Neural Word Embeddings

S. Sathiya Keerthi**

Cloud & Information Services Lab
Microsoft
Mountain View, CA, USA

* Links to pdfs of all mentioned references are in the appendix
**Joint work with Tobias Schnabel (Cornell University) & Rajiv Khanna (UT Austin)

The problem

- Model similarity between textual objects – *words*, *sentences*, *documents*
- This talk will focus on *words*
- In NLP/CL literature this work has been discussed under the topic called *Distributional Similarity Models* (*DSMs*)
- Recently there has been exciting research in *Neural models*
- The aim of this talk is to give an overview of these models

What is an embedding?

- Suppose you have a collection of objects
- The i-th object is represented by an embedding:

$$w_i \in R^d$$

- d is typically small, e.g., in the range 50-1000
- Semantically similar objects are located nearby in the embedding vector space
- *Word embeddings* are also called as *Word vectors*

Compute similarity using embeddings

- Let w_1 and w_2 be the embeddings of two words
- How do we quantify the similarity between the two words?
- Similarity is used in a loose sense – could be synonyms (*midday*, *noon*), hyponym/hypernym (*fruit*, *food*), connected words (*Doctor*, *Hospital*) etc.
- Since all models employ dot products, a (somewhat) natural approach is to use $w_1 \cdot w_2$
- One can also use some normalization, e.g., use *cosine similarity*
- Similarity is usually used to answer questions such as: *Is* w_1 *closer to* w_2 *than* w_3*?*

What are some uses of word embeddings?*

- Computing word similarity, doing better word clustering
- Word classification, e.g., positive versus negative words
- Automatic thesaurus generation – need to use particular kind of similarity that is appropriate for a thesaurus
- Context sensitive spelling correction
- Can be used as features in NLP systems avoiding the need for hand crafted features specific to each NLP problem
- Query expansion – based on words closer to query words in the embedding space. Also applicable to advertising bid terms expansion

* See sec 6.2 of Turney and Pantel (2010)

A tf-idf based approach*

- Representing documents as a vector/bag of words using tf (term frequency) and idf (inverse document frequency) is well known
- To represent a word, you can just invert the idea – represent a word by the documents it is contained in. You can also use concepts dual to tf and idf
- This – and shortened reps via SVD or other means - give a vectorial representation of each word, which can be considered as *word embeddings*
- But the approach is only useful at a gross level

* Landauer et al (1998)

Many problems require finer semantics*

- Authorship attribution
- Author profiling
  - Gender identification
  - Native language identification
- Sentiment analysis
- Detection of illegal behaviors on the web
- Plagiarism detection

* Most folks think doc reps via tf-idf is good for all text classification – but that is not true

How do we build semantic embeddings?

- Labeled datasets are impossible for such a large task
- Plenty of unlabeled textual data is available
- But what is the signal in such data?

"How much do we know at any time?
Much more, or so I believe, than we know we know."
-Agatha Christie, *The Moving Finger*

The distributional hypothesis

- Quotes:
  - *Meaning is use* .. Wittgenstein, 1953
  - *You shall know a word by the company it keeps* .. Firth, 1957

- If two words share similar contexts, we can assume that they have similar meanings

- Comparing meanings reduces to comparing contexts

- Takeaway – Represent each word as a vector of context features

The distributional approach

- Record contexts of words across a large collection of texts (corpus)
- Each word w is represented by a vector of features $x_1, \dots, x_n$
- Each feature x_i records some property of the observed contexts of w
- Words that are found to have similar features are expected to also have similar meaning
- Therefore, the vector representation of words described above is a great choice for word embedding
- Key question: How do we define 'context'?

Pre-processing

Forming *words*

- Raw: `The programmer's programs had been programmed.`
- Tokenized: `the programmer 's programs had been programmed`
- Lemmatized: `the programmer 's program have be program`
- W/ stop-list: `programmer program program`
- Stemmed: `program program program`

Forming word phrases

- Linguistically, it does not make sense to treat *New York* or *Air Canada* as separate words; working with phrases makes a lot of sense

- Mikolov et al (2013a) pick phrases from bigrams using

$$\text{score}(w_i, w_j) = \frac{\text{count}(w_i w_j) - \delta}{\text{count}(w_i) \times \text{count}(w_j)}.$$

- δ prevents infrequent counts to be put down

- Repeat a few passes of this

Down-sampling frequent words

Discard a word w_i with probability

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

where $f(w_i)$ is the frequency of word w_i and t is a threshold

Distributional Semantic Models
(aka Count Models)

