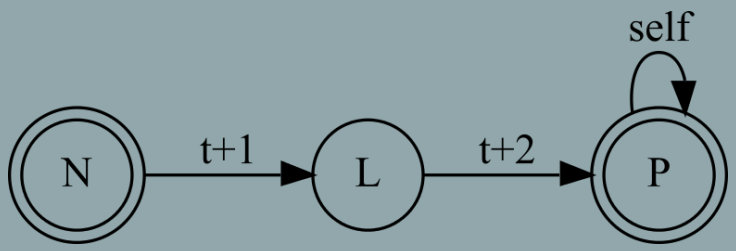
LING-362

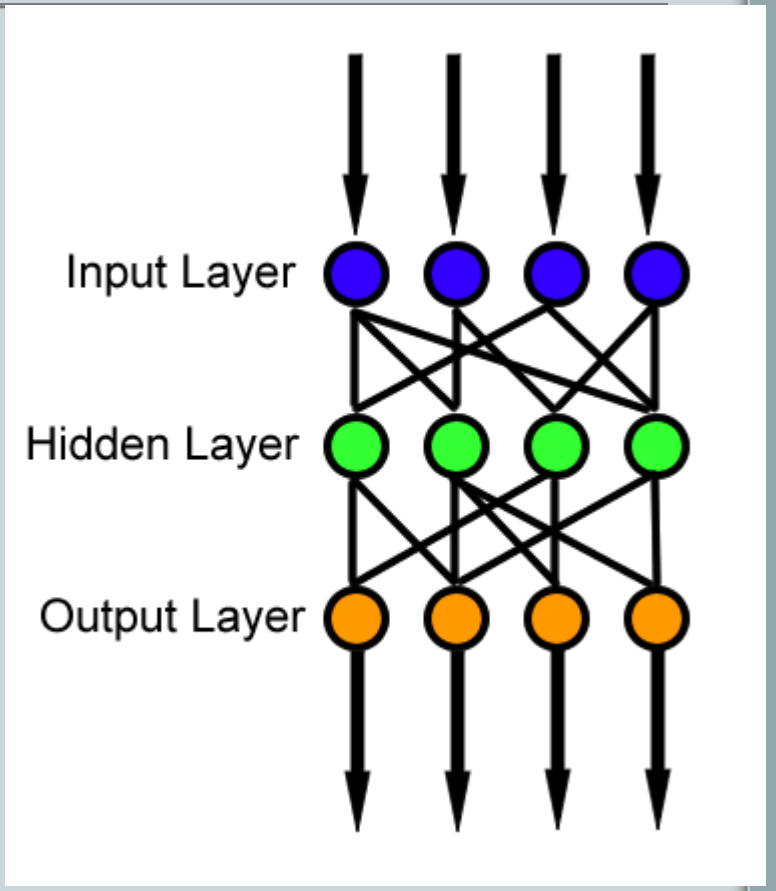
Introduction to Natural Language Processing

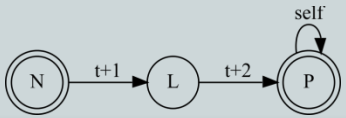
Neural Language Models part 2 / Tagging and Hidden Markov Models 1



Neural LMs

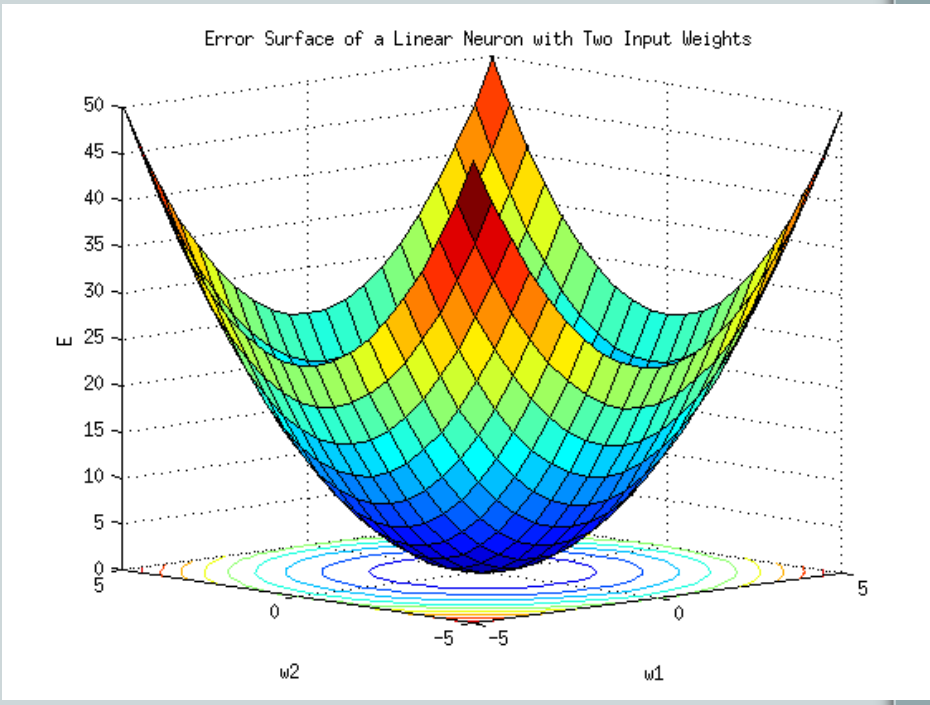
- Feed forward, back propagation
- Map any input to any output, search for optimal weights

