Neural Networks and Deep Learning

Deep Learning in Natural language Processing

Shumin Wu

Department of Computer Science

shumin.wu@colorado.edu

February 12, 2020

Representing Words in ML

- One-hot encoding
  - Build a dictionary of all words (n) from a corpus
  - Represent each word by a vector of size n

```
a      = [1, 0, 0, 0, …, 0]
animal = [0, 1, 0, 0, …, 0]
antler = [0, 0, 1, 0, …, 0]
...
zebra  = [0, 0, 0, 0, …, 1]
```

  - For bi-gram features, build a dictionary of all bi-grams from a corpus

Words/word pairs are treated as completely independent features (even synonyms).

Latent Semantic Analysis

- Create a term-document matrix
  - Each cell representing the appearance or frequency of term i in document j

	doc 1	doc 2	doc 3	doc 4	doc 5	…	doc n
animal	10	5		3	4		20
antler		2		1	2		4
tiger	1				4		5
…							
word m	3			4			10

Latent Semantic Analysis

- Create a term-document matrix
  - Each cell representing the appearance or frequency of term i in document j

	doc 1	doc 2	doc 3	doc 4	doc 5	…	doc n
animal	10	5		3	4		20
antler		2		1	2		4
tiger	1				4		5
…							
word m	3			4			10

LSA Singular Value Decomposition

Perform SVD:

$$X = U\Sigma V^T$$

$$X \in \mathbb{R}^{m \times n}$$

$$U \in \mathbb{R}^{m \times k}$$

$$\Sigma \in \mathbb{R}^{k \times k}$$

$$V^T \in \mathbb{R}^{k \times n}$$

Orthogonal matrix with left singular vectors (each row corresponds to a term). Synonyms tend to have similar vectors.

Diagonal matrix with (usually ≤ 100) largest singular values.

Orthogonal matrix with right singular vectors (each column corresponds to a document). Document is treated as a bag of words.

Language Model

- Modeling the probability of a sequence of text in a given language
- Applications
  - Speech recognition
  - Intelligent keyboard
  - Smart reply
  - Machine translation

Why do cats

hate water

read books

neural network

N-gram Language Model

- Probability of the next word given the first n-1 words

$$P(w_n|w_1,\ldots,w_{n-1})$$

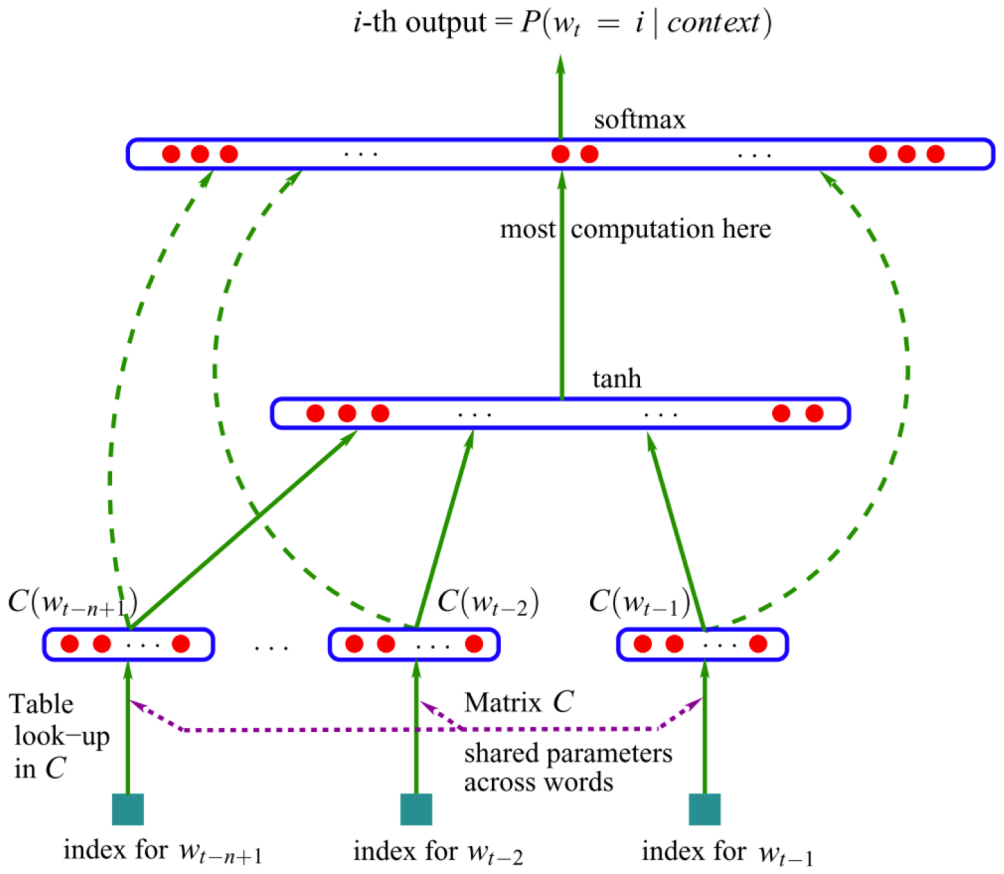
- Classically by counting n-grams in corpus with smoothing for rare or unseen n-grams (Kneser & Ney, 1995)
  - For English, good coverage to 3-grams, usually not longer than 5-grams

