PARSEME

# Deep Learning and MWE

Giuseppe Attardi
Dipartimento di Informatica
Università di Pisa

# Welcome to Malta

This is a non smoking airport

# Statistical Machine Learning

- Training on large document collections
- Requires ability to process Big Data
  - If we used same algorithms 10 years ago they would still be running
- The Unreasonable Effectiveness of Big Data

# Supervised Statistical ML Methods

- Devise a set of features to represent data:

  $\phi(x) \in \mathbb{R}^{\mathcal{D}}$ and weights $w_k \in \mathbb{R}^{\mathcal{D}}$

- Objective function

  $f(x) = \text{argmax}_k \, w_k \, \phi(x)$

- Minimize error wrt training examples
- Freed us from devising rules or algorithms
- Required creation of annotated training corpora
- Imposed the tyranny of feature engineering

# Deep Neural Network Model

**Output layer**

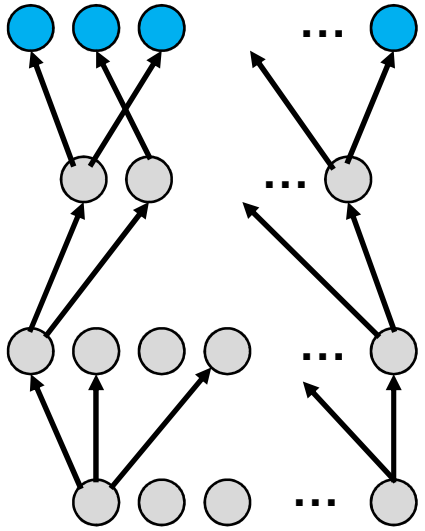
Prediction of target

**Hidden layers**

Learn more abstract representations

**Input layer**

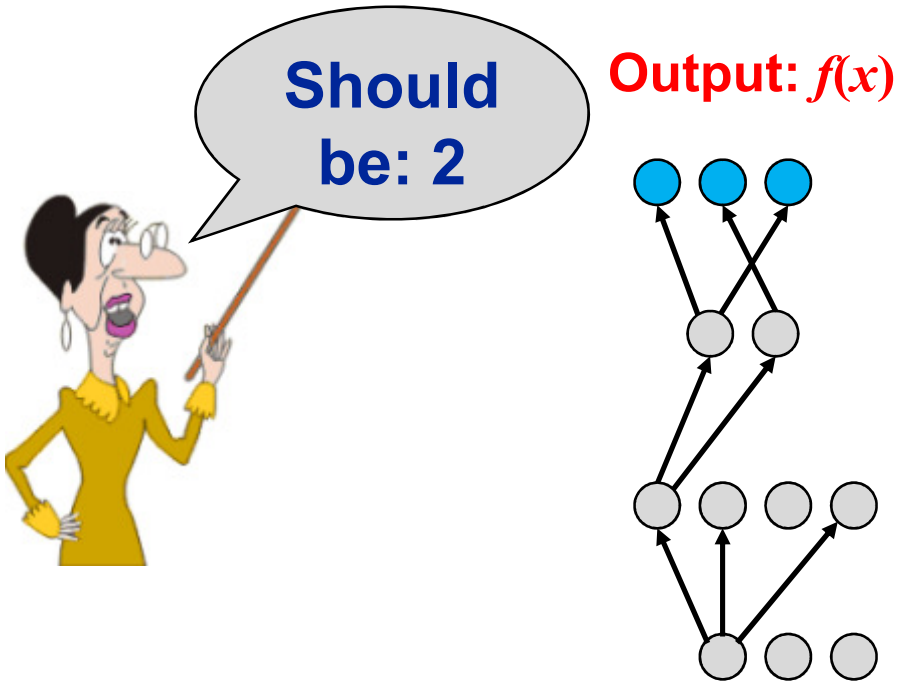
Raw input



# Deep Learning Breakthrough: 2006

- Unsupervised learning of shallow features from large amounts of unannotated data
- Features are tuned to specific tasks with second stage of supervised learning