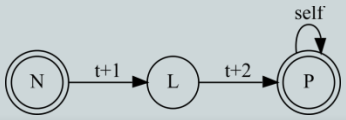




Gradient descent

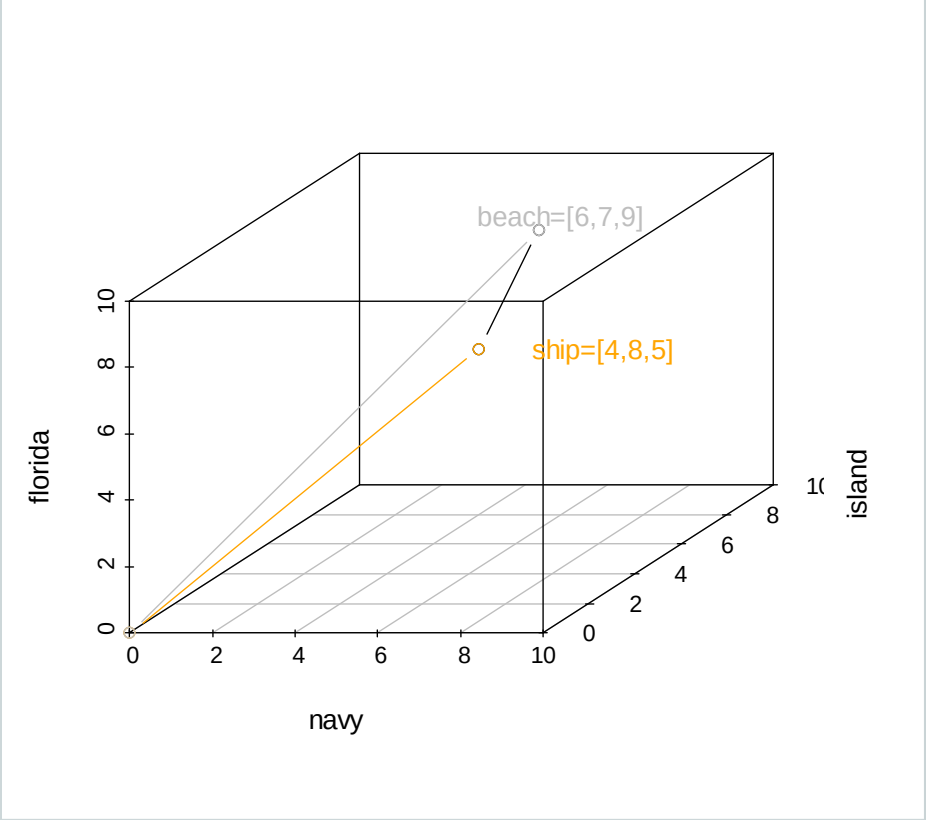
- Use **derivative** of **loss function** (gradient descent)
- Change weights iteratively, usually in **mini-batches**
- Slow down as we go along (**learning rate** decay)

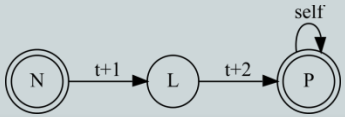




Represent words as vectors

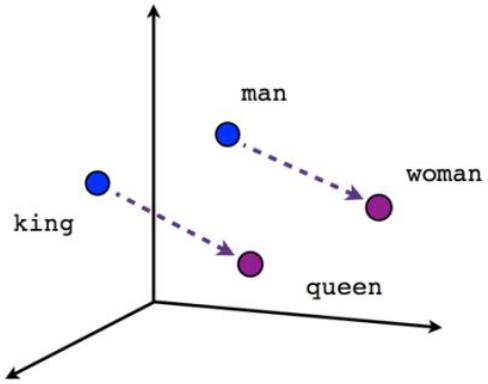
- Co-occurrence frequencies (or transformations thereof) make a vector space
- Allows similarity metrics for words and documents ***(later!)***
- Models of meaning based on neighboring words



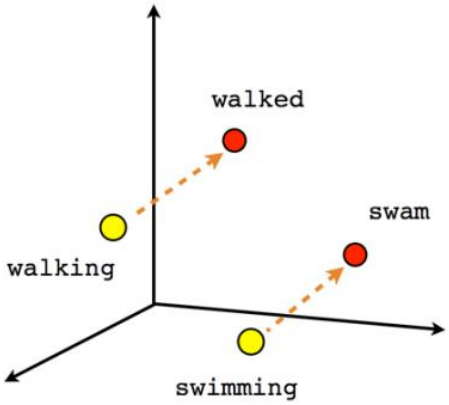


Applications

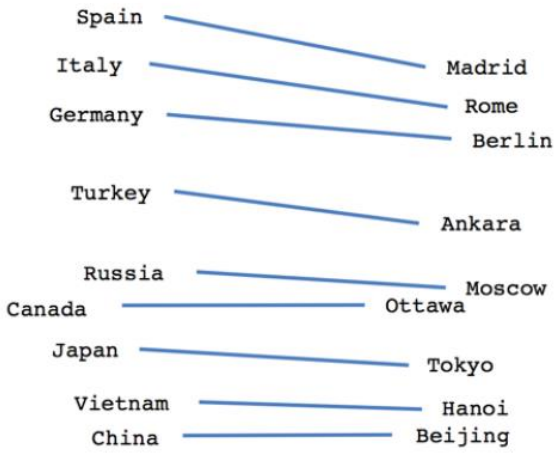
- Projecting vectors to lower dimensions
  - Reveal systematic relationships
  - Word level similarity



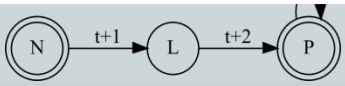
Male-Female



Verb tense



Country-Capital



Word2Vec

Don't count, predict! (see Baroni et al. 2014)