What about clues that are further away:

I have a dog and **he really likes to _____**

Neural Language Model: Bengio et al., 2003

- Dense vector representation of words (30-100 dimensions)
- Simultaneously learns word representation (embedding) and n-gram language model.
  - Does not benefit from SVD initialization of word embeddings



Source: A Neural Probabilistic Language Model

Neural Language Model: Bengio et al., 2003

Results on Brown corpus

h is the hidden layer size

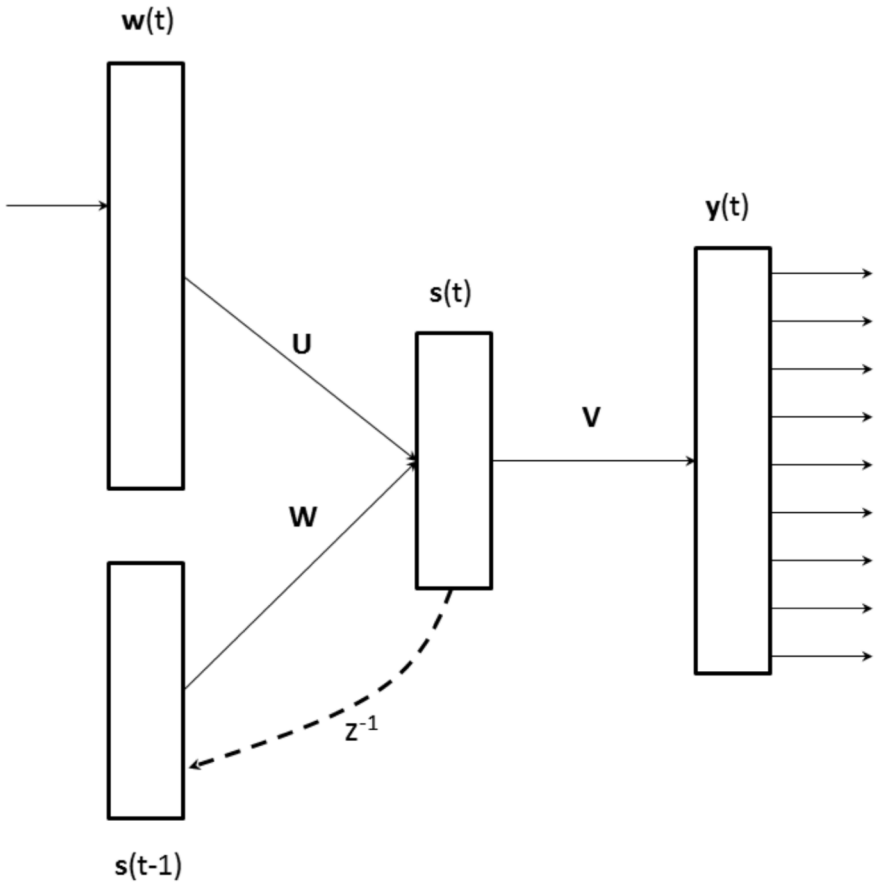
m is the word embedding size

Direct indicates whether there is direct connection from word embedding to output

Mix indicates whether network output is mixed w/ trigram model

	n	c	h	m	direct	mix	train.	valid.	test.
MLP1	5		50	60	yes	no	182	284	268
MLP2	5		50	60	yes	yes		275	257
MLP3	5		0	60	yes	no	201	327	310
MLP4	5		0	60	yes	yes		286	272
MLP5	5		50	30	yes	no	209	296	279
MLP6	5		50	30	yes	yes		273	259
MLP7	3		50	30	yes	no	210	309	293
MLP8	3		50	30	yes	yes		284	270
MLP9	5		100	30	no	no	175	280	276
MLP10	5		100	30	no	yes		265	**252**
Del. Int.	3						31	352	336
Kneser-Ney back-off	3							334	323
Kneser-Ney back-off	4							332	321
Kneser-Ney back-off	5							332	321
class-based back-off	3	150						348	334
class-based back-off	3	200						354	340
class-based back-off	3	500						326	**312**
class-based back-off	3	1000						335	319
class-based back-off	3	2000						343	326
class-based back-off	4	500						327	312
class-based back-off	5	500						327	312

Recurrent Neural Network based LM



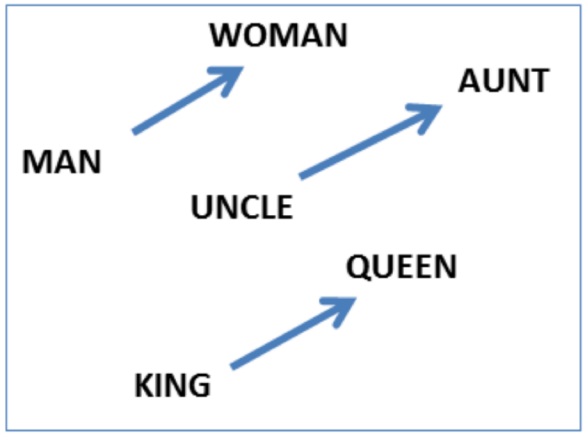
$$s(t) = \sigma(Uw(t) + Ws(t-1))$$
$$y(t) = \mathrm{softmax}(Vs(t))$$