# Supervised Fine Tuning




Courtesy: G. Hinton

# Vector Representation of Words

- From discrete to distributed representation
- Word meanings are dense vectors of weights in a high dimensional space
- Algebraic properties
- Background
  - Philosophy: Hume, Wittgenstein
  - Linguistics: Firth, Harris
  - Statistics ML: Feature vectors

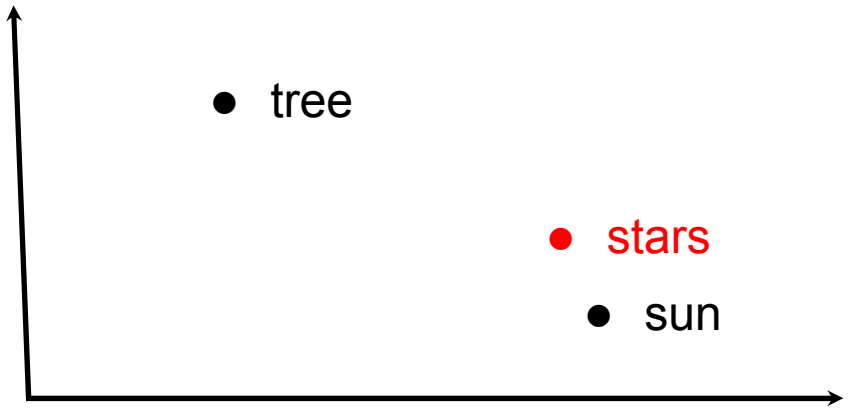
”You shall know a word by the company it keeps” (Firth, 1957).

# Distributional Semantics

- Co-occurrence counts

| | shining | bright | trees | dark | look |
|---|---|---|---|---|---|
| stars | 38 | 45 | 2 | 27 | 12 |

- High dimensional sparse vectors
- Similarity in meaning as vector similarity



# Co-occurrence Vectors

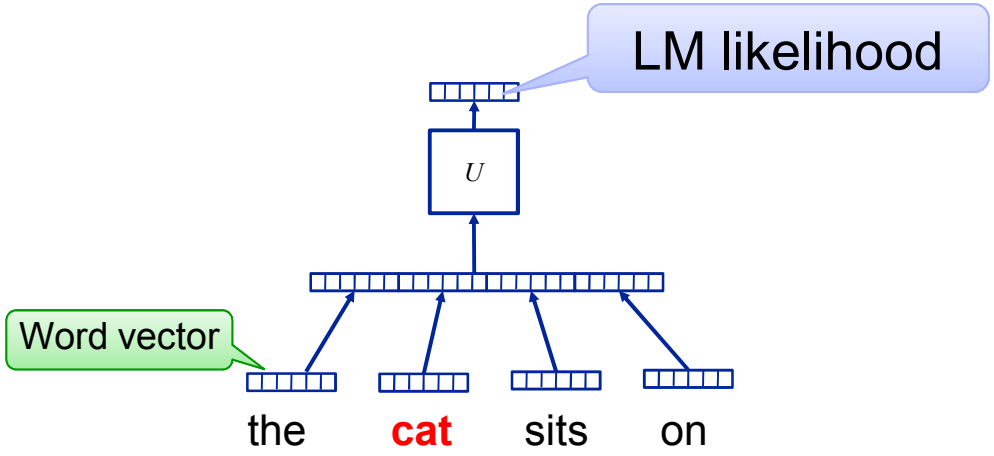
| FRANCE<br>454 | JESUS<br>1973 | XBOX<br>6909 | REDDISH<br>11724 | SCRATCHED<br>29869 | MEGABITS<br>87025 |
|---|---|---|---|---|---|
| PERSUADE | THICKETS | DECADENT | WIDESCREEN | ODD | PPA |
| FAW | SAVARY | DIVO | ANTICA | ANCHIETA | UDDIN |
| BLACKSTOCK | SYMPATHETIC | VERUS | SHABBY | EMIGRATION | BIOLOGICALLY |
| GIORGI | JFK | OXIDE | AWE | MARKING | KAYAK |
| SHAFFEED | KHWARAZM | URBINA | THUD | HEUER | MCLARENS |
| RUMELLA | STATIONERY | EPOS | OCCUPANT | SAMBHAJI | GLADWIN |
| PLANUM | GSNUMBER | EGLINTON | REVISED | WORSHIPPERS | CENTRALLY |
| GOA'ULD | OPERATOR | EDGING | LEAVENED | RITSUKO | INDONESIA |
| COLLATION | OPERATOR | FRG | PANDIONIDAE | LIFELESS | MONEO |
| BACHA | W.J. | NAMSOS | SHIRT | MAHAN | NILGRIS |

