CS546: Machine Learning in NLP (Spring 2020)
http://courses.engr.illinois.edu/cs546/

Lecture 1 Introduction/Admin

Julia Hockenmaier
juliahmr@illinois.edu
3324 Siebel Center
Office hours: Monday, 11am—12:30pm

Welcome to CS546!

Julia Hockenmaier (Instructor)

juliahmr@illinois.edu

Office hours: Monday, 11am—12:30pm, 3324 Siebel

Zhenbang Wang (TA)

zw11@illinois.edu

Office hours: TBD

Class website:

https://courses.grainger.illinois.edu/cs546

What will you learn in this class?

CS546: Machine Learning in NLP

Questions you should be able to answer after CS546:

What Machine Learning (ML) techniques and tools work well for which Natural Language Processing (NLP) tasks?

What are the challenges in applying ML to NLP tasks?

What we're aiming to cover in CS546 this year:

Focus on neural approaches ("deep learning") to NLP

Background and current research

Overview of different types of neural models and NLP tasks

What you need to do in CS546:

Read, present and discuss research paper(s)

Do a research project

Prerequisites

CS447 Introduction to NLP (or equivalent)

Basic understanding of NLP tasks and models

CS446 Machine Learning (or equivalent)

Basic understanding of ML

Python programming

Most neural network toolkits use it (Tensorflow, Pytorch)

How will we run this class?

This class consists of…

… lectures

Wednesdays/Fridays, 3:30-4:45, DCL1310

Many of these will be paper presentations by students

… office hours

TA office hours are intended for hands-on help with projects

My office hours are mainly intended for paper presentations

… research projects

These can be done in groups of up to four students

… a Compass page

For grades and to submit reports and paper reviews

… a Piazza page

For discussions and to find teammates for projects

… a website https://courses.grainger.illinois.edu/cs546

For slides, syllabus etc.

Assessment

Your grade will consist of

… 35%: your presentation of a research paper in class

… 50%: your research project

… 10%: your written reviews of research papers
(graded mostly for completion)

… 5%: your participation in class

Paper presentations

Everybody needs to prepare a 15-minute oral presentation and a two-page writeup about one research paper to be shared with the class.

NB: This paper shouldn't come from your own research group, nor can it be a paper you presented in your qualifying exam.

- We will send out a sign-up sheet with dates and papers for each class.
- You will have to come to my office hours *the Monday of the week when you're presenting* with your slides to show them to me, otherwise you will only get half credit for your presentation.
- You have one week after your presentation to send in your writeup (so that you can reflect any in-class discussion)

Short Paper Reviews

For 10 lectures where papers are discussed, you will have to submit a review of one of the papers that was discussed in class.

—Due to the size of the class, we can largely grade you for completion (although we will spot-check your answers)
— You will have to submit the reviews through Compass.
— In the past, we've used a LaTeX template for this, but we may switch to tests inside Compass

We encourage you to get into the habit of taking notes about the papers you read. Hopefully this will get you started!

Research projects

You will have to complete a sizable research project.

Due to the size of the class, you will have to work in groups (we're aiming for 3–4 students/team).

There will be several milestones:
— Initial proposal
— Intermediate report and presentation
— Final report and presentation

We have applied for accounts and GPU hours on BlueWaters for these projects.

Research projects

The aim is for each team to produce something that could be submitted to a conference:

— You should aim to make an actual contribution to research
— Your presentation should be sufficiently polished

If you build on existing research, talk to me, and loop your advisor in as well if necessary.

If you're doing related projects in other classes, let me and the other professor know.

DRES accommodations

If you need any disability related accommodations, talk to DRES (http://disability.illinois.edu, disability@illinois.edu, phone 333-4603)

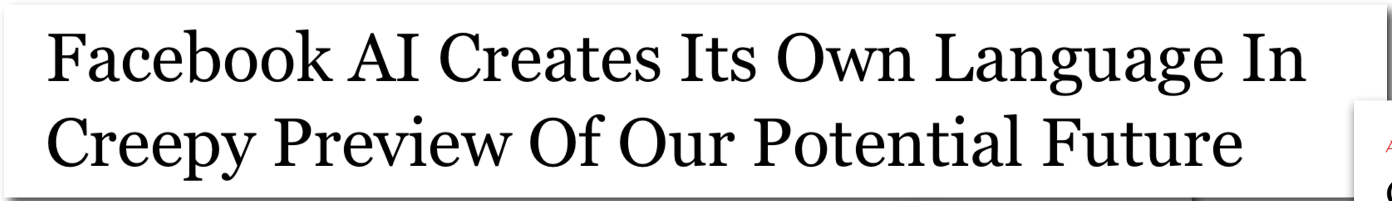
- If you are concerned you have a disability-related condition that is impacting your academic progress, there are academic screening appointments available on campus that can help diagnosis a previously undiagnosed disability by visiting the DRES website and selecting "Sign-Up for an Academic Screening" at the bottom of the page."

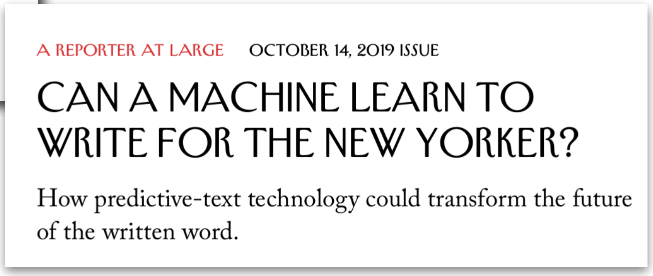
Come and talk to me as well, especially once you have a letter of accommodation from DRES.

- Do this early enough so that we can take your requirements into account

What is NLP?