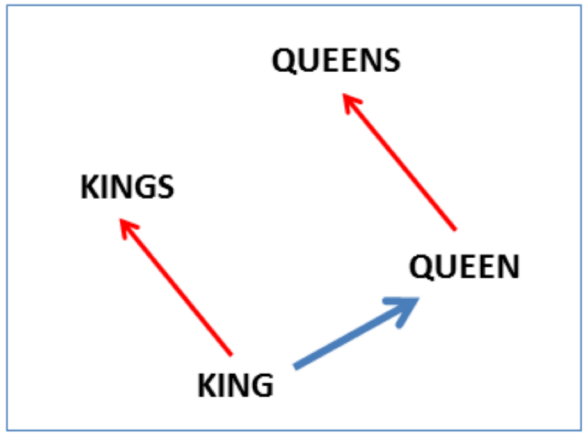
Mikolov et al. 2010

RNN LM Linguistic Regularities

Given normalized word representations learned by the RNN LM, for an analogy question a:b c:d, take the word embeddings x_i for words a, b, c, d and compute:

$$y = x_b - x_a + x_c$$





Then confirm whether x_d is the nearest neighbor to y

Mikolov et al. 2013

RNN LM Linguistic Regularities Results

Method	Adjectives	Nouns	Verbs	All
LSA-80	9.2	11.1	17.4	12.8
LSA-320	11.3	18.1	20.7	16.5
LSA-640	9.6	10.1	13.8	11.3
RNN-80	9.3	5.2	30.4	16.2
RNN-320	18.2	19.0	45.0	28.5
RNN-640	21.0	25.2	54.8	34.7
RNN-1600	**23.9**	**29.2**	**62.2**	**39.6**

Compared with LSA

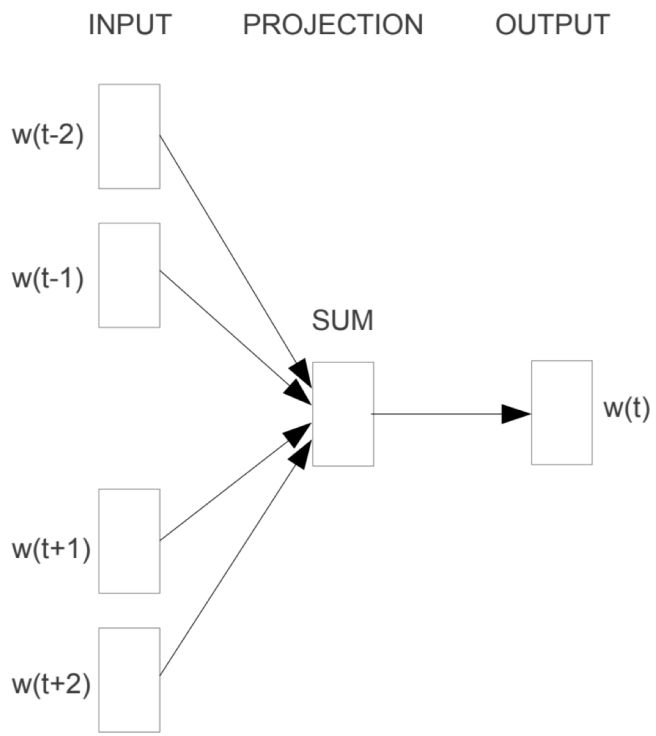
Method	Adjectives	Nouns	Verbs	All
RNN-80	**10.1**	8.1	**30.4**	**19.0**
CW-50	1.1	2.4	8.1	4.5
CW-100	1.3	4.1	8.6	5.0
HLBL-50	4.4	5.4	23.1	13.0
HLBL-100	7.6	**13.2**	30.2	18.7

Compared with earlier NN LM

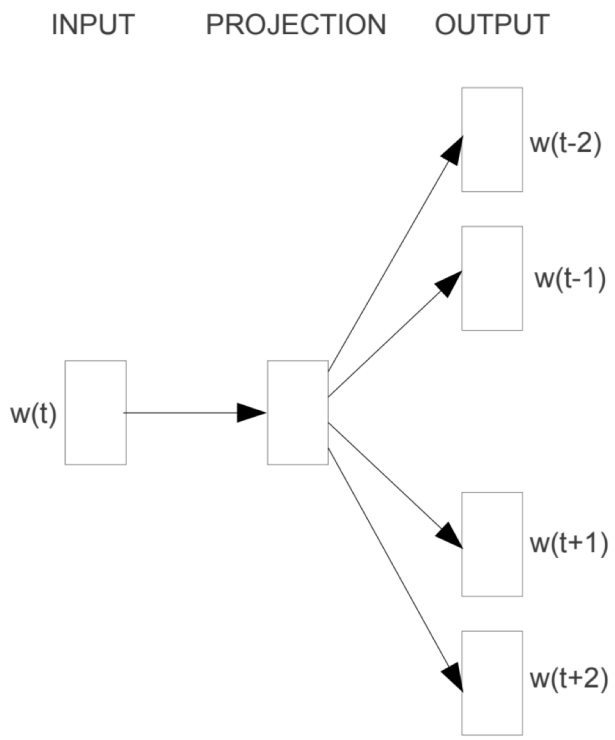
Word2vec

- Mikolov et al. 2013a, 2013b (negative sampling)
- Learning word representation without explicit language modeling
- Two tasks/models
  - Continuous bags of words (CBOW): given a window of surrounding words (before and after) without order, predict the missing middle word
  - Skip-gram: given the middle word, predict a window of surrounding words
- Widely used as input to many NLP tasks.