neighboring words are **not** semantically related

# Techniques for Creating Word Embeddings

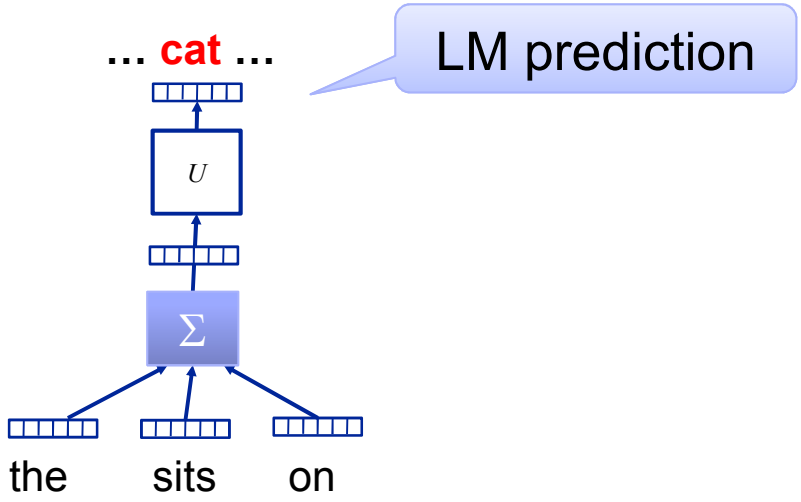
- Collobert et al.
  - SENNA
  - Polyglot
  - DeepNL
- Mikolov et al.
  - word2vec
- Lebret & Collobert
  - DeepNL
- Socher & Manning
  - GloVe

# Neural Network Language Model




Expensive to train:

- 3-4 weeks on Wikipedia




Quick to train:

- 40 min.on Wikipedia
- tricks:
  - parallelism
  - avoid synchronization

# Lots of Unlabeled Data

- Language Model
  - Corpus: 2 B words
  - Dictionary: 130,000 most frequent words
  - 4 weeks of training
- Parallel + CUDA algorithm
  - 40 minutes

# Word Embeddings

| FRANCE | JESUS | XBOX | REDDISH | SCRATCHED | MEGABITS |
|---|---|---|---|---|---|
| 454 | 1973 | 6909 | 11724 | 29869 | 87025 |
| AUSTRIA | GOD | AMIGA | GREENISH | NAILED | OCTETS |
| BELGIUM | SATI | PLAYSTATION | BLUISH | SMASHED | MB/S |
| GERMANY | CHRIST | MSX | PINKISH | PUNCHED | BIT/S |
| ITALY | SATAN | IPOD | PURPLISH | POPPED | BAUD |
| GREECE | KALI | SEGA | BROWNISH | CRIMPED | CARATS |
| SWEDEN | INDRA | PSNUMBER | GREYISH | SCRAPED | KBIT/S |
| NORWAY | VISHNU | HD | GRAYISH | SCREWED | MEGAHERTZ |
| EUROPE | ANANDA | DREAMCAST | WHITISH | SECTIONED | MEGAPIXELS |
| HUNGARY | PARVATI | GEFORCE | SILVERY | SLASHED | GBIT/S |
| SWITZERLAND | GRACE | CAPCOM | YELLOWISH | RIPPED | AMPERES |