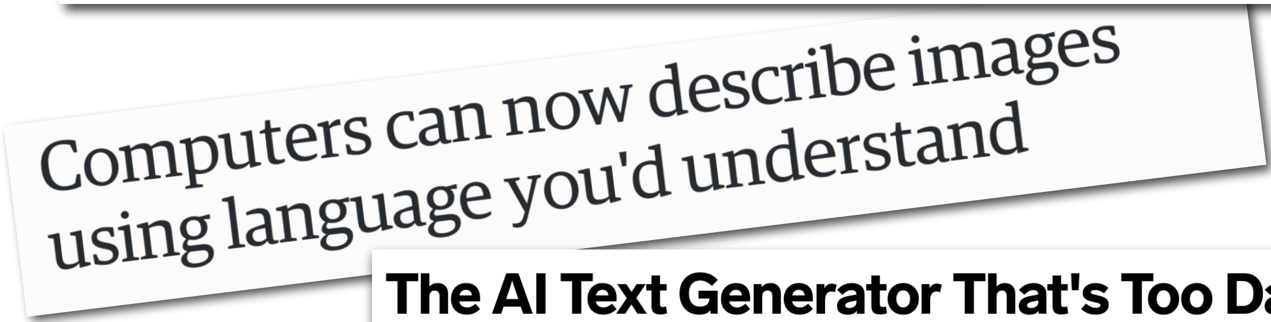
NLP in the news…

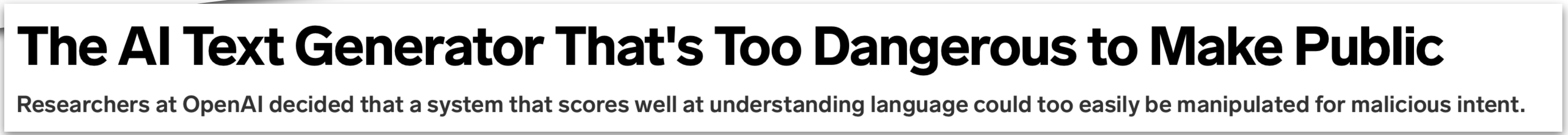

Facebook AI Creates Its Own Language In Creepy Preview Of Our Potential Future


A REPORTER AT LARGE OCTOBER 14, 2019 ISSUE

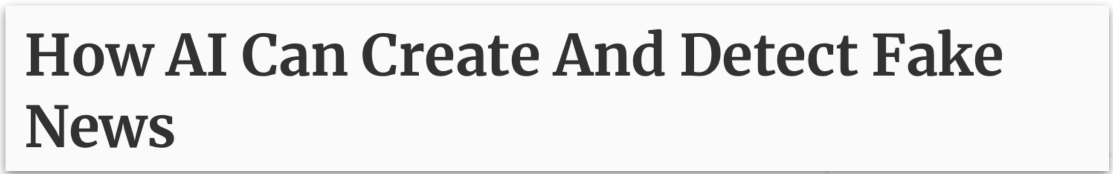
CAN A MACHINE LEARN TO WRITE FOR THE NEW YORKER?

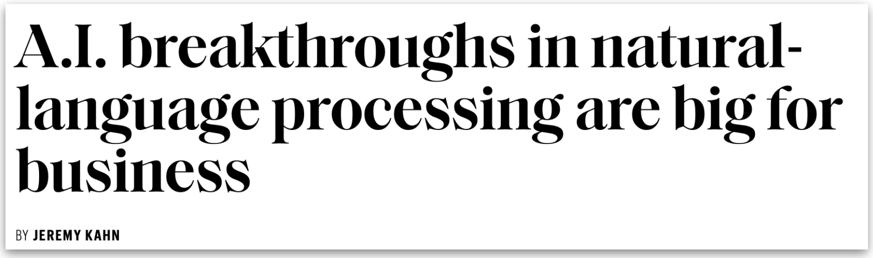
How predictive-text technology could transform the future of the written word.


Computers can now describe images using language you'd understand

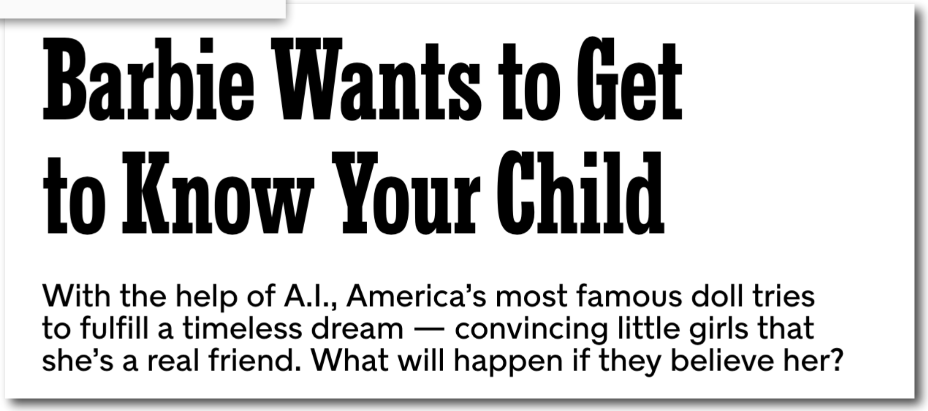

The AI Text Generator That's Too Dangerous to Make Public

Researchers at OpenAI decided that a system that scores well at understanding language could too easily be manipulated for malicious intent.