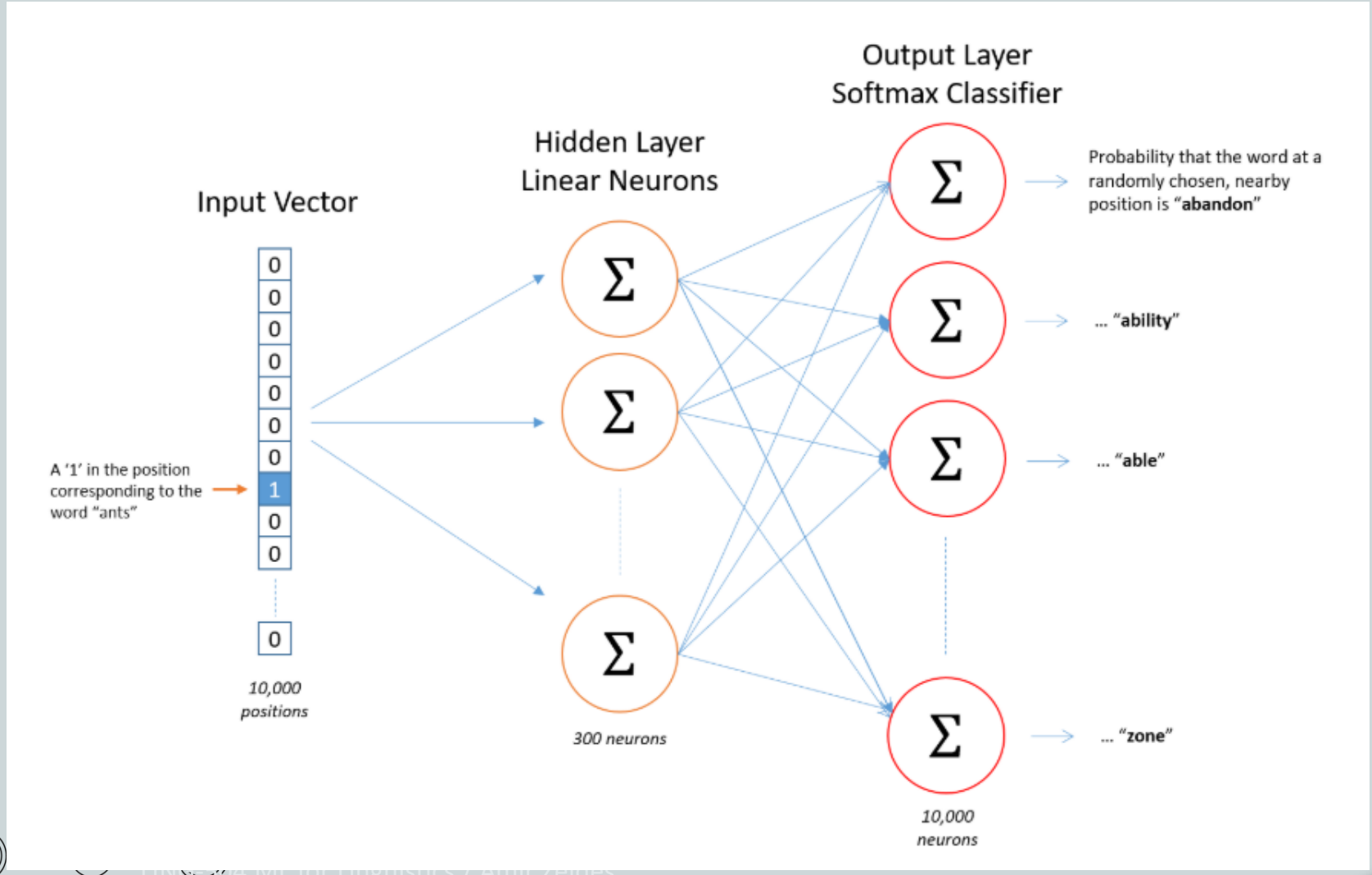
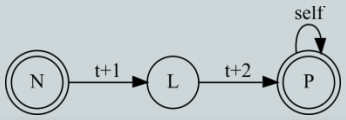


Image: mccormickml.com

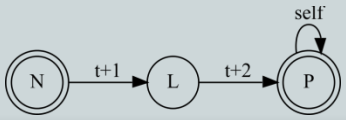
Training a model – toy example

```
from gensim import models

sotu = "sotu_sent_per_line.txt"

with open(sotu,'r',encoding="utf8") as f:
  plain_text = f.read()


sentences = plain_text.split("\n")
```



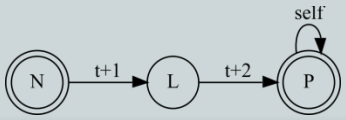
Training a model – toy example

```
tokenized = []

for sentence in sentences:
   tokens = sentence.strip().lower().split(" ")
   tokenized.append(tokens)

model = models.Word2Vec(tokenized, min_count=2, size=50)

print(model['america'])
-- [ 0.17634061  0.58502656  0.27098337 -0.17523931 -
0.24094008 -1.72932017 ...]
```



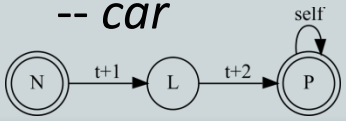
Training a model – toy example

```
# Is 'america' more similar to 'country' or 'goal'?
print(model.wv.similarity('america', 'country') >
model.similarity('america','goal'))

-- True

# Let's find the most similar words to 'america'
print(model.wv.most_similar(positive=['america'],topn=3))

-- [('world', 0.7713112235069275), ('nation', 0.737435519695282), ('freedom', 0.72567957639691)]
-- [('world', 0.7955950498580933), ('freedom', 0.7540770173072815), ('nation', 0.73620635271072)]
-- [('nation', 0.85256427526474), ('best', 0.8303712606430054), ('future', 0.8137349486351013)
...
```