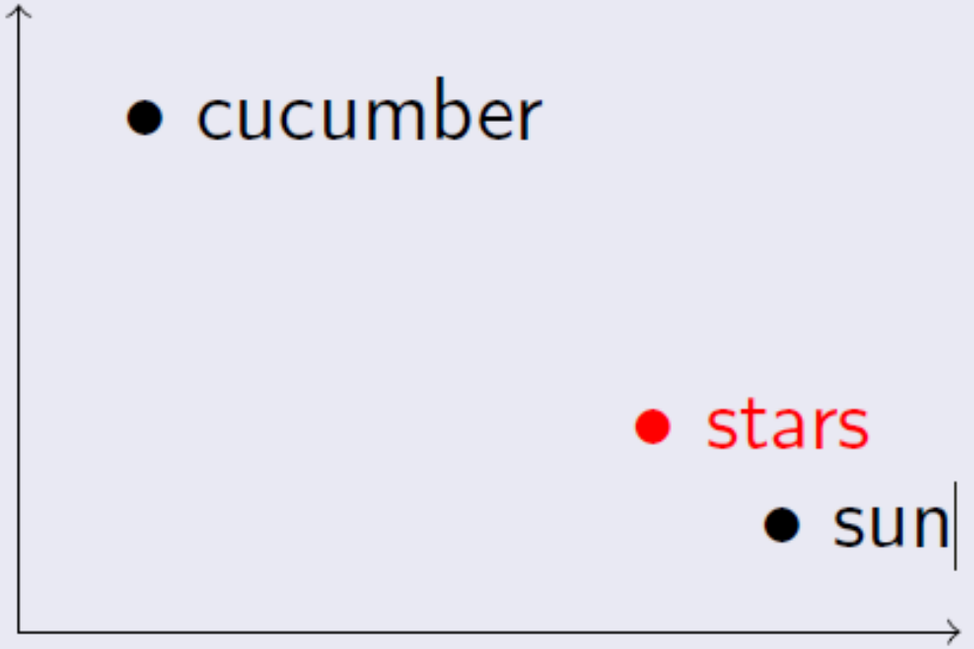
he curtains open and the stars shining in on the barely
ars and the cold , close stars " . And neither of the w
rough the night with the stars shining so brightly , it
made in the light of the stars . It all boils down , wr
 surely under the bright stars , thrilled by ice-white
sun , the seasons of the stars ? Home , alone , Jay pla
m is dazzling snow , the stars have risen full and cold
un and the temple of the stars , driving out of the hug
 in the dark and now the stars rise , full and amber a
bird on the shape of the stars over the trees in front
 But I could n't see the stars or the moon , only the
 they love the sun , the stars and the stars . None of
r the light of the shiny stars . The plash of flowing w
man 's first look at the stars ; various exhibits , aer
rief information on both stars and constellations, inc

* Thanks – Grefenstette et al (2014)

Construct vector representations

	shining	bright	trees	dark	look
stars	38	45	2	27	12

Similarity in meaning as vector similarity



Variations in the definition of *co-occurrence*

Co-occurrence with words, window of size 2, scaling by distance to target:

... two *[intensely bright stars in the]* night sky ...

	intensely	bright	in	the
stars	0.5	1	1	0.5

Variations in the type of context features

	Doc1	Doc2	Doc3
stars	38	45	2

	$\xleftarrow{dobj}$ see	$\xrightarrow{mod}$ bright	$\xrightarrow{mod}$ shiny
stars	38	45	44

	The nearest • to Earth	stories of • and their
stars	12	10

Context = the Grammatical relations of the word to other words in the sentence

Example describing contexts*

Giggs|NNP *scored*|VBD *the*|DT *first*|JJ *goal*|NN *of*|IN *the*|DT *football*|NN *tournament*|NN *at*|IN *Wembley*|NNP ,|, *North*|NNP *London*|NNP .|.

Contextual elements for target word *goal* using a 7-word window method:
{*scored*, *the*, *first*, *of*, *football*}

Contextual elements with parts-of-speech:
{*scored*|VBD, *the*|DET, *first*|JJ, *of*|IN, *football*|NN}

Contextual elements with direction (L for left, R for right):
{*scored*|L, *the*|L, *first*|L, *of*|R, *the*|R, *football*|R}

Contextual elements with position (e.g. 1L is 1 word to the left):
{*scored*|3L, *the*|2L, *first*|1L, *of*|1R, *the*|2R, *football*|3R}

Contextual elements as grammatical relations:
{*first*|`ncmod`, *the*|`det`, *scored*|`dobj`}

* Clark (2015)

Weighting of context elements

- Give more weight to surprising (rare) contexts and less weight to expected (frequent) contexts
  - Example: For the word *rat*, the contexts *dissect* and *exterminate* are more discriminative than words such as *have* and *like*

- One could use *tf-idf* ideas here

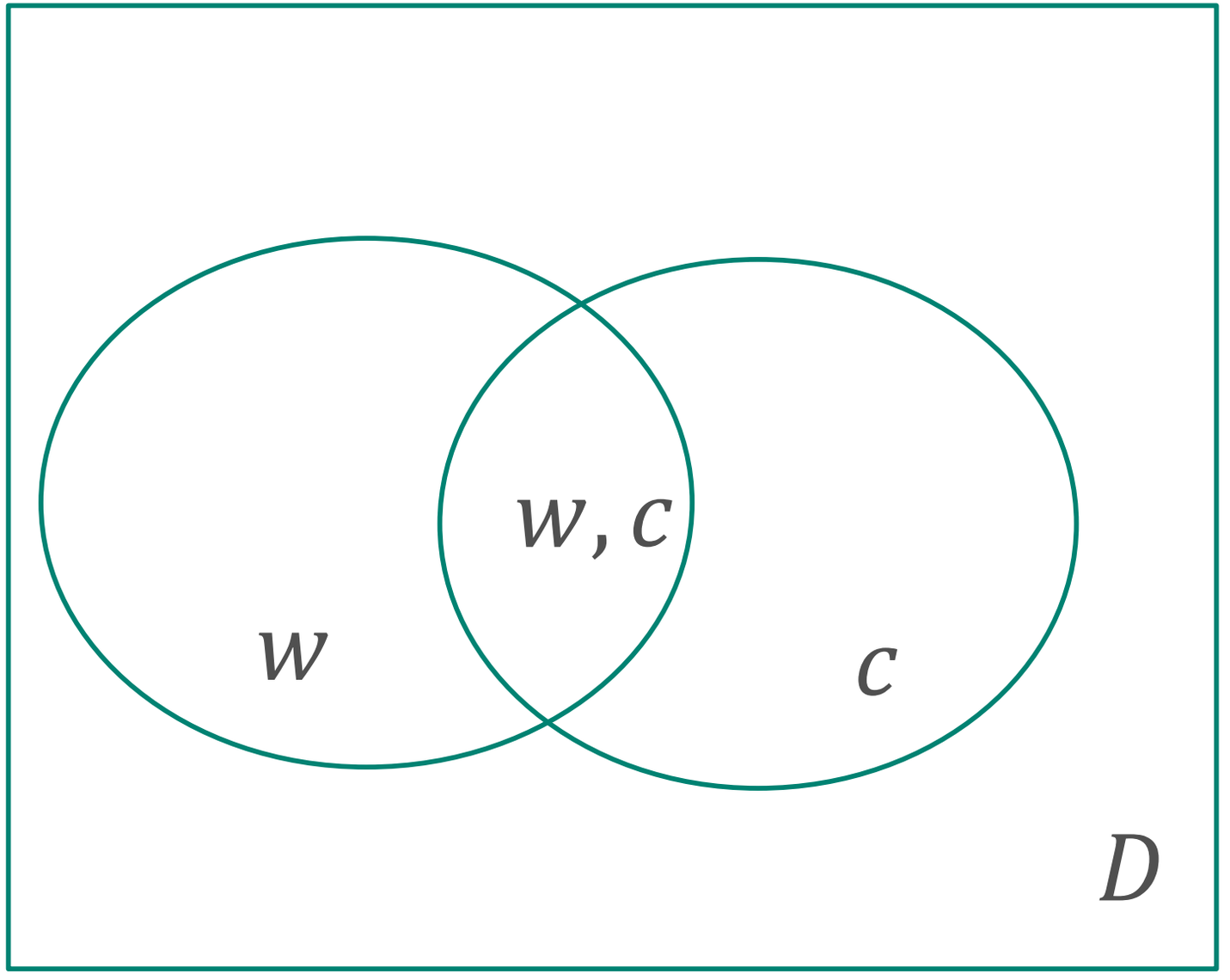
- A range of different tests of statistics are used; e.g. pointwise mutual Information (PMI), log odds ratio, t-test, log likelihood