How AI Can Create And Detect Fake News


A.I. breakthroughs in natural-language processing are big for business

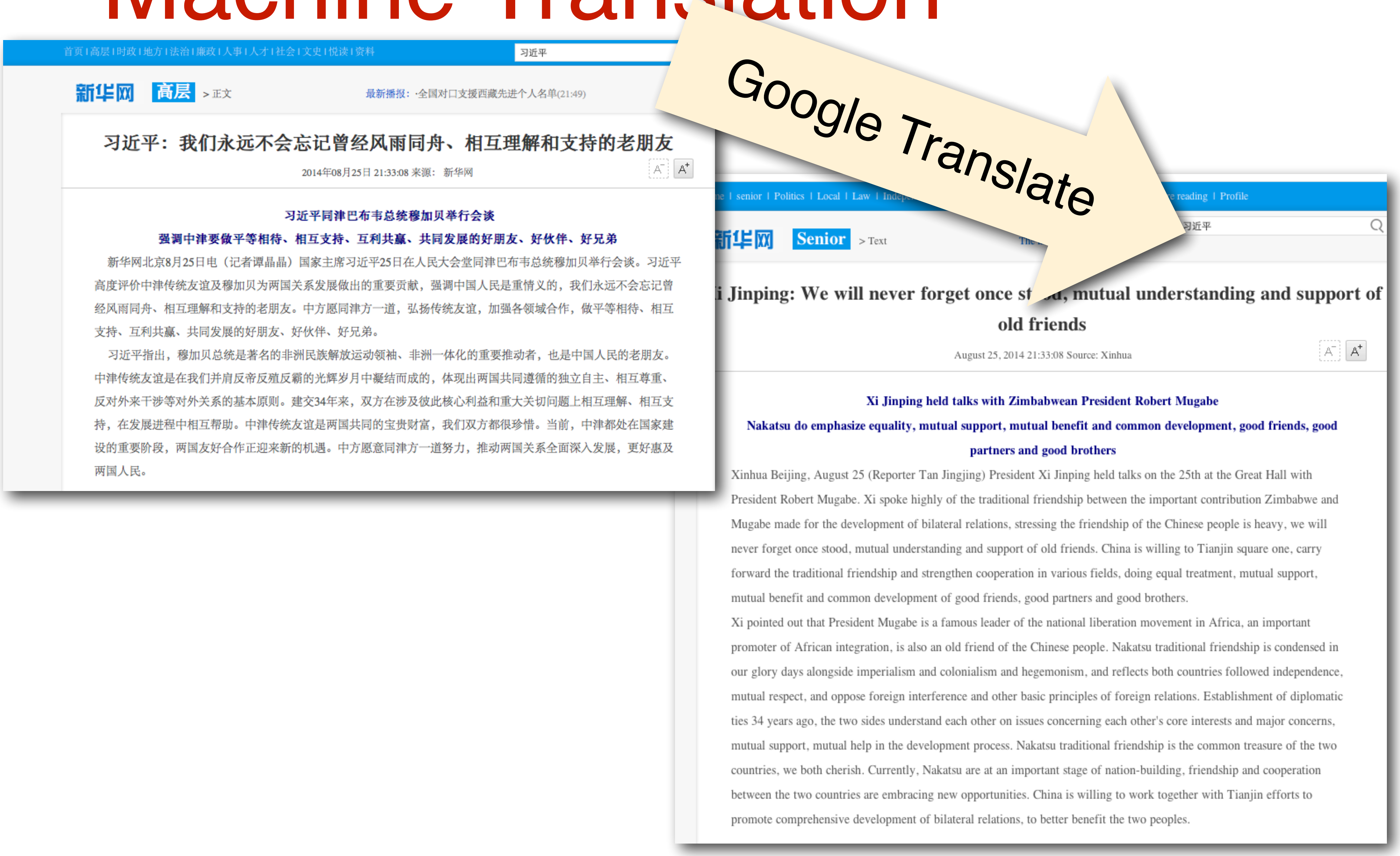
BY JEREMY KAHN


Barbie Wants to Get to Know Your Child

With the help of A.I., America's most famous doll tries to fulfill a timeless dream — convincing little girls that she's a real friend. What will happen if they believe her?

IBM's Watson wins at Jeopardy!

Machine Translation



首页 | 高层 | 时政 | 地方 | 法治 | 廉政 | 人事 | 人才 | 社会 | 文史 | 悦读 | 资料

习近平

新华网 高层 > 正文

最新播报: ·全国对口支援西藏先进个人名单(21:49)

习近平：我们永远不会忘记曾经风雨同舟、相互理解和支持的老朋友

2014年08月25日 21:33:08 来源： 新华网

习近平同津巴布韦总统穆加贝举行会谈

强调中津要做平等相待、相互支持、互利共赢、共同发展的好朋友、好伙伴、好兄弟

新华网北京8月25日电（记者谭晶晶）国家主席习近平25日在人民大会堂同津巴布韦总统穆加贝举行会谈。习近平高度评价中津传统友谊及穆加贝为两国关系发展做出的重要贡献，强调中国人民是重情义的，我们永远不会忘记曾经风雨同舟、相互理解和支持的老朋友。中方愿同津方一道，弘扬传统友谊，加强各领域合作，做平等相待、相互支持、互利共赢、共同发展的好朋友、好伙伴、好兄弟。

习近平指出，穆加贝总统是著名的非洲民族解放运动领袖、非洲一体化的重要推动者，也是中国人民的老朋友。中津传统友谊是在我们并肩反帝反殖反霸的光辉岁月中凝结而成的，体现出两国共同遵循的独立自主、相互尊重、反对外来干涉等对外关系的基本原则。建交34年来，双方在涉及彼此核心利益和重大关切问题上相互理解、相互支持，在发展进程中相互帮助。中津传统友谊是两国共同的宝贵财富，我们双方都很珍惜。当前，中津都处在国家建设的重要阶段，两国友好合作正迎来新的机遇。中方愿意同津方一道努力，推动两国关系全面深入发展，更好惠及两国人民。

senior | Politics | Local | Law | Indep

reading | Profile

习近平

新华网 Senior > Text

i Jinping: We will never forget once st , mutual understanding and support of old friends

August 25, 2014 21:33:08 Source: Xinhua

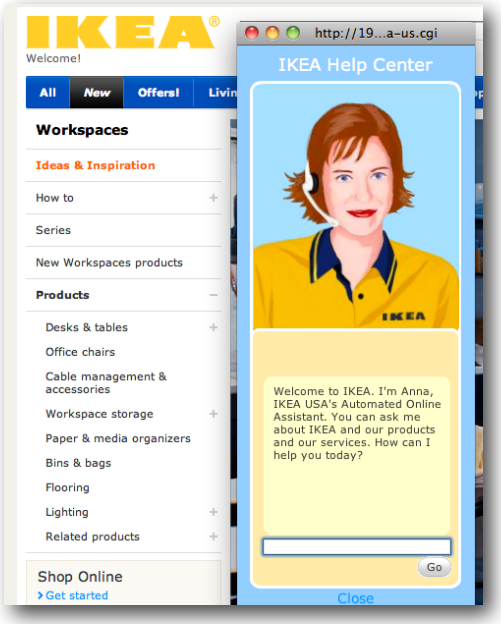
Xi Jinping held talks with Zimbabwean President Robert Mugabe

Nakatsu do emphasize equality, mutual support, mutual benefit and common development, good friends, good partners and good brothers

