2

**Problem:** What if “students opened their” never occurred in data? Then we can’t calculate probability for any $w$!

**Partial Solution:** Just condition on “opened their” instead. This is called backoff.

**Note:** Increasing $n$ makes sparsity problems worse. Typically we can’t have $n$ bigger than 5.
Language Modeling: RNNs
Language Modeling

- Let’s have some fun!
- You can train a RNN-LM on any kind of text, then generate text in that style.
- RNN-LM trained on Obama speeches:

  The United States will step up to the cost of a new challenges of the American people that will share the fact that we created the problem. They were attacked and so that they have to say that all the task of the final days of war that I will not be able to get this done.
Language Modeling

• Let’s have some fun!
• You can train a RNN-LM on any kind of text, then generate text in that style.
• RNN-LM trained on *Harry Potter*:

  “Sorry,” Harry shouted, panicking—“I’ll leave those brooms in London, are they?”

  “No idea,” said Nearly Headless Nick, casting low close by Cedric, carrying the last bit of treacle Charms, from Harry’s shoulder, and to answer him the common room perched upon it, four arms held a shining knob from when the spider hadn’t felt it seemed. He reached the teams too.

Language Modeling

• Let’s have some fun!
• You can train a RNN-LM on any kind of text, then generate text in that style.
• RNN-LM trained on recipes:

  Title: CHOCOLATE RANCH BARBECUE
  Categories: Game, Casseroles, Cookies, Cookies
  Yield: 6 Servings
  2 tb Parmesan cheese — chopped
  1 c Coconut milk
  3 Eggs, beaten

  Place each pasta over layers of lumps. Shape mixture into the moderate oven and simmer until firm. Serve hot in bodied fresh, mustard, orange and cheese.

  Combine the cheese and salt together the dough in a large skillet; add the ingredients and stir in the chocolate and pepper.

Source: https://gist.github.com/nylki/1efbaa36635956d35bcc
**Machine Translation**

*Machine Translation (MT)* is the task of translating a sentence $x$ from one language (the source language) to a sentence $y$ in another language (the target language).

$x: \quad L\'homme est né libre, et partout il est dans les fers$

$\downarrow$

$y: \quad Man \ is \ born \ free, \ but \ everywhere \ he \ is \ in \ chains$

- Rousseau
Statistical Machine Translation

- **Core idea**: Learn a **probabilistic model from data**
- Suppose we’re translating French → English.
- We want to find **best English sentence** \( y \), **given French sentence** \( x \)

\[
\arg\max_y P(y|x)
\]

- Use Bayes Rule to break this down into **two components** to be learnt separately:

\[
= \arg\max_y P(x|y) P(y)
\]

**Translation Model**
Models how words and phrases should be translated (**fidelity**). Learnt from parallel data.

**Language Model**
Models how to write good English (**fluency**). Learnt from monolingual data.
Statistical Machine Translation

- SMT was a huge research field
- The best systems were extremely complex
  - Hundreds of important details we haven’t mentioned here
  - Systems had many separately-designed subcomponents
- Lots of feature engineering
  - Need to design features to capture particular language phenomena
- Require compiling and maintaining extra resources
  - Like tables of equivalent phrases
- Lots of human effort to maintain
  - Repeated effort for each language pair!
2014

Neural Machine Translation

MT research

(dramatic reenactment)
Neural Machine Translation (NMT)

The sequence-to-sequence model

Encoding of the source sentence.
Provides initial hidden state for Decoder RNN.

Source sentence (input)

Decoder RNN is a Language Model that generates target sentence, conditioned on encoding.

Note: This diagram shows test time behavior: decoder output is fed in \( \cdots \rightarrow \) as next step’s input.

Encoder RNN produces an encoding of the source sentence.
Sequence-to-sequence is versatile!