- PMI and PPMI (Positive PMI) have shown up well in evaluations – see Bullinaria et al, 2011, 2012 for extensive computational experiments

Re-weight the counts using corpus-level statistics to reflect co-occurrence *significance*.

Point-wise Mutual Information (PMI)

$$\mathrm{PMI}(\mathrm{target}, \mathrm{ctxt}) = \log \frac{\mathrm{P}(\mathrm{target}, \mathrm{ctxt})}{\mathrm{P}(\mathrm{target})\mathrm{P}(\mathrm{ctxt})}$$



$$P(w,c) = \frac{\#w,c}{\#D}$$

$$P(w) = \frac{\#w}{\#D}$$

$$P(c) = \frac{\#c}{\#D}$$

$$PMI = \log P(w,c) - \log P(w) - \log P(c)$$

More on PMI…

- Very low values of $\#(w, c)$ is noise and so a simple idea is to leave out negative *PMI* values from the vectorial representation of w. This is *PPMI* (*Positive PMI*)

- More aggressive: Consider only *PMI* values above $\log k$ where k is a threshold (hyperparameter) – *Shifted PPMI*

- SVD of the *PMI* matrix (#words x #ctxt) may be used to reduce the dimension of the embedding

Neural Models
(aka Predict Models)

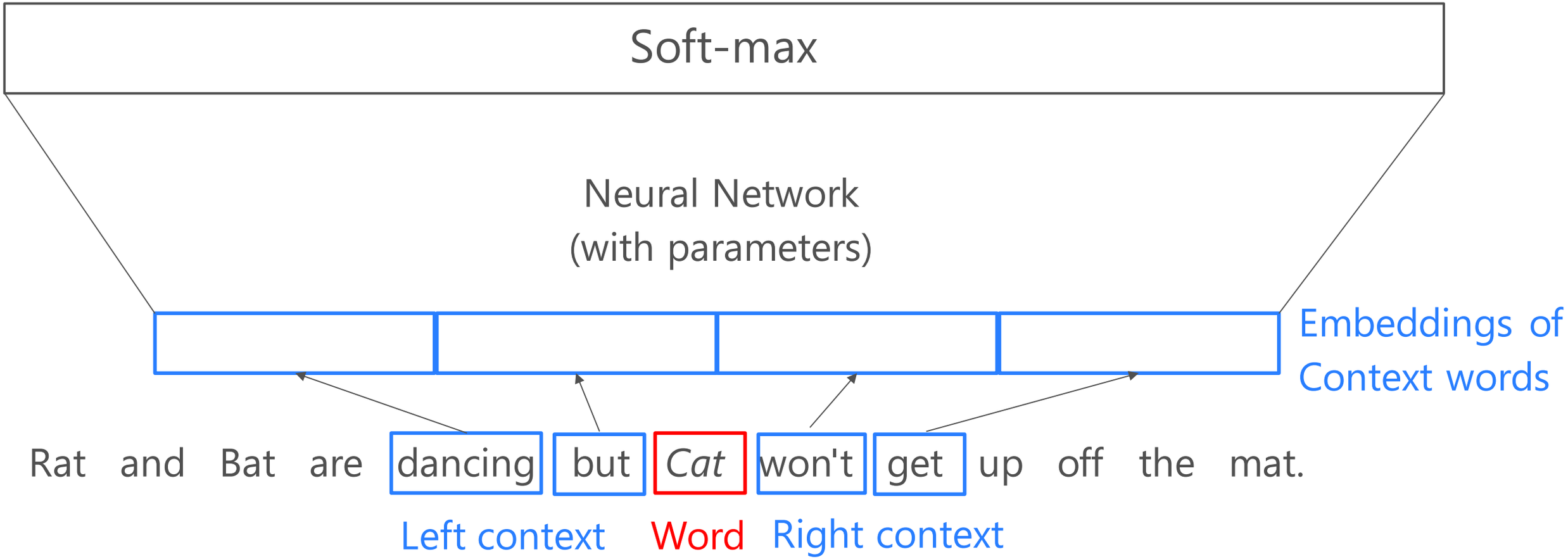
History

- Origins in the work of Bengio et al (2003) – Probabilistic Language models
- Mnih & Hinton (2007) proposed the Log bilinear scoring model
- Mikolov & colleagues simplified the model in various ways for forming word embeddings