Training a model – toy example

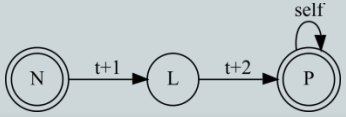
```
# What's a leader/king like?
print(model.wv.most_similar(positive=['leader','king'],topn=2))
-- [('elected', 0.9522648453712463)]
-- [('emperor', 0.851976871490785)]


# What if we're looking for words more distant from king?
print(model.wv.most_similar(positive=['american','leader'],
negative=['king'],topn=2))
-- [('freedom', 0.803984224796295 2)]
-- [('human', 0.6412678956985474)]

# Spot the odd one out
print(model.wv.doesnt_match(["france","germany","car","japan"]))
-- car
```



Much bigger in real life…

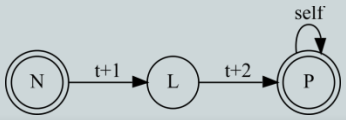
- These examples come from less than 2M tokens / 10 MB of text – often rather bad!
- You can use ready-made examples from Google, Wikipedia, …
  - Google Word2Vec text is 3.6 GB, popular trimmed News version ~80MB
  - GloVe 300D pre-trained embeddings ~1GB (Paddington et al.)
  - BERT Base even larger, BERT Large has 345M parameters based on 3.5G words, takes about 4 days on 16 cloud TPUs (~about 100 mile car drive of electricity!)
- Not trained on the fly – used as saved trained models
- Typically held in memory, not loaded for each function call
- Still slow to load on a laptop, require high GPU RAM

Gensim lets you train your own medium sized models!



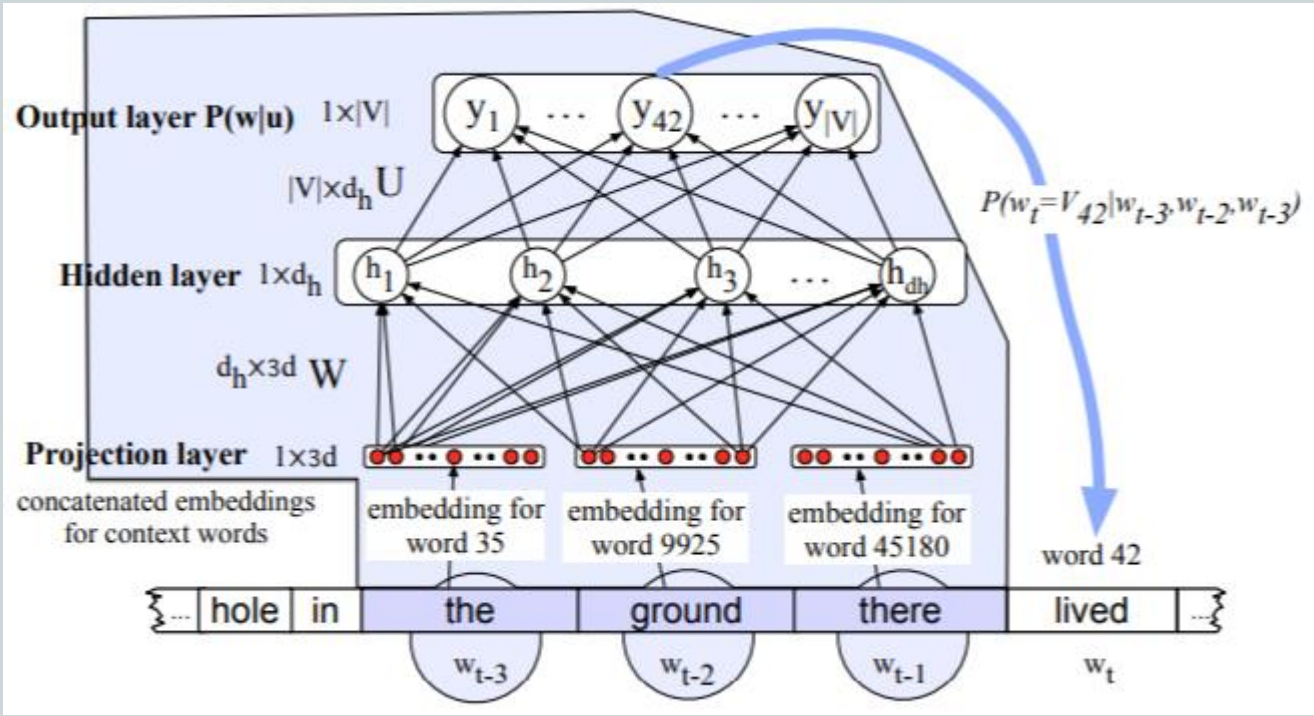
Beyond word similarity

- Word embeddings are now one of the most popular ways of representing text:
  - Feed tools embeddings instead of words
  - Probabilities based on vector dimensions – allows reasonable behavior for OOV items
  - Open to mathematical operations:
    - Sentence meaning = avg. of word vectors?
    - Classify sentence sentiment using vectors?
    - Discourse segmentation via differences in sentence meaning?
    - …

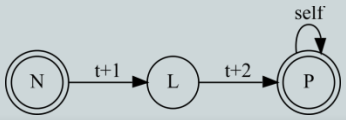


Using them in language models

- In a feed forward network we could do:

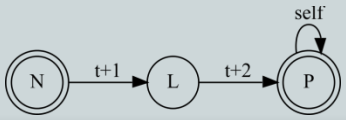


Jurafsky & Martin (2017)

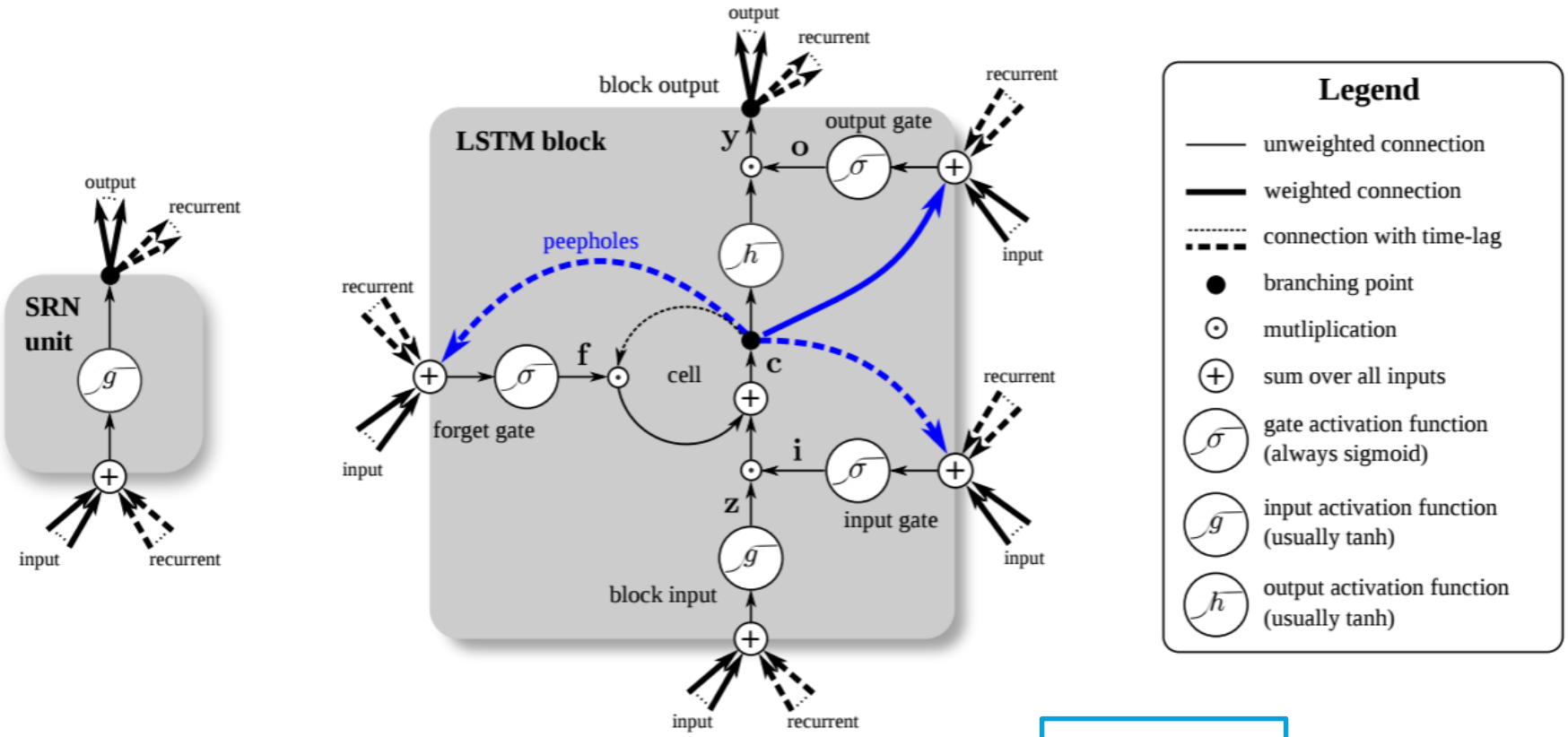


Using memory

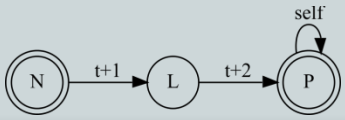
- Recent neural models use **memory** based architectures (Recurrent Neural Networks - RNNs) or attention weights (Transformers)
- Popular type: Long Short Term Memory (LSTM) networks
  - RNN cells don't just get input 'synapse' weights, but also activate themselves
  - Allows cells to remember previous states
  - In LSTMs: cells also learn when to forget what they've seen



LSTM cell structure



Greff et al. (2015)



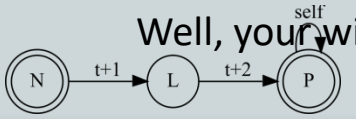
What can LSTMs do?

- Intuitive example: character level sequence to sequence modeling
- Example – trained on Shakespeare:

PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

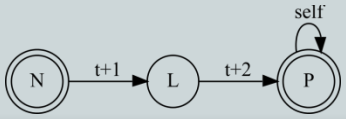


http://karpathy.github.io/2015/05/21/rnn-effectiveness/

What can LSTMs do?