Word2vec



CBOW



Skip-gram

Subsample frequent words

Negative sampling: update parameters of few non-label words

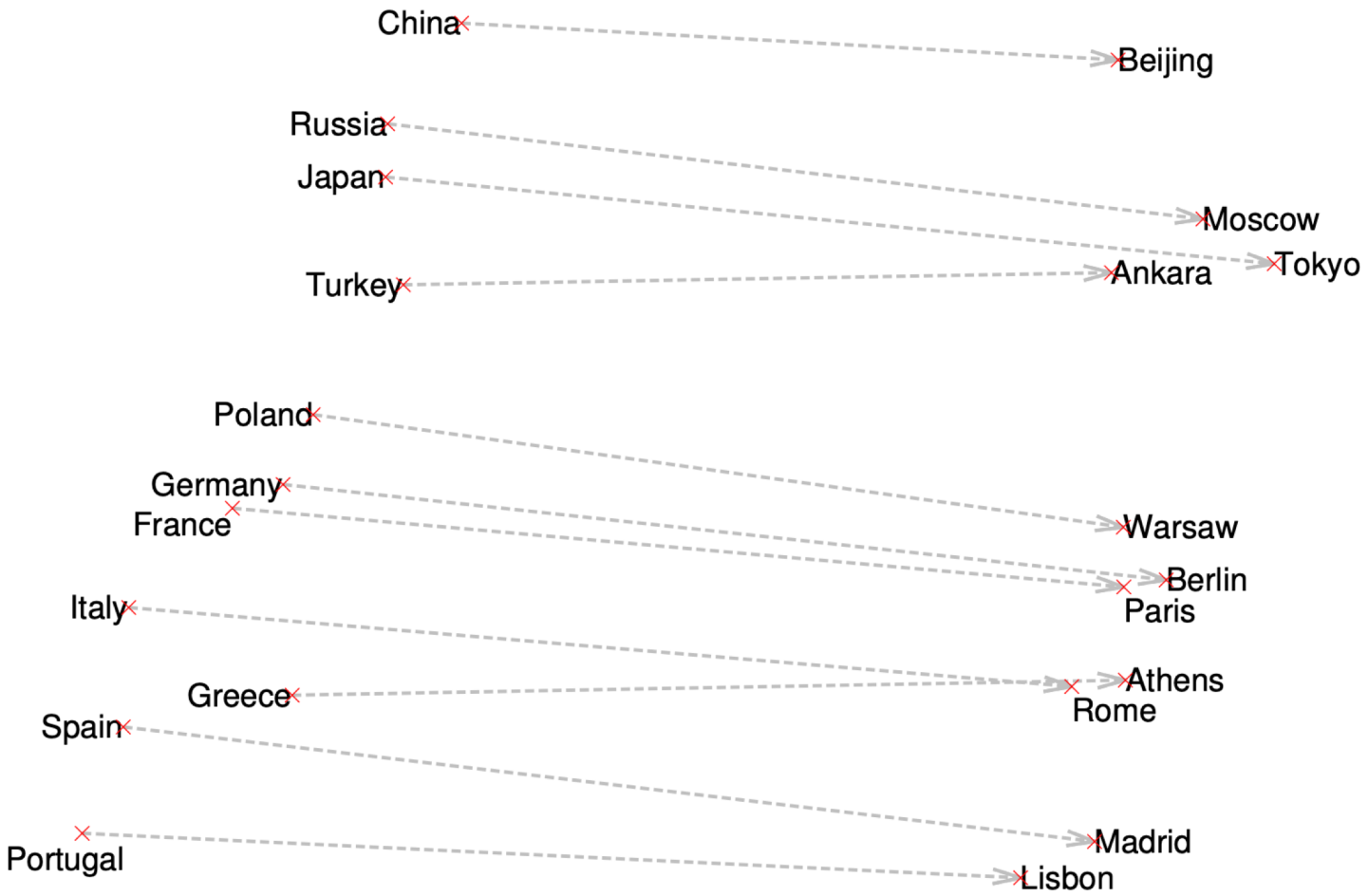
Word2vec Results

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Model	Vector Dimensionality	Training words	Accuracy [%]		
			Semantic	Syntactic	Total
Collobert-Weston NNLM	50	660M	9.3	12.3	11.0
Turian NNLM	50	37M	1.4	2.6	2.1
Turian NNLM	200	37M	1.4	2.2	1.8
Mnih NNLM	50	37M	1.8	9.1	5.8
Mnih NNLM	100	37M	3.3	13.2	8.8
Mikolov RNNLM	80	320M	4.9	18.4	12.7
Mikolov RNNLM	640	320M	8.6	36.5	24.6
Huang NNLM	50	990M	13.3	11.6	12.3
Our NNLM	20	6B	12.9	26.4	20.3
Our NNLM	50	6B	27.9	55.8	43.2
Our NNLM	100	6B	34.2	**64.5**	50.8
CBOW	300	783M	15.5	53.1	36.1
Skip-gram	300	783M	**50.0**	55.9	**53.3**