CBOW

$$P(Cat) = \exp(s(Cat))/Z \text{ where } Z = \sum_w \exp(s(w))$$

Each word w in vocabulary gets a score $s(w)$



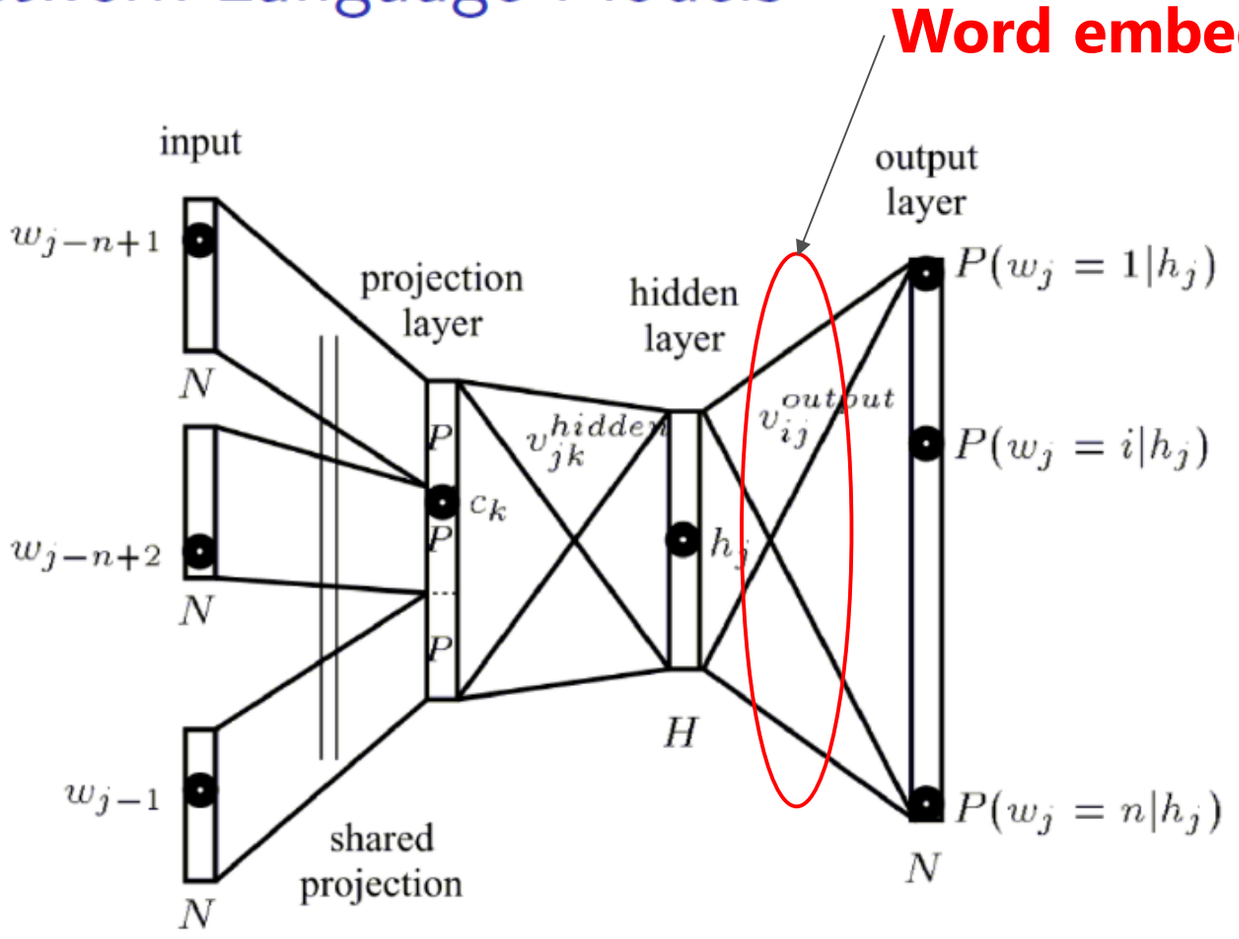
Context embeddings and neural net parameters are all unknowns that are tuned

But where are the word embeddings??

Training process

- Apply the model described treating every word occurrence in the data as a training example
- Since no manual labeling is involved, it is easy to collect billions of examples
- Minimize the total Negative Log Likelihood formed with the soft-max probabilities
- Stochastic gradient descent is usually used
- After training, there is no role for context embeddings – scope for research

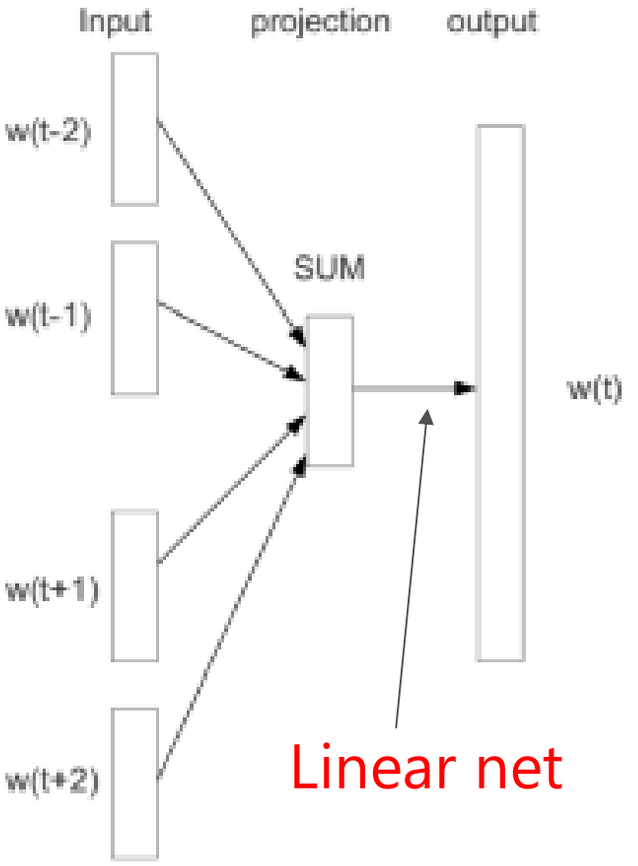
Neural Network Language Models



Bengio, Y., Schwenk, H., Sencal, J. S., Morin, F., & Gauvain, J. L. (2006). Neural probabilistic language models. In Innovations in Machine Learning (pp. 137-186). Springer Berlin Heidelberg.