neighboring words are semantically related

# Back to Co-occurrence Counts

$$P(w_t, w_{t-i} \dots w_{t-1}) = \frac{P(w_t, w_{t-i} \dots w_{t-1})}{P(w_{t-i} \dots w_{t-1})}$$

| | breeds | computing | cover | food | is | meat | named | as |
|---|---|---|---|---|---|---|---|---|
| cat | 0.04 | 0.0 | 0.0 | 0.13 | 0.53 | 0.02 | 0.16 | 0.1 |
| dog | 0.11 | 0.0 | 0.0 | 0.12 | 0.39 | 0.06 | 0.18 | 0.17 |
| cloud | 0/0 | 0.29 | 0.19 | 0.0 | 0.12 | 0.0 | 0.0 | 0.4 |

- Big matrix |V| × |V| (~ 100k × 100k)
- Dimensionality reduction:
  - Principal Component Analysis, Hellinger PCA, SVD
- Reduce to:100k × 50, 100k × 100

# Weighting

Weight the counts using corpus-level statistics to reflect co-occurrence significance

**Pointwise Mutual Information**

$$PMI(w_t, w_{t-i} \dots w_{t-1}) = \log \frac{P(w_t, w_{t-i} \dots w_{t-1})}{P(w_t) P(w_t, w_{t-i} \dots w_{t-1})}$$

# Online Demos

- Polyglot
- Attardi
- Lebret

# Applications

- NER

| Approach | F1 |
|---|---|
| Ando et al. 2005 | 89.31 |
| Word2Vec | 88.20 |
| GloVe | 88.30 |
| SENNA | **89.51** |

- IMDB Movie Reviews

| Model | Accuracy |
|---|---|
| Wang & Manning | 91.2 |
| Brychcin & Habernal | **92.2** |
| H-PCA | 89.9 |

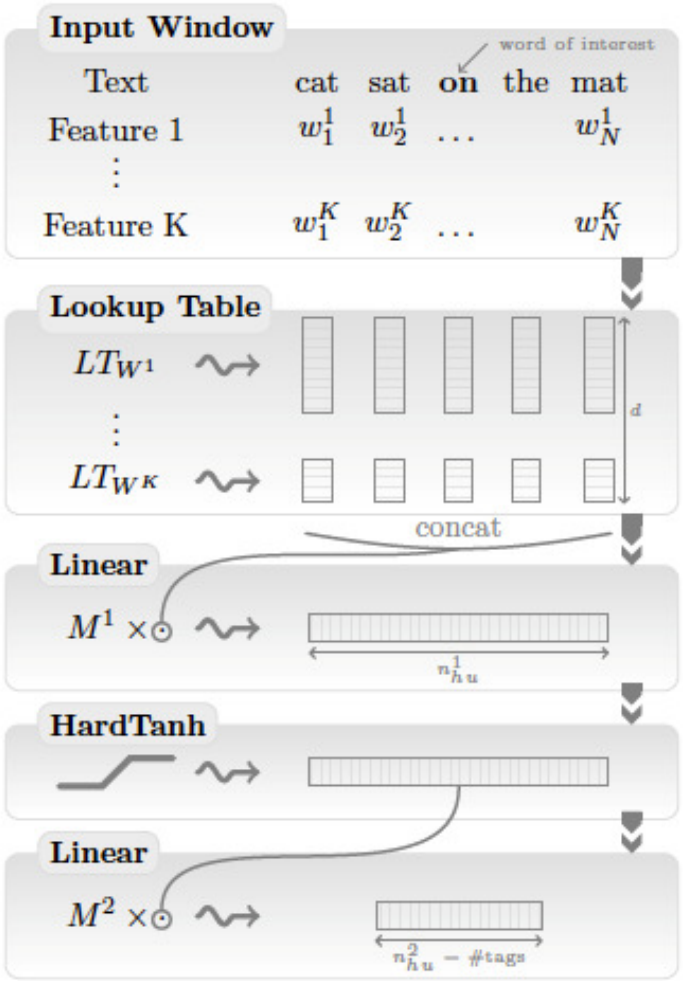
# Visualizing Embeddings

- t-SNE: tool for visualization of high-dimensional dataset

# Deep Learning for NLP

# A Unified Deep Learning Architecture for NLP

- NER (Named Entity Recognition)
- POS tagging
- Chunking
- Parsing
- SRL (Semantic Role Labeling)
- Sentiment Analysis