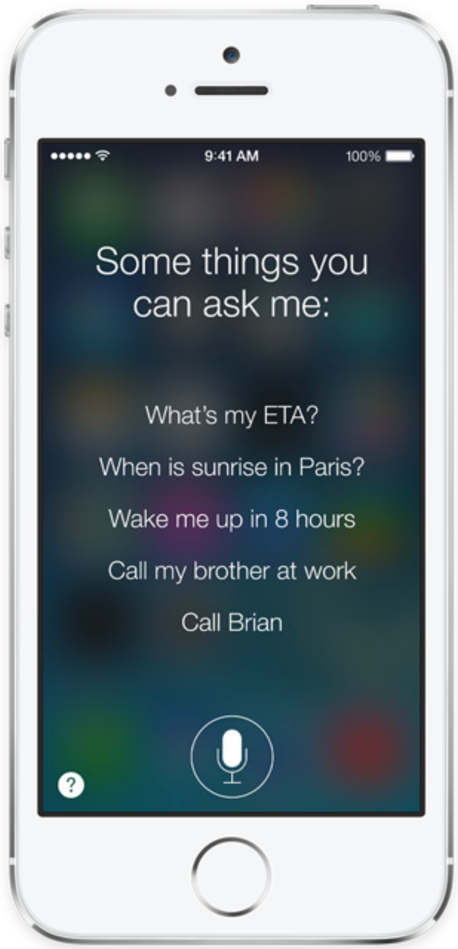
Xinhua Beijing, August 25 (Reporter Tan Jingjing) President Xi Jinping held talks on the 25th at the Great Hall with President Robert Mugabe. Xi spoke highly of the traditional friendship between the important contribution Zimbabwe and Mugabe made for the development of bilateral relations, stressing the friendship of the Chinese people is heavy, we will never forget once stood, mutual understanding and support of old friends. China is willing to Tianjin square one, carry forward the traditional friendship and strengthen cooperation in various fields, doing equal treatment, mutual support, mutual benefit and common development of good friends, good partners and good brothers.

Xi pointed out that President Mugabe is a famous leader of the national liberation movement in Africa, an important promoter of African integration, is also an old friend of the Chinese people. Nakatsu traditional friendship is condensed in our glory days alongside imperialism and colonialism and hegemonism, and reflects both countries followed independence, mutual respect, and oppose foreign interference and other basic principles of foreign relations. Establishment of diplomatic ties 34 years ago, the two sides understand each other on issues concerning each other's core interests and major concerns, mutual support, mutual help in the development process. Nakatsu traditional friendship is the common treasure of the two countries, we both cherish. Currently, Nakatsu are at an important stage of nation-building, friendship and cooperation between the two countries are embracing new opportunities. China is willing to work together with Tianjin efforts to promote comprehensive development of bilateral relations, to better benefit the two peoples.

Dialog systems, chatbots, digital assistants



Welcome to IKEA. I'm Anna, IKEA USA's Automated Online Assistant. You can ask me about IKEA and our products and our services. How can I help you today?



Some things you can ask me:

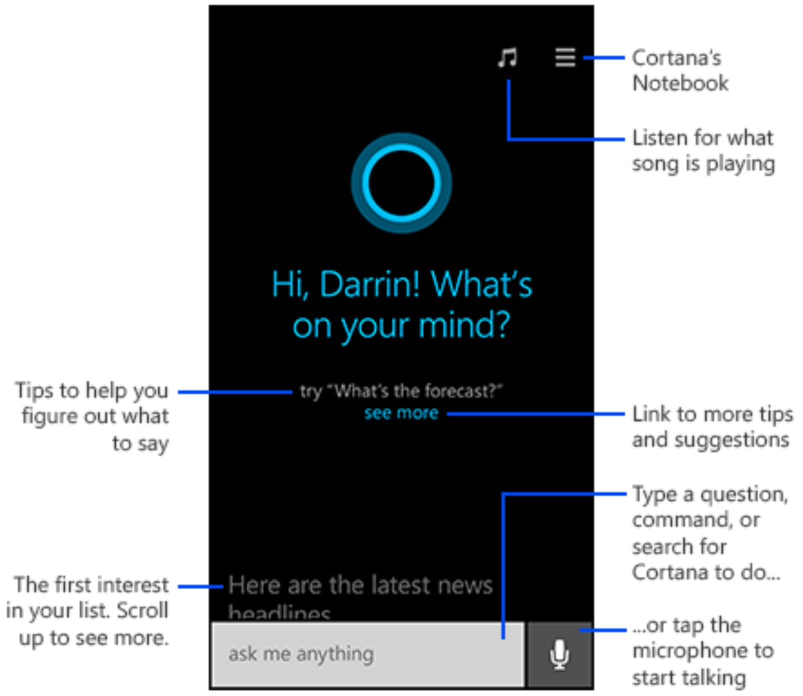
What's my ETA?

When is sunrise in Paris?

Wake me up in 8 hours

Call my brother at work

Call Brian


Cortana's Notebook
Listen for what song is playing
Hi, Darrin! What's on your mind?
Tips to help you figure out what to say
try "What's the forecast?"
see more
Link to more tips and suggestions
Type a question, command, or search for Cortana to do...
The first interest in your list. Scroll up to see more.
Here are the latest news headlines
ask me anything
...or tap the microphone to start talking

ELIZA: a computer therapist from 1966

Computer: TELL ME MORE ABOUT YOUR FAMILY

Human: My mother takes care of me.

Computer: WHO ELSE IN YOUR FAMILY TAKES CARE OF YOU

Human: My father.

Computer: YOUR FATHER

Human: You are like my father in some ways.

Computer: WHAT RESEMBLANCE DO YOU SEE

Human: You are not very aggressive but I think you don't want me to notice that.

Computer: WHAT MAKES YOU THINK I AM NOT VERY AGGRESSIVE

Human: You don't argue with me.

Computer: WHY DO YOU THINK I DON'T ARGUE WITH YOU

Human: You are afraid of me.

Weizenbaum (1966), ELIZA.

The ELIZA effect

People *really* trusted this system:

Weizenbaum: *“I had not realized ... that extremely short exposures to a relatively simple computer program could induce powerful delusional thinking in quite normal people”*

ELIZA was based on a cascade of finite-state transducers:

Human: *You don't argue with me.*

1. Replace **you** with ***I*** and ***me*** with ***you:***

   I don't argue with you.