CBOW

- CBOW = Continuous bag-of-words model
- Sums the embeddings of context words
- After that it is just a linear net to form word scores
- The parameters of the linear net give the word embeddings



* Mikolov et al (2013a,b)

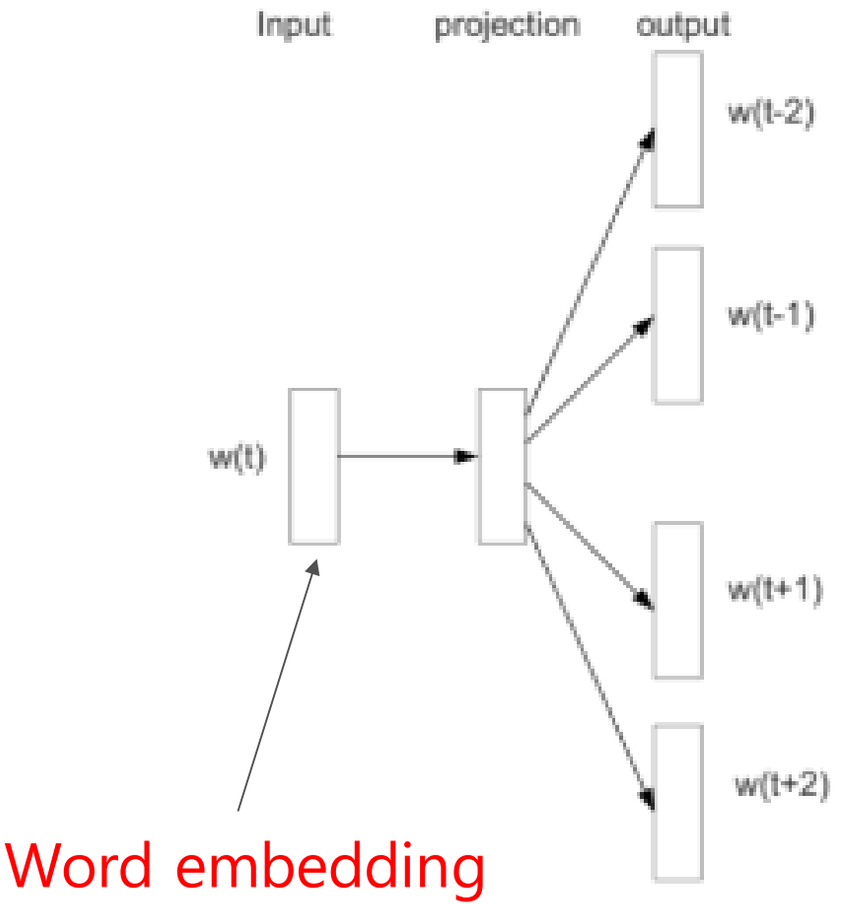
Log bilinear (LBL) model*

- For a word occurrence w, let C be the set of context words
- Let α_c be the context embedding for context word c
- Let β_w be the word embedding for word w
- Score $s(w) = (\sum_{c \in C} \alpha_c) \cdot \beta_w$ (LBL is more general, actually)
- Now use this to form the soft-max probability calculation

* Mnih and Hinton (2007)

Skip-gram model

- Reverse CBOW – Use the word to predict the context words
- Complication: Output is a sequence
- To simplify, the context words prediction is treated as a separate (but identical) set of soft-maxes



* Mikolov et al (2013a,b)

Computational bottleneck of soft-max

- Normalization term Z involves a summation over all words in the vocabulary
- Large vocabularies
  - Wikipedia - 2M words, 3.2B word occurrences
  - Google News – 1M words, 6B word occurrences
- Modeling probabilities is not so important, unlike language models
- Three methods for efficient handling
  - Hierarchical soft-max
  - Negative sampling
  - Noise-contrastive estimation

Hierarchical soft-max*

- Form a hierarchical clustering of the vocabulary – typically done according to frequency – so that frequently occurring words have short paths to leaves – *Huffman coding*
- Thus, each leaf corresponds to one word
- At each internal node, a sigmoid operating on the scores is used to decide which direction to move – the probabilities associated with its child nodes
- Products of probabilities along a traversed path from the root to a leaf gives the probability of the word associated with the leaf
- Thus there are only $\log V$ computations for dealing with each example, where V is the vocabulary size

* Morin and Bengio (2005); Mikolov et al (2013a,b)

Negative sampling

- Each training instance has a word w that is to be predicted – take that as the single positive example
- Randomly choose k other words, $n_1, \ldots, n_k$ and treat them as negative examples
- Let $\sigma(x)$ be the sigmoid. Take $\sigma(s(w))$ as the likelihood of w and $\sigma(-s(n_1)), \ldots, \sigma(-s(n_k))$ as the likelihoods associated with the negative words. Use these to form the training objective
- The probabilistic modeling here is hand-wavy, but it is effective for forming word embeddings

* Mikolov et al (2013a)

Noise-contrastive estimation

- Choose a noise distribution (usually some kind of a mean distribution over all words)
- Choose negative examples according to the noise distribution
- Unlike Negative sampling, this method enjoys some nice asymptotic properties