- Intuitive example: character level sequence to sequence modeling
- Example – trained on Wikipedia:

Naturalism and decision for the majority of Arab countries' capitalide was grounded by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal in the [[Protestant Immineners]], which could be said to be directly in Cantonese Communication, which followed a ceremony and set inspired prison, training. The emperor travelled back to [[Antioch, Perth, October 25|21]] to note, the Kingdom of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known in western [[Scotland]], near Italy to the conquest of India with the conflict. … Many governments recognize the military housing of the [[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]], that is sympathetic to be to the [[Punjab Resolution]] (PJS)[http://www.humah.yahoo.com/guardian.cfm/7754800786d17551963s89.htm

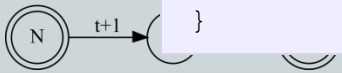


http://karpathy.github.io/2015/05/21/rnn-effectiveness/

What can LSTMs do?

- Example – trained on Linux source code:

```
/*
 * If this error is set, we will need anything right after
that BSD.
 */
static void action_new_function(struct s_stat_info *wb)
{
  unsigned long flags;
  int lel_idx_bit = e->edd, *sys & ~((unsigned long)
*FIRST_COMPAT);
  buf[0] = 0xFFFFFFFF & (bit << 4);
  min(inc, slist->bytes);
  printk(KERN_WARNING "Memory allocated %02x/%02x, "
    "original MLL instead\n"),
    min(min(multi_run - s->len, max) * num_data_in),
    frame_pos, sz + first_seg);
  return disassemble(info->pending_bh);
}

static void num_serial_settings(struct tty_struct *tty)
{
  if (tty == tty)
    disable_single_st_p(dev);
  pci_disable_spool(port);
  return 0;
}
```



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

What are these cells learning?

Cell sensitive to position in line:

```
The sole importance of the crossing of the Berezina lies in the fact
that it plainly and indubitably proved the fallacy of all the plans for
cutting off the enemy's retreat and the soundness of the only possible
line of action--the one Kutuzov and the general mass of the army
demanded--namely, simply to follow the enemy up. The French crowd fled
at a continually increasing speed and all its energy was directed to
reaching its goal. It fled like a wounded animal and it was impossible
to block its path. This was shown not so much by the arrangements it
made for crossing as by what took place at the bridges. When the bridges
broke down, unarmed soldiers, people from Moscow and women with children
who were with the French transport, all--carried on by vis inertiae--
pressed forward into boats and into the ice-covered water and did not,
surrender.
```

Cell that turns on inside quotes:

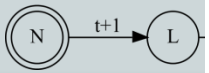
```
"You mean to imply that I have nothing to eat out of.... On the
contrary, I can supply you with everything even if you want to give
dinner parties," warmly replied Chichagov, who tried by every word he
spoke to prove his own rectitude and therefore imagined Kutuzov to be
animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating
smile: "I meant merely to say what I said."
```

Cell that robustly activates inside if statements:

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
        siginfo_t *info)
{
  int sig = next_signal(pending, mask);
  if (sig) {
    if (current->notifier) {
      if (sigismember(current->notifier_mask, sig)) {
        if (!(current->notifier)(current->notifier_data)) {
          clear_thread_flag(TIF_SIGPENDING);
          return 0;
        }
      }
    }
    collect_signal(sig, pending, info);
  }
  return sig;
}
```

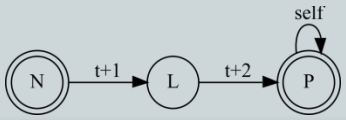
A large portion of cells are not easily interpretable. Here is a typical example:

```
/*  Unpack a filter field's string representation from user-space
 *  buffer.  */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
  char *str;
  if (!*bufp || (len == 0) || (len > *remain))
    return ERR_PTR(-EINVAL);
  /*  Of the currently implemented string fields, PATH_MAX
   *  defines the longest valid length.
   */
```



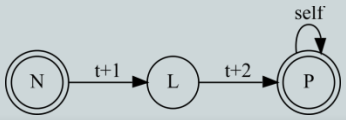
Training your own

- A relatively simple model works out of the box using PyTorch:
  - pip install torch
  - https://github.com/pytorch/examples/tree/master/word_language_model
- Other good libraries: Tensorflow, Keras

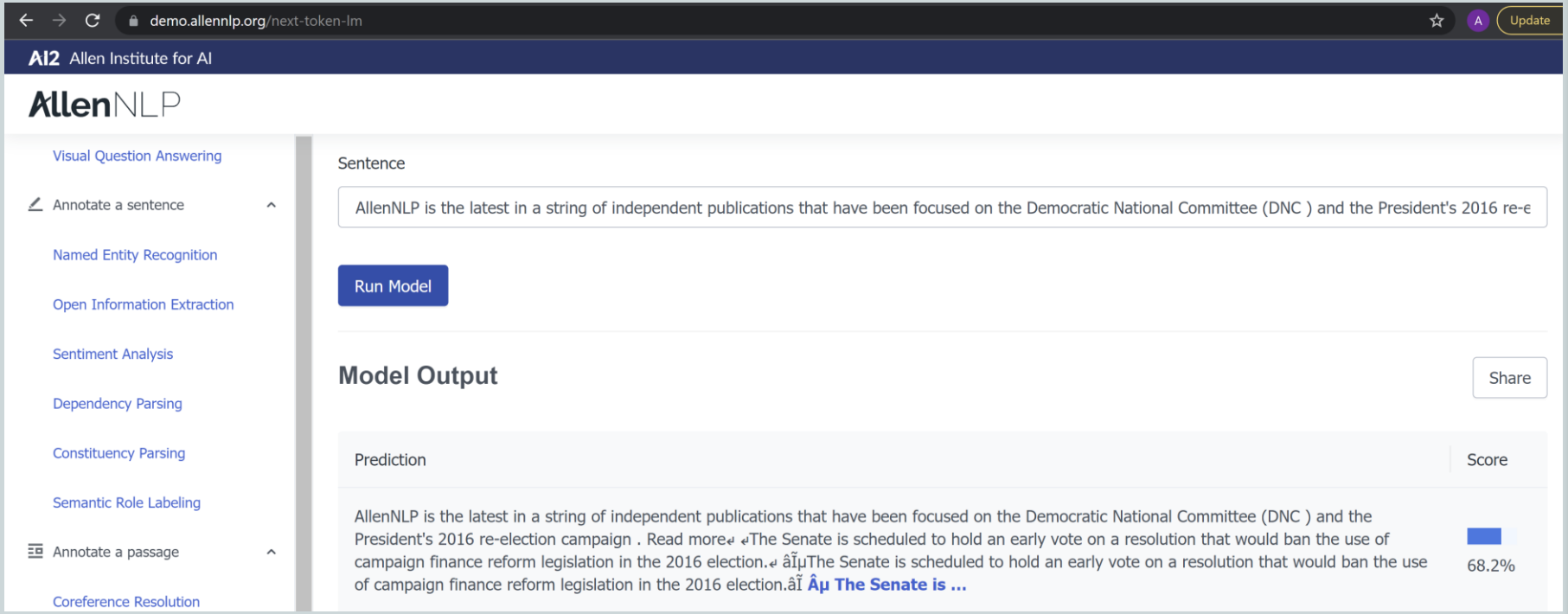


Bonus fun

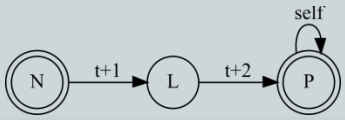
- You can test AllenNLP's neural LM here:
  - https://demo.allennlp.org/next-token-lm
- And you can chat with a neural network trained on conversational pairs
- Example:
  - http://neuralconvo.huggingface.co/
  - (also compare Microsoft's TAY: https://twitter.com/tayandyou)



Do neural LMs solve all problems?

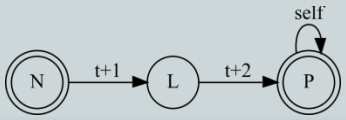


Generated on 2021-10-18



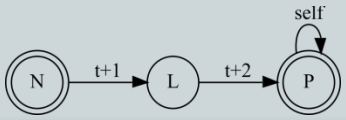
More information

- We will learn more about practical applications of neural networks later
- Learning how neural models work in depth is outside the scope of this course
  - Jurafsky & Martin 2017, C7 is a good starting point
  - Grad students: once you are confident in coding, consider taking LING-504/COSC-576
- Further reading:
  - Jurafsky & Martin (2017, C7)
  - *Hands-on Machine Learning with Scikit-Learn and TensorFlow* / A. Geron, 2019 (https://github.com/ageron/handson-ml)



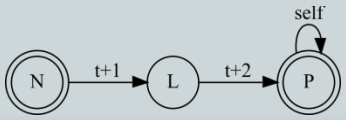
A more abstract view of ngrams

- What do language modes really 'model'?
  - Probabilities of individual words
  - Probabilities of sequences of words
- How is our language model using them?
  - Get **transitional** ~~possibilities~~ probabilities
  - What are the odds of moving **from word X to word Y**?
- *We've seen something like this before…*



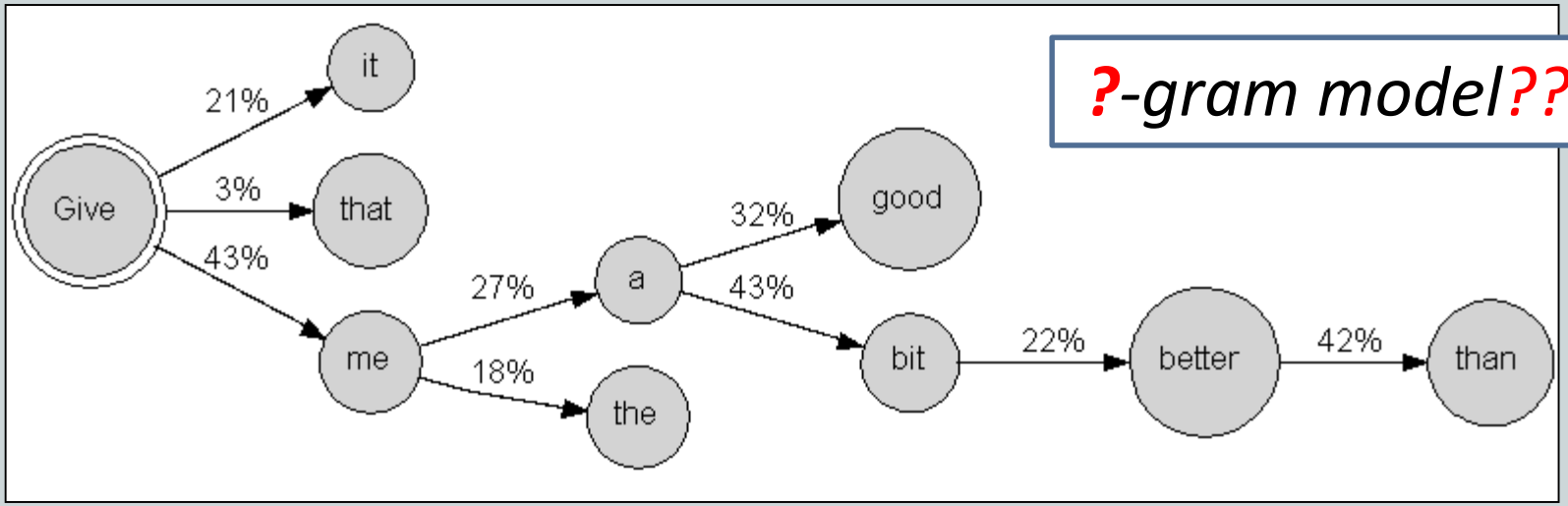
Transitional probabilities

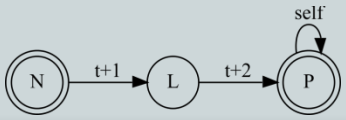
- Moving through a language model is like progressing in a web of words:
  - Let's say I type the word "Give" into my smartphone's message app
  - This is a job for the auto SMS wizard! ☺
  - Here's what happens when I click next, next…
    - *Give me a bit better than the bus and should be there in a couple of days ago…*



Language models as FSAs

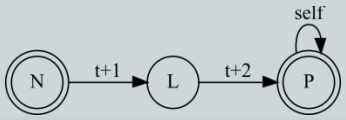
- Language model choices are probabilistic – different from deterministic FSAs
- But we can represent them as **weighted** automata:





Markov Chains

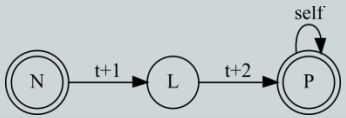
- A set of ordered variables with probabilities following the **Markov Property:**
  - The probability of each value of $\boldsymbol{X_i}$ in the sequence depends **only** on $\boldsymbol{X_{i-1}}$
  - (Or in variants: some other sufficiently small number: ***second order Markov Model***, ***third order***… etc.)
  - In other words: context effects are limited, but can chain
  - Formally: $P(\boldsymbol{X_i}=x \mid \boldsymbol{X_{i-1}}=x_1, \boldsymbol{X_{i-2}}=x_2,\ldots) = P(\boldsymbol{X_i}=x \mid \boldsymbol{X_{i-1}}=x_1)$