Word2vec

PCA projection down to 2D of capital cities:

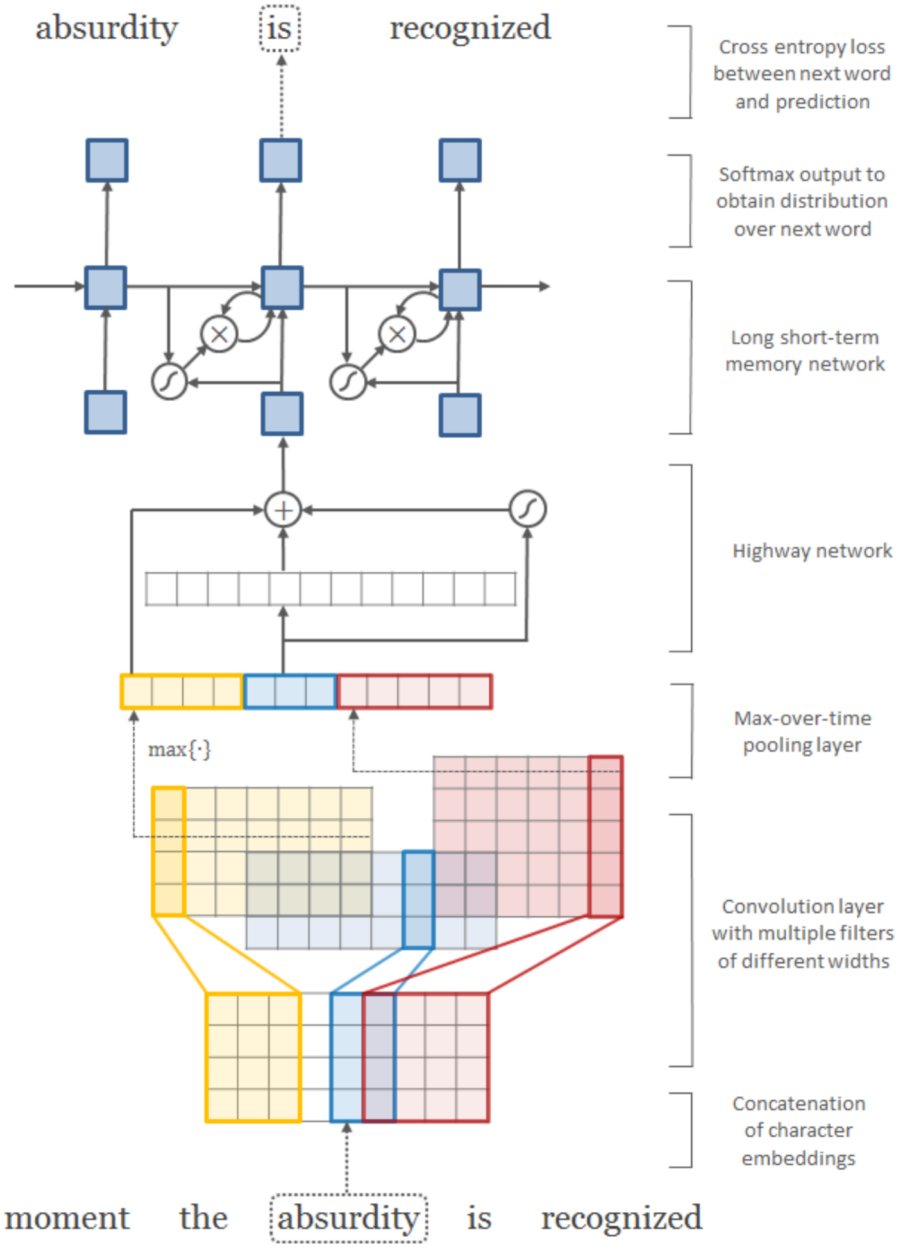


Character-based

Language Model: Kim et al., 2016

- Stacks RNN on top of CNN
  - CNN Highway network
- Well suited languages:
  - low resource
  - morphologically rich
  - large character set (eastern languages)
- Resilient to out-of-vocabulary words