* Mnih and Kavukcuoglu (2013)

Speedup

- Hierarchical soft-max is an order of magnitude faster than soft-max
- Training time reduced from days to a few hours
- Negative sampling and Noise-contrastive estimation are also very fast

Word2vec

- Codes:
  - Google - http://code.google.com/p/word2vec/
  - Gensim - http://radimrehurek.com/gensim/models/word2vec.html

- Tools for training word vectors using CBOW and Skip-Gram models – both negative sampling and hierarchical softmax are supported

- Optimized for very large datasets - >billions of training (word, context) pairs

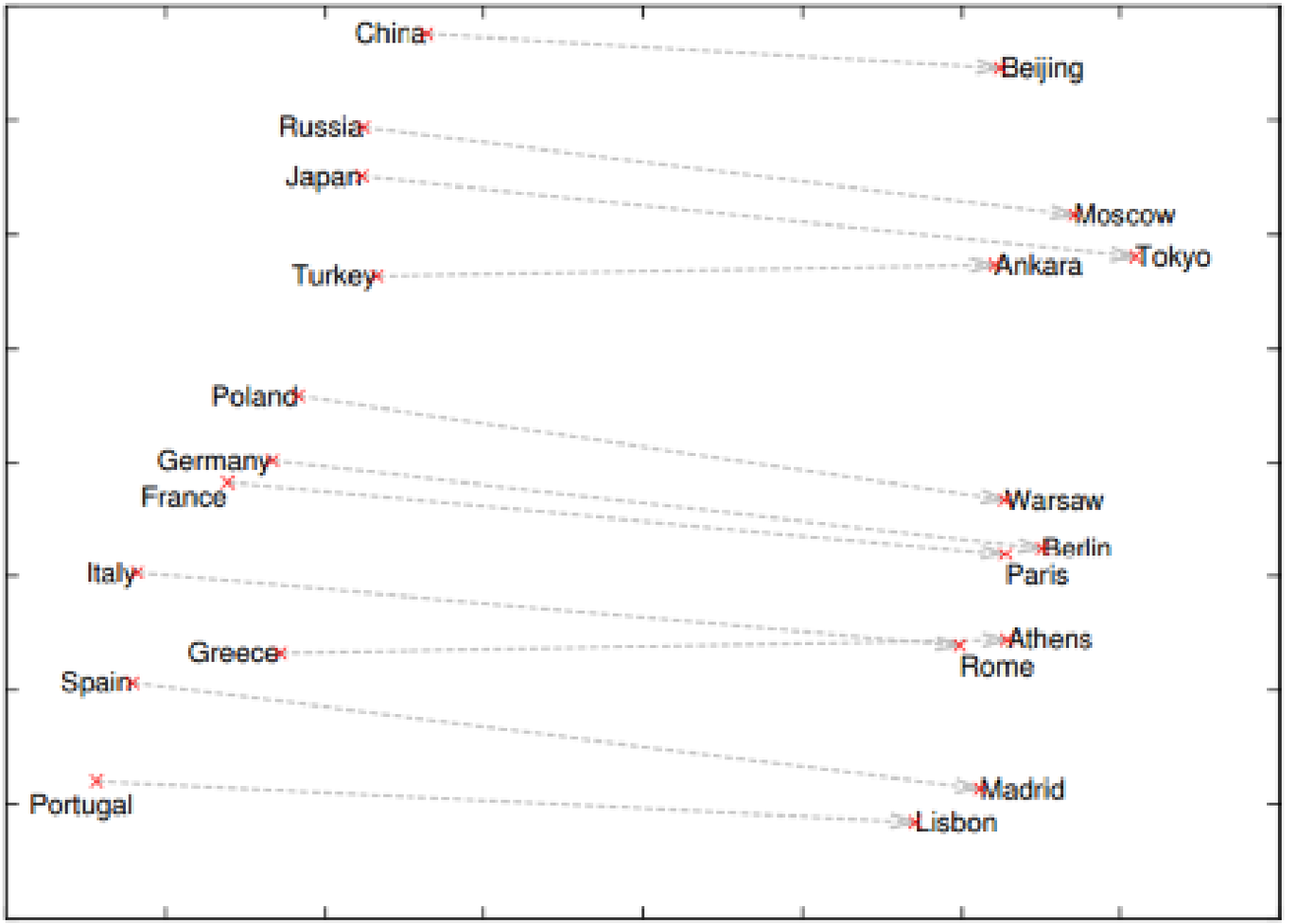
- Pre-trained word vectors are also available to play with

Linguistic Regularities

- Word vectors from neural models exhibit linear structure that enables checking analogies with vector arithmetic

- Given two examples of a relationship, e.g.,
  - *Beijing* is to *China* as *Moscow* is to *Russia*
  - The closest point to vec[*Beijing*] + (vec[*Russia*] – vec[*Moscow*]) will be vec[*China*]

Country and Capital vectors projected by PCA



Tomas Mikolov, COLING 2014

* Mikolov, Yih and Zweig (2013); Mikolov et al (2013a,b)

More examples

Expression	*Nearest token*
Paris - France + Italy	Rome
bigger - big + cold	colder
sushi - Japan + Germany	bratwurst
Cu - copper + gold	Au
Windows - Microsoft + Google	Android
Montreal Canadiens - Montreal + Toronto	Toronto Maple Leafs

Non-linear networks satisfy such linear linguistic regularities poorly!