- This is a shameless, but very useful simplification! ☺



An example

- Suppose the difficulty of a homework assignment is influenced by the previous one
  - If the last assignment was easy, this next one will be hard with 70% probability (but 30%: easy)
  - If the last one was hard, 60% that the next will be easy (but 40%: still hard)
- Results are uncertain, but depend **only on last time**
- Globally we still model a process where difficulty alternates **across the chain**

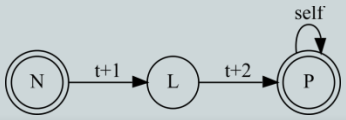
➢ **This works not just for words!!**



The Markov property assumption

- Note n% transition depends only on the current state
- We get a 'plausible' sequence overall
- What we need for this:
  - Probabilities of each $P(w_k|w_{k-1})$
  - Smoothing for missing values

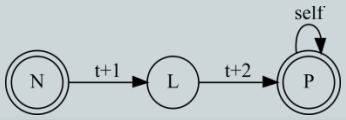
- We know how to get these for words, but what about other categories?



Beneath the surface

- Token n-gram models represent transitions between **actually observed characters/words**
- We can call them **Visible Markov Models (VMMs)**

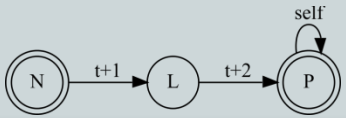
- Besides properties that are overt, we are interested in the probabilities of **hidden** categories
- These will require **Hidden Markov Models (HMMs)**



Invisible categories

- Which categories are hidden?
  - We may not be interested in a specific adjective like ***number (than)***
  - We might want to know the likelihood of **any** comparative adjective at this position
  - Or the probabilities that words refer to a company, or have positive sentiment, or …
  - How can we look at categories that are not in the data explicitly?

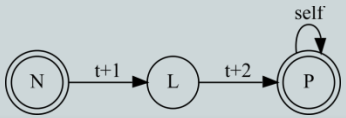
- Let's look at an example



POS tagging

- Probably the most widely used 'hidden' category in NLP
  - Assume each token has exactly 1 correct part of speech
  - We can't see it, but it's there
  - If we knew the POS tags of a text, we could create n-gram models describing them:
    - ART ADJ N → NP trigram!
    - TO ADV V → split infinitive!

- What tags are there?



Tag sets for English

Common in the US:

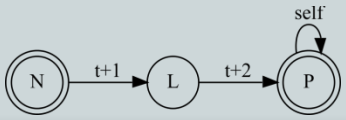
- Penn Treebank Tagset (PTB) — 36 Tags
- Extended PTB (AMALGAM/TT) — 56 Tags

Common in the UK:

- CLAWS 5 — 62 Tags
- CLAWS 7 — 137 Tags

Other notable mentions:

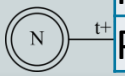
- Brown tag set — 85 Tags
- Google “Universal Tags” (V2) — 17 Tags



The PTB tag set (vanilla)

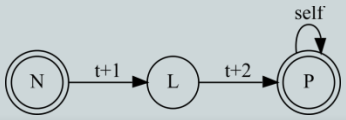
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun

PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person sg. present
VBZ	Verb, 3rd person sg. present
WDT	Wh-determiner
WP	Wh-pronoun
WP$	Possessive wh-pronoun
WRB	Wh-adverb



Tagging exercise

- Tag the text: ***Is ISIS Going Broke?***
  - What's easy and what's hard?
  - How do we determine the correct tag?
  - How can a computer do it?



The PTB tag set

- There is a lot to be said about the PTB tag set
  - Successes and shortcomings
  - Extensions since its inception – notably through the AMALGAM project (2001), TreeTagger, OntoNotes, English Web Treebank…
- We don't have time to discuss these…
- For this course: PTB (a.k.a. vanilla PTB) will be our only tag set for English (more: LING-367)