2. Replace ***<...>*** with ***Why do you think <...>:***

   Why do you think I don't argue with you.

Computer: *WHY DO YOU THINK I DON'T ARGUE WITH YOU*

— How could you break a system like ELIZA?

— Where do you see the ELIZA effect today?

Have chatbots developed their own language?

https://code.facebook.com/posts/1686672014972296/deal-or-no-deal-training-ai-bots-to-negotiate/

Bob: “I can can I I everything else.”
Alice: “Balls have zero to me to me to me to me to me to me to me to me to.”

No, not really…
http://www.snopes.com/facebook-ai-developed-own-language/

How well does that Barbie do?

Barbie: *“Do you have any sisters?”*
Child: “Yeah, I only have one.”
Barbie: *“What’s something nice that your sister does for you?”*
Child: “She does nothing nice to me”
Barbie: *“Well, what is the last nice thing your sister did?”*
Child: “She helped me with my project
— and then she destroyed it.”
Barbie: *“Oh, yeah, tell me more!”*
Child: “That’s it, Barbie,”
Barbie: *“Have you told your sister lately how cool she is?”*
Child: “No. She is not cool,”
Barbie: *“You never know, she might appreciate hearing it”*

(Thanks to Barbara Grosz for pointing out the example from the NYT)
https://www.nytimes.com/2015/09/20/magazine/barbie-wants-to-get-to-know-your-child.html

What is the current state of NLP?

Lots of commercial applications and interest.

Some applications are working pretty well already, others not so much.

A lot of hype around “deep learning” and “AI”

- Neural nets are powerful classifiers and sequence models
- Public libraries (Tensorflow, Pytorch, etc..) and datasets make it easy for anybody to get a model up and running
- “End-to-end” models put into question whether we still need the traditional NLP pipeline that this class is built around
- We’re still in the middle of this paradigm shift
- But many of the fundamental problems haven’t gone away

Examples of NLP applications

(What can NLP be used for?)

Natural language (and speech) interfaces

Search/IR, database access, image search, image description

Dialog systems (e.g. customer service, robots, cars, tutoring), chatbots

Information extraction, summarization, translation

Process (large amounts of) text automatically to obtain meaning/knowledge contained in the text

Translate text automatically from one language to another

Convenience, social science

Grammar/style checking, automate email filing, autograding

Identify/analyze trends, opinions, etc. (e.g. in social media)

Examples of NLP tasks
(What capabilities do NLP systems need?)

Natural language understanding
Extract information (e.g. about entities or events) from text
Translate raw text into a meaning representation
Reason about information given in text
Execute NL instructions

Natural language generation and summarization
Translate database entries or meaning representations to raw natural language text
Produce (appropriate) utterances/responses in a dialog
Summarize (newspaper or scientific) articles, describe images

Natural language translation
Translate one natural language to another

The NLP (NLU) Pipeline

A (traditional) NLP system may use some or all of the following steps:

Tokenizer/Segmenter
  to identify words and sentences

Morphological analyzer/POS-tagger
  to identify the part of speech and structure of words

Word sense disambiguation
  to identify the meaning of words

Syntactic/semantic Parser
  to obtain the structure and meaning of sentences

Coreference resolution/discourse model
  to keep track of the various entities and events mentioned

NLP Pipeline: Assumptions

Each step in the NLP pipeline embellishes the input with **explicit information** about its linguistic structure

POS tagging: parts of speech of word,

Syntactic parsing: grammatical structure of sentence,….

Each step in the NLP pipeline requires its own explicit **(“symbolic”) output representation:**

POS tagging requires a POS tag set

(e.g. NN=common noun singular, NNS = common noun plural, …)

Syntactic parsing requires constituent or dependency labels

(e.g. NP = noun phrase, or nsubj = nominal subject)

These representations should capture linguistically appropriate **generalizations/abstractions**

Designing these representations requires linguistic expertise

NLP Pipeline: Shortcomings

Each step in the pipeline relies on a **learned model** that will return the *most likely* representations

- This requires a lot of **annotated training data** for each step
- Annotation is **expensive** and sometimes **difficult** (people are not 100% accurate)
- These models are **never 100% accurate**
- Models make more mistakes if their input contains mistakes

How do we know that we have captured the "***right***" **generalizations** when designing representations?

- Some representations are **easier to predict** than others
- Some representations are **more useful** for the next steps in the pipeline than others
- But we won't know how easy/useful a representation is until we have a model that we can plug into a particular pipeline

NLP research questions redux

How do you represent (or predict) words?

Do you treat words in the input as atomic categories, as continuous vectors, or as structured objects?

How do you handle rare/unseen words, typos, spelling variants, morphological information?

Lexical semantics: do you capture word meanings/senses?

How do you represent (or predict) word sequences?

Sequences = sentences, paragraphs, documents, dialogs,…

As a vector, or as a structured object?

How do you represent (or predict) structures?

Structures = labeled sequences, trees, graphs, formal languages (e.g. DB records/queries, logical representations)

How do you represent “meaning”?

Why does NLP need ML?

Two core problems for NLP

Ambiguity: Natural language is highly ambiguous

- Words have multiple senses and different POS
- Sentences have a myriad of possible parses
- etc.

Coverage (compounded by Zipf's Law)

- Any (wide-coverage) NLP system will come across words or constructions that did not occur during training.
- We need to be able to generalize from the seen events during training to unseen events that occur during testing (i.e. when we actually use the system).

Statistical models for NLP

NLP makes heavy use of statistical models as a way to handle both the ambiguity and the coverage issues.

- Probabilistic models (e.g. HMMs, MEMMs, CRFs, PCFGs)
- Other machine learning-based classifiers

Basic approach:

- Decide which output is desired
  (may depend on available labeled training data)
- Decide what kind of model to use
- Define features that could be useful (this may require further processing steps, i.e. a pipeline)
- Train and evaluate the model.
- Iterate: refine/improve the model and/or the features, etc.