Machine translation: Ling et al., 2016; Lee et al., 2017

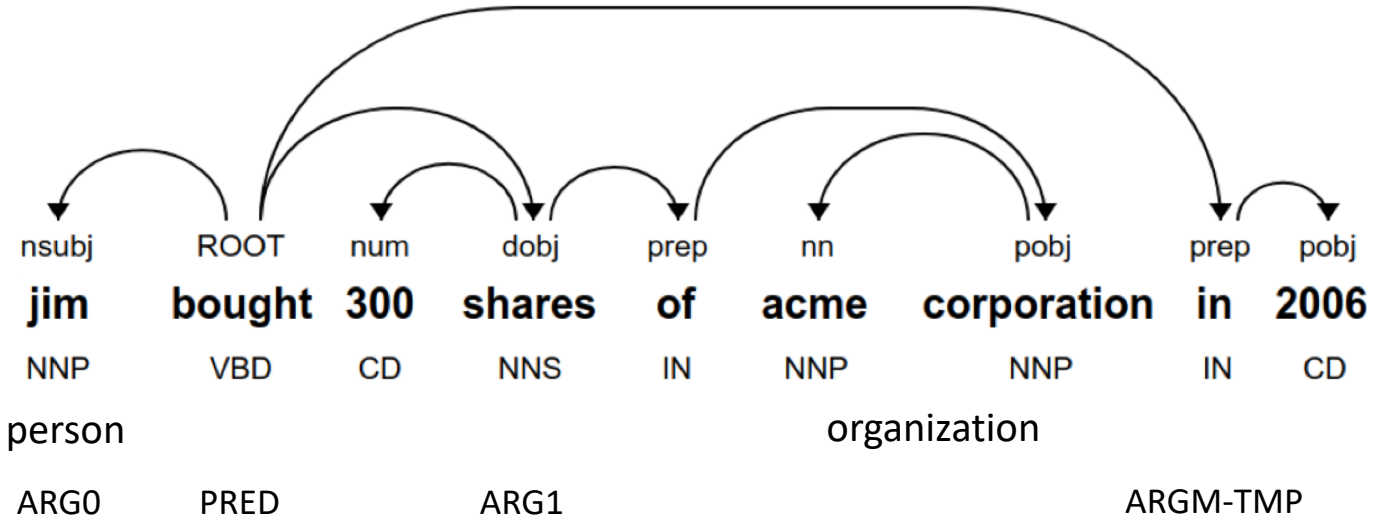


Classical NLP Building Blocks

Pipelined processing:

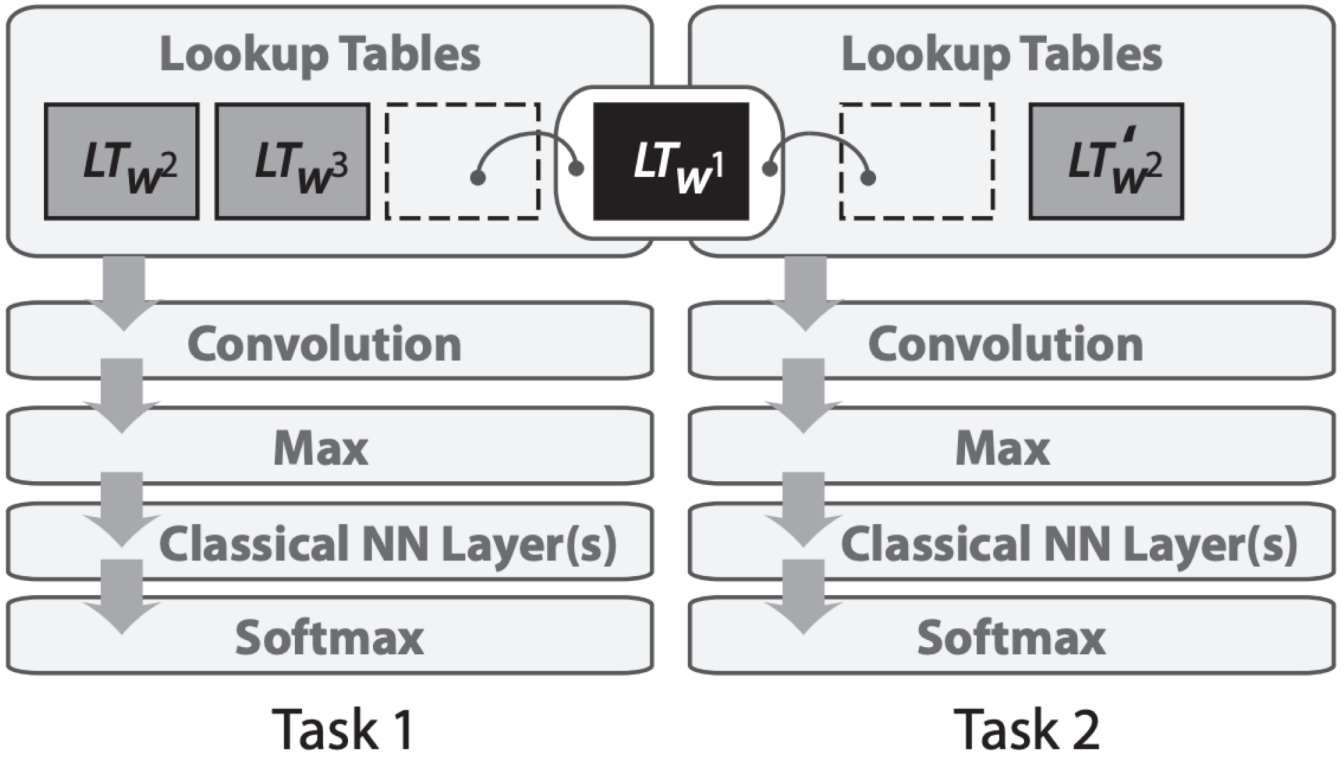
- Part-of-speech tagging
- Named-entity recognition
- Syntactic parsing
- Semantic parsing
- Downstream NLU tasks