Evaluation

Dataset 1

- MSR dataset with 8K questions, mostly focuses on syntax - examples
  - *good:better rough: _____*
  - *good:best rough: _____*
  - *better:best rougher: _____*
  - *year:years law: _____*
  - *see:saw return: _____*

Dataset 2

- Google word-based dataset, almost 20K questions, focuses on both syntax and semantics
  - *Athens:Greece* *Oslo: _____*
  - *Angola:kwanza* *Iran: _____*
  - *brother:sister* *grandson: _____*
  - *possibly:impossibly* *ethical: _____*
  - *walking:walked* *swimming: _____*

Dataset 3

- Google phrase-based dataset, focuses on semantics
  - *New York:New York Times Baltimore: _____*
  - *Boston:Boston Bruins Montreal: _____*
  - *Detroit:Detroit Pistons Toronto: _____*
  - *Austria:Austrian Airlines Spain: _____*
  - *Steve Ballmer:Microsoft Larry Page: _____*

Some performance numbers

These are two different ways of checking analogies

Objective	Representation	MSR	GOOGLE
3COSADD	Embedding	53.98%	62.70%
	Explicit	29.04%	45.05%
3COSMUL	Embedding	**59.09%**	66.72%
	Explicit	56.83%	**68.24%**

Table 3: Comparison of **3COSADD** and **3COSMUL**.

The linguistic regularities are not exclusive property of neural net based representations: *Linguistic Regularities in Sparse and Explicit Word Representations* (Levy & Goldberg, 2014)

Count model

Predict model

* Levy and Goldberg (2014)

Better to *Predict* than *Count*

	rg	ws	wss	wsr	men	toefl	ap	esslli	battig	up	mcrae	an	ansyn	ansem
best setup on each task														
cnt	74	62	70	59	72	76	66	84	98	41	27	49	43	60
pre	84	75	**80**	**70**	**80**	91	75	86	**99**	41	28	**68**	**71**	**66**
other models														
soa	**86**	**81**	77	62	76	**100**	**79**	**91**	96	**60**	**32**	61	64	61
dm	82	35	60	13	42	77	76	84	94	51	29	NA	NA	NA
cw	48	48	61	38	57	56	58	61	70	28	15	11	12	9

Table 2: Performance of count (cnt), predict (pre), dm and cw models on all tasks.

Comparison to traditional vector space models: *Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors* (Baroni et al, 2014)

* Baroni et al (2014) – The datasets of row 1 are expanded in the appendix

Extensions

Connecting *Count* and *Predict* Models

- Levy and Goldberg (2014c) have established the following
  - SGNS (Skip-gram with negative sampling) is a count based model
  - An extreme case of it gives Shifted PPMI

- We have done a more detailed analysis – indicates that overfitting could be the reason for the inferior performance of PMI methods

- Regularization can help with this – study underway

- The low dimensionality of the embedding space used by SGNS gives an automatic regularization due to reduced number of embedding parameters

Supervised models

- For a given supervised learning problem, it is appropriate to
  - Use the extra labeled data to guide the embedding vectors
  - Make the right choice of context and other hyper-parameters

Multiple word prototypes

- Polysemy. Take the word *Plant*. Using only one representation results in *Plant* getting an embedding that is the average of its different semantic versions relating to biology, placement, manufacturing and power generation.

- Need ways of identifying polysemous words and allow such words to have multiple word vectors

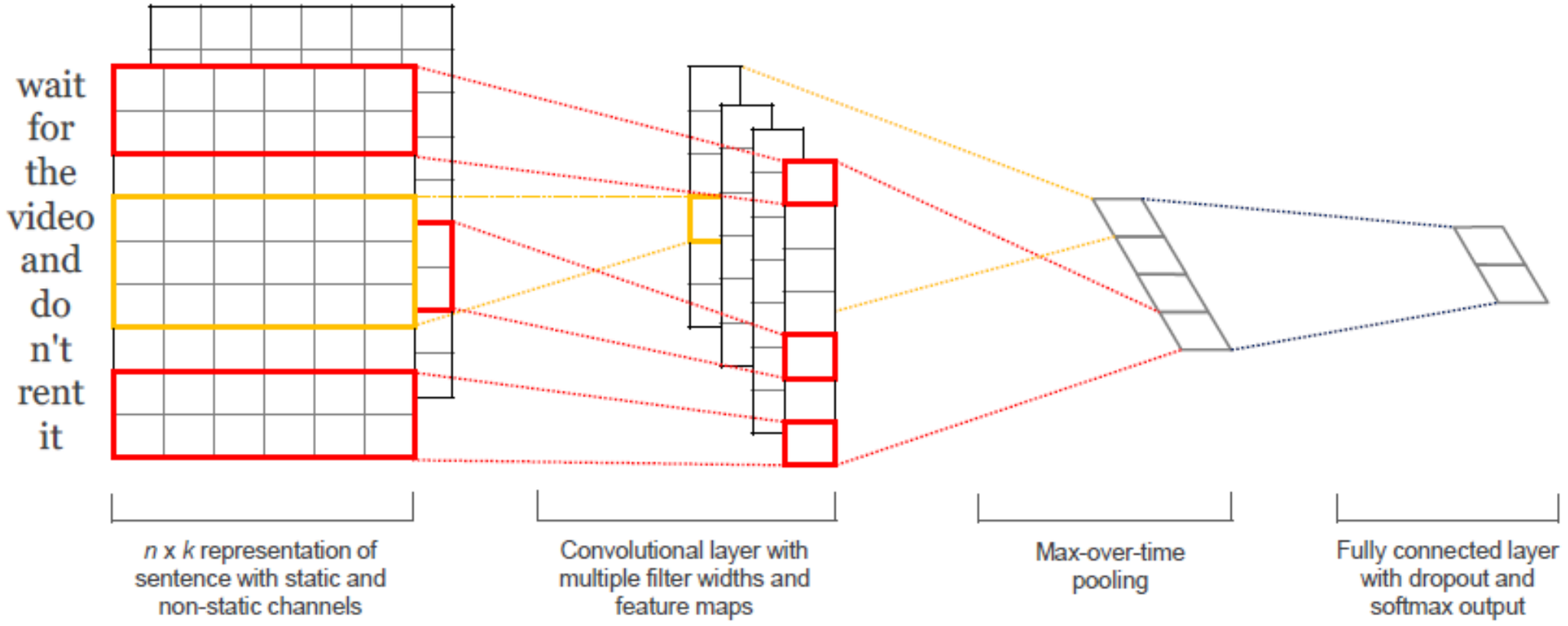
* Neelakantan et al (2014)

Compositional models

- Creation of vectors/embeddings for phrases, sentences and documents
- *Man killed dog* to be much closer to *Man murdered cat* than *Man was killed by dog*
- Much of the existing work seems to be from the Distributional literature
- Convolutional neural network ideas are being suggested
- A lot of scope for neural models here

ConvNets for sentence classification

- Example: Is a sentence about a movie review express positive sentiment?