Example: Language Modeling

A language model defines a distribution $P(\mathbf{w})$ over the strings $\mathbf{w} = w_1 w_2 .. w_i \ldots$ in a language

Typically we factor $P(\mathbf{w})$ such that we compute the probability word by word:

$$P(\mathbf{w}) = P(w_1)\, P(w_2 \mid w_1) \ldots P(w_i \mid w_1 \ldots w_{i-1})$$

Standard n-gram models make the Markov assumption that w_i depends only on the preceding n–1 words:

$$P(w_i \mid w_1 \ldots w_{i-1}) := P(w_i \mid w_{i-n+1} \ldots w_{i-1})$$

We know that this independence assumption is invalid (there are many long-range dependencies), but it is computationally and statistically necessary

(we can't store or estimate larger models)

Motivation for neural approaches to NLP: Markov assumptions

Traditional sequence models (n-gram language models, HMMs, MEMMs, CRFs) make rigid Markov assumptions (bigram/trigram/n-gram).

Recurrent neural nets (RNNs, LSTMs) and transformers can capture arbitrary-length histories without requiring more parameters.

Features in statistical NLP

Many statistical NLP systems use explicit features:

- Words (does the word “river” occur in this sentence?)
- POS tags
- Chunk information, NER labels
- Parse trees or syntactic dependencies
  (e.g. for semantic role labeling, etc.)

Feature design is usually a big component of building any particular NLP system.

Which features are useful for a particular task and model typically requires experimentation, but there are a number of commonly used ones (words, POS tags, syntactic dependencies, NER labels, etc.)

Motivation for neural approaches to NLP: Features can be brittle

Word-based features:

How do we handle unseen/rare words?

Many features are produced by other NLP systems (POS tags, dependencies, NER output, etc.)

These systems are often trained on labeled data.

Producing labeled data can be very expensive.

We typically don't have enough labeled data from the domain of interest.

We might not get accurate features for our domain of interest.

Features in neural approaches

Many of the current successful neural approaches to NLP do not use traditional discrete features.
— End-to-end models: no pipeline!

Words in the input are often represented as **dense vectors** (aka. word embeddings, e.g. word2vec)

Traditional approaches: each word in the vocabulary is a separate feature. No generalization across words that have similar meanings.

Neural approaches: Words with similar meanings have similar vectors. Models generalize across words with similar meanings

Other kinds of features (POS tags, dependencies, etc.) **are often ignored**.

Neural approaches to NLP

What is “deep learning”?

Neural networks, typically with several hidden layers

(depth = # of hidden layers)

Single-layer neural nets are linear classifiers

Multi-layer neural nets are more expressive

Very impressive performance gains in computer vision (ImageNet) and speech recognition over the last several years.

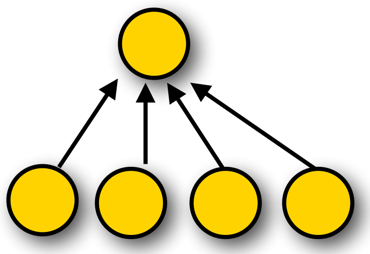
Neural nets have been around for decades.
Why have they suddenly made a comeback?

Fast computers (GPUs!) and (very) large datasets have made it possible to train these very complex models.

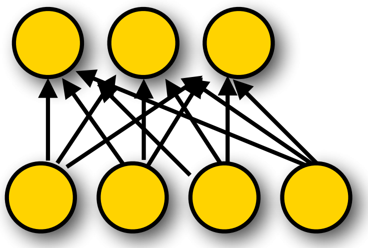
What are neural nets?

Simplest variant: single-layer feedforward net

For binary classification tasks:
Single output unit
Return 1 if $y > 0.5$
Return 0 otherwise



For multiclass classification tasks:
K output units (a vector)
Each output unit
y_i = class i
Return $\text{argmax}_i(y_i)$



Multiclass models: $\mathrm{softmax}(y_i)$

Multiclass classification = predict one of K classes.

Return the class i with the highest score: $\mathrm{argmax}_i(y_i)$

In neural networks, this is typically done by using the **softmax** function, which maps real-valued vectors in R^N into a distribution over the N outputs

For a vector $\mathbf{z} = (z_0 \ldots z_K)$: $P(i) = \mathrm{softmax}(z_i) = \exp(z_i) \;/\; \sum_{k=0..K} \exp(z_k)$

(NB: This is just logistic regression)

Single-layer feedforward networks

Single-layer (linear) feedforward network

$y = \mathbf{wx} + b$ (binary classification)

$\mathbf{w}$ is a weight vector, b is a bias term (a scalar)

This is just a linear classifier (aka Perceptron)

(the output y is a linear function of the input $\mathbf{x}$)

Single-layer non-linear feedforward networks:

Pass $\mathbf{wx} + b$ through a non-linear activation function,

e.g. $y = \tanh(\mathbf{wx} + b)$

Nonlinear activation functions

Sigmoid (logistic function): $\sigma(x) = 1/(1 + e^{-x})$

Useful for output units (probabilities) [0,1] range

Hyperbolic tangent: $\tanh(x) = (e^{2x} - 1)/(e^{2x}+1)$

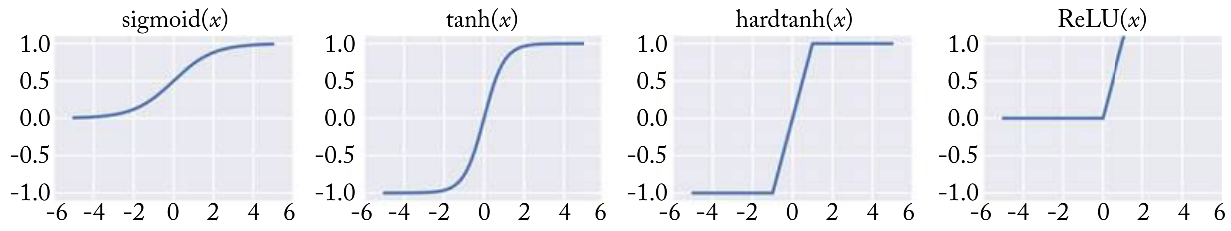
Useful for internal units: [-1,1] range

Hard tanh (approximates tanh)