- Error in one stage tend to propagate to the next



Multi-task Learning: Collobert & Weston

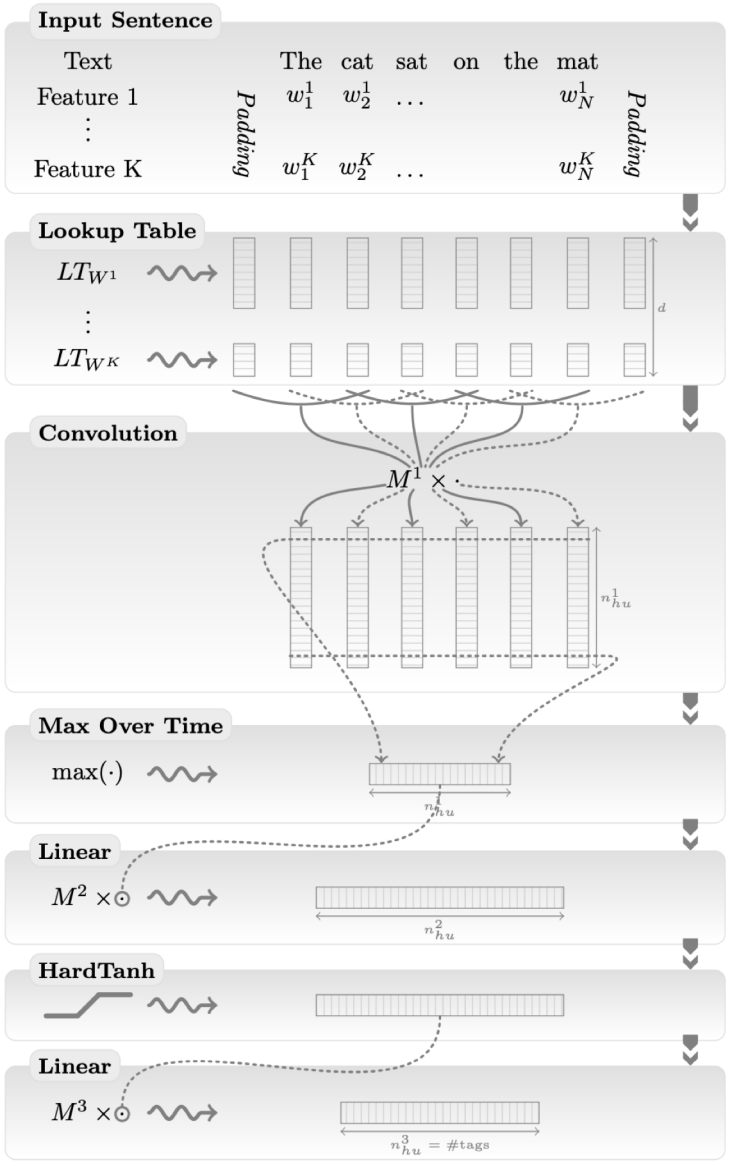
- Training multiple NLP tasks together
  - POS tagging
  - Chunking
  - NER
  - SRL
  - LM
  - WordNet relations
- Shared word embedding vector



Sources:
A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning, 2008
Natural Language Processing (Almost) from Scratch, 2011

Multi-task Learning: Collobert & Weston

- CNN layers embedding entire sentence
- Each word encoded w/ positional feature and