Excellent performance on several benchmark datasets!

* Yoon Kim (2014)

Concluding comments

- DSMs (Count models) have been around for some time (10-15 years). Predict models give a freshening view.
- Predict models seem to be robust to many levels of simplification.
- There is mostly *It works* kind of papers. It would be useful to have *Why It works* kind of papers.
- Good and clean evaluations are very important for saying one method is better than another.

Some open problems that you can easily start working on

- Predict model (Log bilinear) uses dot products between embeddings of words and context words. But finally dot product of word embeddings is used for similarity. What is the basis?
- Also, how did the linguistic regularities come about? Formulation had nothing about it.
- Can you find ways of adding domain knowledge, e.g., from Wordnet?
- How do the methods work on small corpuses?
- Do detailed experiments on the effect of various context modelings on the quality of the word vectors.
- Find a good semi-supervised formulation that mixes the Log bilinear model with a supervised model that uses word vectors.
- Composition of word vectors to deal with higher objects such as sentences, paragraphs and documents is a more difficult space, but very open right now. It is hot because of its usefulness in machine translation, question answering, document summarization, non-thematic classification problems such as sentiment analysis, etc.

Appendix

References

1. M. Baroni et al (2014) – Don’t count, predict.. – ACL [pdf]
2. Y. Bengio et al (2003) – A neural probabilistic.. – JMLR [pdf]
3. J.A. Bullinaria and J.P. Levy (2007) – Extracting semantic representations.. 1 – Behavior Research Methods, 39, 510-526 [pdf]
4. J.A. Bullinaria and J.P. Levy (2012) – Extracting semantic representations.. 2 – Behavior Research Methods, 44, 890-907 [pdf]
5. S. Clark (2015) – Vector space models.. – Handbook of Contemp. Semantics [pdf]
6. E. Grefenstette (2014) - New Directions in Vector Space Models of Meaning - ACL Tutorial [pdf]
7. T.K. Landauer et al (1998) – Learning human-like knowledge.. – NIPS [pdf]
8. O. Levy and Y. Goldberg (2014a) - word2vec Explained.. - arXiv [pdf]
9. O. Levy and Y. Goldberg (2014b) - Linguistic Regularities in Sparse.. - CoNLL [pdf]
10. O. Levy and Y. Goldberg (2014c) - Neural Word Embeddings as Implicit.. - NIPS [pdf]

References..

11. T. Mikolov, W.T. Yih and G. Zweig (2013) – Linguistic regularities.. – HLT-NAACL [pdf]
12. T. Mikolov et al (2013a) Distributed representations.. – NIPS [pdf]
13. T. Mikolov et al (2013b) Efficient estimation of word.. – arXiv – [pdf]
14. A. Mnih and G. Hinton (2007) - A Scalable Hierarchical.. – NIPS [pdf]
15. A. Mnih and K. Kavukcuoglu (2013) - Learning word embeddings – NIPS [pdf]
16. F. Morin and Y. Bengio (2005) – Hierarchical probabilistic neural.. – AISTATS [pdf]
17. A. Neelakantan et al (2014) - Efficient Non-parametric Estimation of Multiple Embeddings… - EMNLP [pdf]
18. P.D. Turney and P. Pantel (2010) - From frequency to meaning.. – JAIR [pdf]
19. Yoon Kim (2014) – Convolutional neural networks for.. – EMNLP [pdf]

Evaluation datasets*

- Semantic relatedness: In this task, humans were asked to rate the relatedness of two words, and the system correlation with these scores is measured.
  - **rg**: 65 noun pairs by Rubenstein & Goodenough (1965)
  - **ws**: Wordsim353, 353 word pairs by Finkelstein et al. (2002)
  - **wss**: Subset of Wordsim353 focused on similarity (Agirre et al., 2009)
  - **wsr**: Subset of Wordsim353 focused on relatedness (Agirre et al., 2009)
  - **men**: 1000 word pairs by Bruni et al. (2014)

- Synonym detection: The system needs to find the correct synonym for a word.
  - **toefl**: 80 multiple-choice questions with 4 synonym candidates (Landauer & Dumais, 1997).

* Baroni et al (2014)

- Concept categorization: Given a set of concepts, the system needs to group them into semantic categories.
  - **ap**: 402 concepts in 21 categories (Almuhareb, 2006)
  - **esslli**: 44 concepts in 6 categories (Baroni et al., 2008)
  - **battig**: 83 concepts in 10 categories (Baroni et al., 2010)

- Selectional preferences: Using some examples, the system needs to decide how likely a noun is to be a subject or object for a specific verb.
  - **up**: 221 word pairs (Pado, 2007)
  - **mcrae**: 100 noun-verb pairs (McRae et al., 1998)

- Analogy: Using examples, the system needs to find a suitable word to solve analogy questions, such as: "X is to 'uncle' as 'sister' is to 'brother'"
  - **an**: ~19,500 analogy questions (Mikolov et al., 2013b)
  - **ansyn**: Subset of the analogy questions, focused on syntactic analogies
  - **ansem**: Subset of the analogy questions, focused on semantic analogies