- Sequence-to-sequence is useful for *more than just MT*

- Many NLP tasks can be phrased as sequence-to-sequence:
  - *Summarization* (long text → short text)
  - *Dialogue* (previous utterances → next utterance)
  - *Parsing* (input text → output parse as sequence)
  - *Code generation* (natural language → Python code)
Advantages of Neural Machine Translation

Compared to SMT, NMT has many advantages:

- Better performance
  - More fluent
  - Better use of context
  - Better use of phrase similarities

- A single neural network to be optimized end-to-end
  - No subcomponents to be individually optimized

- Requires much less human engineering effort
  - No feature engineering
  - Same method for all language pairs
Neural Machine Translation

**Disadvantages of NMT?**

Compared to SMT:

- NMT is *less interpretable*
  - Hard to debug

- NMT is *difficult to control*
  - For example, can’t easily specify rules or guidelines for translation
  - Safety concerns!
So is Machine Translation solved?

- **Nope!**
- Using *common sense* is still hard

```
paper jam Edit
```

```
Mermelada de papel
```

![Image of paper jam and marmalade jar]
So is Machine Translation solved?

- **Nope!**
- NMT picks up biases in training data

Source: [https://hackernoon.com/bias-sexist-or-this-is-the-way-it-should-be-ce1f7c8c683c](https://hackernoon.com/bias-sexist-or-this-is-the-way-it-should-be-ce1f7c8c683c)
So is Machine Translation solved?

- Nope!
- Uninterpretable systems do strange things
Attention

- **Attention** provides a solution to the bottleneck problem.

- **Core idea**: on each step of the decoder, use *direct connection to the encoder* to focus on a particular part of the source sequence.

- First we will show via diagram (no equations), then we will show with equations.
Self-Attention

Who
Did what?

I kicked the ball

To whom?
Attention head: Who
Attention head: Did What?

Who

Did what?

I

kicked

the

ball
Importance of residuals

Figure 1: The Transformer - model architecture.
Let’s scale it up!

<table>
<thead>
<tr>
<th>Model</th>
<th>Date</th>
<th>Training Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ULMfit</td>
<td>Jan 2018</td>
<td>1 GPU day</td>
</tr>
<tr>
<td>GPT</td>
<td>June 2018</td>
<td>240 GPU days</td>
</tr>
<tr>
<td>BERT</td>
<td>Oct 2018</td>
<td>256 TPU days</td>
</tr>
<tr>
<td>GPT-2</td>
<td>Feb 2019</td>
<td>~2048 TPU v3 days according to a <a href="https://www.reddit.com">reddit thread</a></td>
</tr>
</tbody>
</table>

Logos: fast.ai, OpenAI, Google AI
Transformer models

All of these models are Transformer architecture models ... so maybe we had better learn about Transformers?

ULMfit
Jan 2018
Training:
1 GPU day

GPT
June 2018
Training
240 GPU days

BERT
Oct 2018
Training
256 TPU days
~320–560 GPU days

GPT-2
Feb 2019
Training
~2048 TPU v3
days according to
a reddit thread

BERT (Bidirectional Encoder Representations from Transformers):
Pre-training of Deep Bidirectional Transformers for Language Understanding

Based on slides from Jacob Devlin

- **Problem**: Language models only use left context *or* right context, but language understanding is bidirectional.
- Why are LMs unidirectional?
- Reason 1: Directionality is needed to generate a well-formed probability distribution.
  - We don’t care about this.
- Reason 2: Words can “see themselves” in a bidirectional encoder.

- **Solution**: Mask out $k\%$ of the input words, and then predict the masked words
  - They always use $k = 15\%$

```
store  gallon
↑     ↑
the man went to the [MASK] to buy a [MASK] of milk
```

- Too little masking: Too expensive to train
- Too much masking: Not enough context

ELMo

BERT (Ours)

OpenAI GPT
Careers

- Microsoft: MS office (parsing, prediction, suggestions), Outlook (Todo finding, next word prediction), Medical partners (Prescriptions, analysis, chatbots), Task managers, Swift key
- Google: Gmail (smart compose, spam detection, priority sorting), Gboard (next word prediction), Google news (summarization and recommendation), Google search QA
- Twitter: Sorting timeline, hate speech detection, bot detection, misinformation and fact-checking, news suggestions
- Many startups: Verneek, Grammarly, Duolingo, Hugging Face ...
Thank you!