$\text{htanh}(x) = -1 \text{ for } x < -1,\ 1 \text{ for } x > 1,\ x \text{ otherwise}$

Rectified Linear Unit: $\text{ReLU}(x) = \max(0, x)$

Useful for internal units

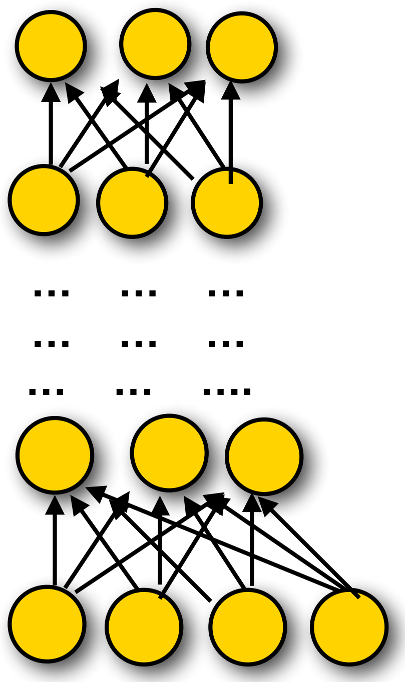


Softmax: $\text{softmax}(z_i) = \exp(z_i) \ / \ \sum_{k=0..K} \exp(z_k)$

Special case for output units (multiclass classification)

Multi-layer feedforward networks

We can generalize this to multi-layer feedforward nets



Output layer: vector **y**

Hidden layer: vector $\mathbf{h_n}$

Hidden layer: vector $\mathbf{h_1}$

Input layer: vector **x**

Challenges in using NNs for NLP

In NLP, the input and output variables are discrete: words, labels, structures.

NNs work best with continuous vectors.

- We typically want to learn a mapping (embedding) from discrete words (input) to dense vectors.
- We can do this with (simple) neural nets and related methods.

The input to a NN is (traditionally) a fixed-length vector. How do you represent a variable-length sequence as a vector?

- With recurrent neural nets: read in one word at the time to predict a vector, use that vector and the next word to predict a new vector, etc.;
- With convolutional nets: use a sliding (fixed-length) window)

How does NLP use NNs?

Word embeddings (word2vec, Glove, etc.)

Train a NN to predict a word from its context (or the context from a word) to get a dense vector representation of each word

Neural language models:

Use recurrent neural networks (RNNs, GRUs, LSTMs) to predict word sequences (or to obtain context-sensitive embeddings (ELMO)

Sequence-to-sequence (seq2seq) models:

From machine translation: use one RNN to encode source string, and another RNN to decode this into a target string.

Also used for automatic image captioning, etc.

Convolutional neural nets

Used e.g. for text classification

Transformers

Neural Language Models

What is a language model?

Probability distribution over the strings in a language, typically factored into distributions $P(w_i \mid \ldots)$ for each word:

$$P(\mathbf{w}) = P(w_1 \ldots w_n) = \prod_i P(w_i \mid w_1 \ldots w_{i-1})$$

N-gram models assume each word depends only preceding n–1 words:

$$P(w_i \mid w_1 \ldots w_{i-1}) =_{\text{def}} P(w_i \mid w_{i-n+1} \ldots w_{i-1})$$

To handle variable length strings, we assume each string starts with n–1 start-of-sentence symbols (BOS), or 〈S〉
and ends in a special end-of-sentence symbol (EOS) or 〈\S〉

An n-gram model $P(w \mid w_1 \ldots w_k)$

— The **vocabulary** V contains *n* types (incl. UNK, BOS, EOS)
— We want to condition each word on *k* preceding words

— [Naive] Each **input word** $w_i \in V$ (that we're conditioning on) is an ***n*-dimensional one-hot vector** $v(w) = (0,\ldots 0, 1, 0 \ldots 0)$
— Our **input layer** $\mathbf{x} = [v(w_1),\ldots,v(w_k)]$ has $n \times k$ elements
— To predict the probability over output words, the **output layer** is a softmax over n elements

$$P(w \mid w_1 \ldots w_k) = \text{softmax}(\mathbf{hW}^2 + \mathbf{b}^2)$$

With (say) one hidden layer $\mathbf{h}$ we'll need two sets of parameters, one for **h** and one for the output

Naive neural n-gram model

Architecture:

Input Layer: $\mathbf{x} = [v(w_1)\ldots.v(w_k)]$

$v(w)$: a one-hot vector of size $\dim(V) = |V|$

Hidden Layer: $\mathbf{h} = g(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1)$

Output Layer: $P(w \mid w1\ldots wk) = \text{softmax}(\mathbf{h}\mathbf{W}^2 + \mathbf{b}^2)$

Parameters:

Weight matrices and biases:

first layer: $\mathbf{W}^1 \in R^{k \cdot \dim(V)\times\dim(\mathbf{h})}$ $\mathbf{b}^1 \in R^{\dim(\mathbf{h})}$

second layer: $\mathbf{W}^2 \in R^{\dim(\mathbf{h})\times|V|}$ $\mathbf{b}^2 \in R^{|V|}$

How many parameters do we need to learn?

Traditional n-gram model: $\dim(V)^k$ parameters

With $\dim(V) = 10{,}000$ and $k=3$: 1,000,000,000,000

Naive neural n-gram model (one-hot encoding of vocabulary):

#parameters going to hidden layer: $k \cdot \dim(V) \cdot \dim(\mathbf{h})$,

with $\dim(\mathbf{h}) = 300$, $\dim(V) = 10{,}000$ and $k=3$: 9,000,000

plus #parameters going to output layer: $\dim(\mathbf{h}) \cdot \dim(V)$

with $\dim(\mathbf{h}) = 300$, $\dim(V) = 10{,}000$: 3,000,000

The neural model requires still a lot of parameters,
but far fewer than the traditional n-gram model

Naive neural n-gram models

Advantages over traditional n-gram models:

— The hidden layer captures interactions among context words

— Increasing the order of the n-gram requires only a small linear increase in the number of parameters.

dim($\mathbf{W}^1$) goes from k · dim(emb)×dim($\mathbf{h}$) to (k+1) · dim(emb)×dim($\mathbf{h}$)

A traditional k-gram model requires dim(V)^k parameters

— Increasing the vocabulary also leads only to a linear increase in the number of parameters