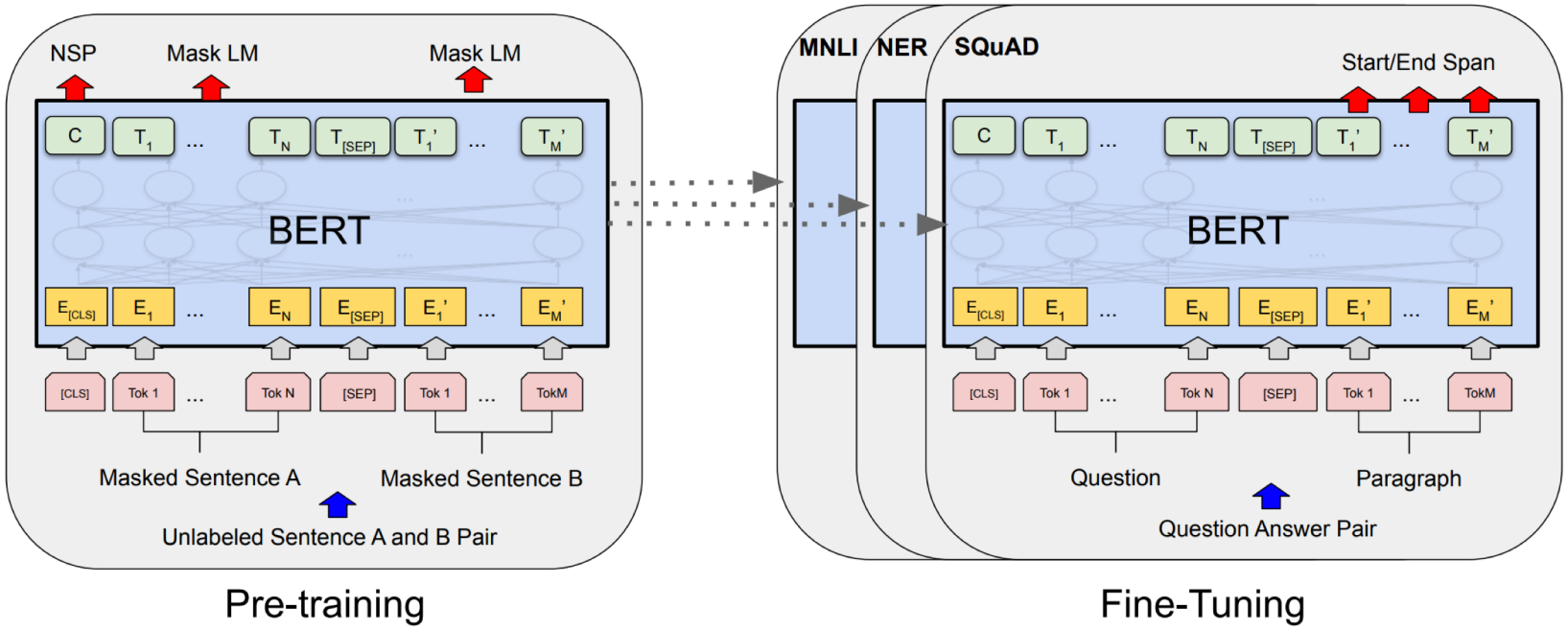
Multi-task Learning: Collobert & Weston

Task		Benchmark	SENNA
Part of Speech (POS)	(Accuracy)	97.24 %	97.29 %
Chunking (CHUNK)	(F1)	94.29 %	94.32 %
Named Entity Recognition (NER)	(F1)	89.31 %	89.59 %
Parse Tree level 0 (PT0)	(F1)	91.94 %	92.25 %
Semantic Role Labeling (SRL)	(F1)	77.92 %	75.49 %

- Results from single model
- State-of-the-art for lower level classification tasks
- Comparable to best CoNLL-2005 results using a single syntactic parse tree
  - Best system uses integer linear programming (ILP) to perform structural inference
  - Most systems trained on syntactic parse tree outputs, previous year's best system on chunking output in low 70s F1.

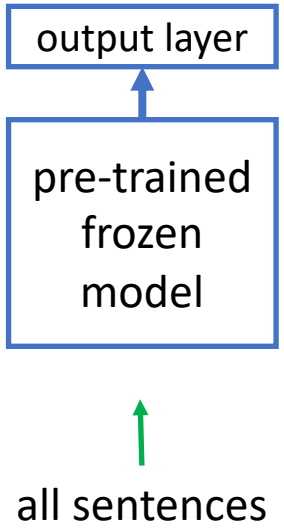
Pre-training and Fine-Tuning: BERT

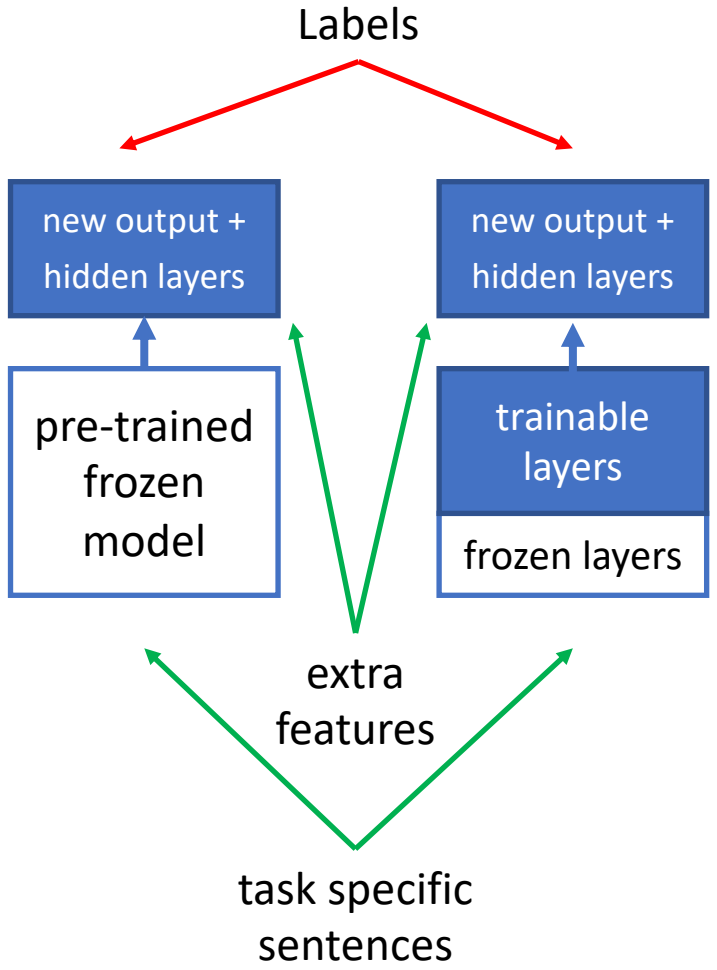
- Using pre-trained LM model to fine-tune on NLU tasks
  - Achieved stat-of-the-art results on 11 GLUE tasks
  - Good perform on new NLP task w/o a lot of task-specific training data



Pre-training and Fine-Tuning

- Start w/ pre-trained model
- Add new output layer to last layer of network
  - Add other layers in between if needed
- Train new layers w/ task specific data
- Unfreeze top (or all) layers of pre-trained model
  - Retrain all trainable layers w/ task specific data





Pre-trained Models

Image models:

- Xception
- VGG
- ResNet
- Inception
- MobileNet

NLP

- Transformer models (Bert, Albert, etc)
- OpenAI GPT-2

Framework model repository

- Tensorflow
- PyTorch
- Keras
- CNTK