Better neural language models

Naive neural models have similar shortcomings as standard n-gram models

- — Models get very large (and sparse) as n increases
- — We can't generalize across similar contexts
- — N-gram Markov (independence) assumptions are too strict

Better neural language models overcome these by…

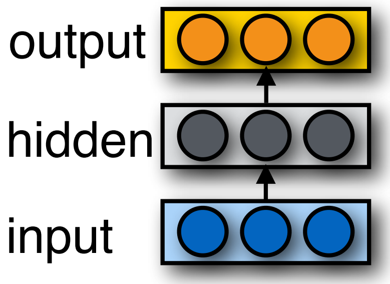
… using **word embeddings** instead of one-hots as input:
Instead of representing context words as distinct, discrete symbols (i.e. very high-dimensional one-hot vectors), use a dense low-dimensional vector representation where similar words have similar vectors [next]

… using **recurrent nets** instead of feedforward nets:
Instead of a fixed-length (n-gram) context, use recurrent nets to encode variable-lengths contexts [later class]

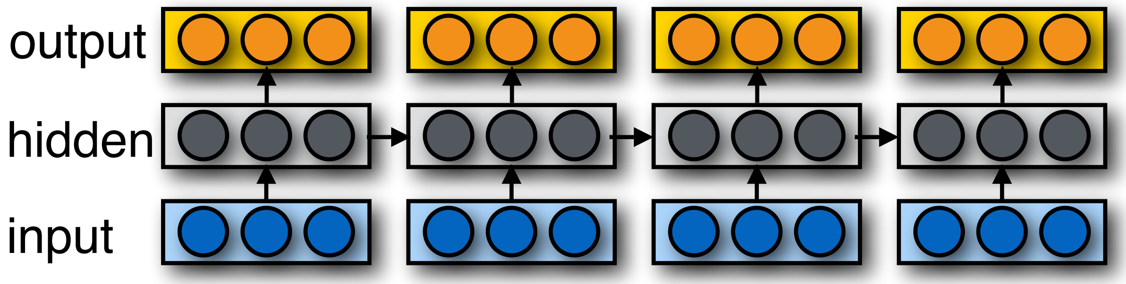
Recurrent neural networks (RNNs)

Basic RNN: Modify the standard feedforward architecture (which predicts a string $w_0 \ldots w_n$ one word at a time) such that the output of the current step (w_i) is given as additional input to the next time step (when predicting the output for w_{i+1}).

"Output" — typically (the last) hidden layer.



Feedforward Net



Recurrent Net

Word Embeddings (e.g. word2vec)

Main idea:
If you use a feedforward network to predict the probability of words that appear in the context of (near) an input word, the hidden layer of that network provides a dense vector representation of the input word.

Words that appear in similar contexts (that have high distributional similarity) wils have very similar vector representations.

These models can be trained on large amounts of raw text (and pretrained embeddings can be downloaded)

Sequence-to-sequence (seq2seq) models

Task (e.g. machine translation):

Given one variable length sequence as input,
return another variable length sequence as output

Main idea:

Use one RNN to encode the input sequence (“encoder”)

Feed the last hidden state as input to a second RNN (“decoder”) that then generates the output sequence.

Use attention mechanisms (e.g. to focus on certain parts of the input when generating output)

Transformers

Non-recurrent architecture for seq2seq tasks:

— Also has an encoder and a decoder, but these process the input at once
(decoder may mask future outputs to generate output sequentially)

— May use positional embeddings to capture sequence information

— May use multiple self-attention (attention within a sequence) mechanisms in parallel

— Can be (pre)trained on very large amounts of data, and then fine-tuned for specific tasks

— Yields state-of-the-art context-dependent encodings (BERT) and language models (GPT-2)

Syllabus for this class

Schedule

01/22	*Introduction*	Overview,Policies
01/24	*Static Word Embeddings*	Basic intro to Word2Vec, GloVe etc.
01/29	*Static Word Embeddings*	More in-depth analysis of static embeddings
01/31	*Basic Neural Architectures*	Recurrent Neural Nets and variants thereof
02/05	*Basic Neural Architectures*	Convolutional Neural Nets
02/07	*Lexical Encodings*	Character RNNs, BPE, FastText etc.
02/12	*Context-Dependent Word Embeddings*	Elmo etc.
02/14	*Advanced Neural Architectures*	Transformers
02/19	*Context-Dependent Word Embeddings*	BERT, GPT-2 et al.
02/21	*Context-Dependent Word Embeddings*	BERT, GPT-2 et al. c'td
02/26	*Advanced Neural Architectures*	Deep Learning for Graphs
02/28	*Project Proposal Presentations*	
03/04	*NLP Applications*	Neural Generation
03/06	*NLP Applications*	Neural Summarization
03/11	*NLP Applications*	Neural Machine Translation
03/13	*NLP Applications*	Neural Sequence Labeling
03/25	*NLP Applications*	Neural Structure Prediction
03/27	*NLP Applications*	Neural Question Answering
04/01	*NLP Applications*	More on Neural Question Answering
04/03	*NLP Applications*	Multimodal NLP
04/08	*NLP Applications*	Neural Dialogue
04/10	*NLP Applications*	More on Neural Dialogue
04/15	*Project Update Presentations*	
04/17	*Advanced Neural Architectures*	Intro to GANs in NLP
04/22	*Advanced Neural Architectures*	More on GANs in NLP
04/24	*Training Regimes*	Reinforcement Learning in NLP
04/29	*(TBC) Training Regimes*	(TBC) More on Reinforcement Learning in NLP
05/01	*(TBC) Training Regimes*	(TBC) Deep Learning in Low Resource Settings
05/07	*Final Project Presentations*	

Admin

You will receive an email with a link to a Google form where you can sign up for slots to present.

— Please sign up for at least three slots so that I have some flexibility in assigning you to a presentation

We will give you one week to fill this in.

You will have to meet with me the Monday before your presentation to go over your slides.

Grading criteria for presentations

— Clarity of exposition and presentation
— Analysis (don’t just regurgitate what’s in the paper)
— Quality of slides (and effort that went into making them — just re-using other people’s slides is not enough)