# Creating the Embeddings

# Obtain Text Corpora

- Wikipedia
  - Get XML dumps from:
    - http://download.wikimedia.org/
  - Get WikiExtractor from:
    - http://medialab.di.unipi.it/wiki/Wikipedia_Extractor
  - Extract the text:
    - WikiExtractor.py -o text itwiki-latest-pages-articles.xml.bz2

# Sentence splitting and tokenization

- NLTK
  - Punkt sentence splitter

```
import nltk.data
splitter = nltk.data.load('tokenizers/punkt/english.pickle')
for line in file:
        for sent in splitter.tokenize(line.strip()):
                print sent
```

  - Tokenizer

```
tokenizer = splitter._lang_vars. word_tokenize
print ' '.join(tokenizer(sent))
```

- Normalize (optional)
  - Convert to lowercase
  - Replace digits with '0'

# Create Embeddings

- word2vec
  - Download and compile:

    > svn checkout http://word2vec.googlecode.com/svn/trunk/ word2vec

    > cd word2vec

    > make
  - Run

    > word2vec word2vec -train train.txt -output vectors.txt
    -cbow 1 -size 50 -window 5 -min-count 40 -negative 0
    -hs 1 -sample 1e-3 -threads 24 -debug 0

# Sequence Taggers

# Train POS Tagger

- Input: tab separated token/tag, one per line

```
Mr.          NNP
Vinken       NNP
is           VBZ
chairman     NN
of           IN
Elsevier     NNP
```

```
> dl-pos.py pos.dnn -t wsj.pos
    --vocab vocab.txt --vectors vectors.txt
    --caps --suffixes suffix.list -w 5 -n 300 –e 10
```

# Test POS Tagger

```
> dl-pos.py pos.dnn < input.tokens
```

# Train NER Tagger

- Input in Conll03 tab separated format:

| | | | |
|---|---|---|---|
| EU | NNP | I-NP | I-ORG |
| rejects | VBZ | I-VP | O |
| German | JJ | I-NP | I-MISC |
| call | NN | I-NP | O |
| to | TO | I-VP | O |
| boycott | VB | I-VP | O |
| British | JJ | I-NP | I-MISC |
| lamb | NN | I-NP | O |

```
> dl-ner.py ner.dnn -t wsj.conll03
        --vocab vocab.txt --vectors vectors.txt
        --caps --suffixes suffix.list -w 5 -n 300
        –e 10 --gazetteer entities.list
```

# Perform NER Tagging

```
> dl-ner.py ner.dnn < input.file
```

# Train Dependency Parser

- Configuration file:

```
Features        FORM -1 0 1
Features        LEMMA -1 0 1 prev(0) leftChild(0) rightChild(0)
Features        POSTAG -2 -1 0 1 2 3 prev(0) leftChild(-1) leftChild(0)
Features        CPOSTAG -1 0 1 rightChild(0) rightChild(rightChild(0))
Features        FEATS -1 0 1
Features        DEPREL leftChild(-1) leftChild(
Feature         CPOSTAG(-1) CPOSTAG(0)
Feature         CPOSTAG(0) CPOSTAG(1)
if(POSTAG(0) = "IN", LEMMA(0)) LEMMA(last(POSTAG, "V"))
```

Linguistic feature: use lemma of last verb if current word is preposition

> desr -c conf -t -m model train.corpus

# Perform Dependency Parsing

```
> desr -m model < input.conll
```

# Evaluate Dependency Parser

> eval09.py –g gold.file –s system.file


Parser Online Demo

# Annotating, Visualizing Dependencies

- DgaAnnotator
- Brat Rapid Annotation Tool

# Code Base

- NLPNET

  https://github.com/attardi/nlpnet

- DeepNL

  https://github.com/attardi/deepnl

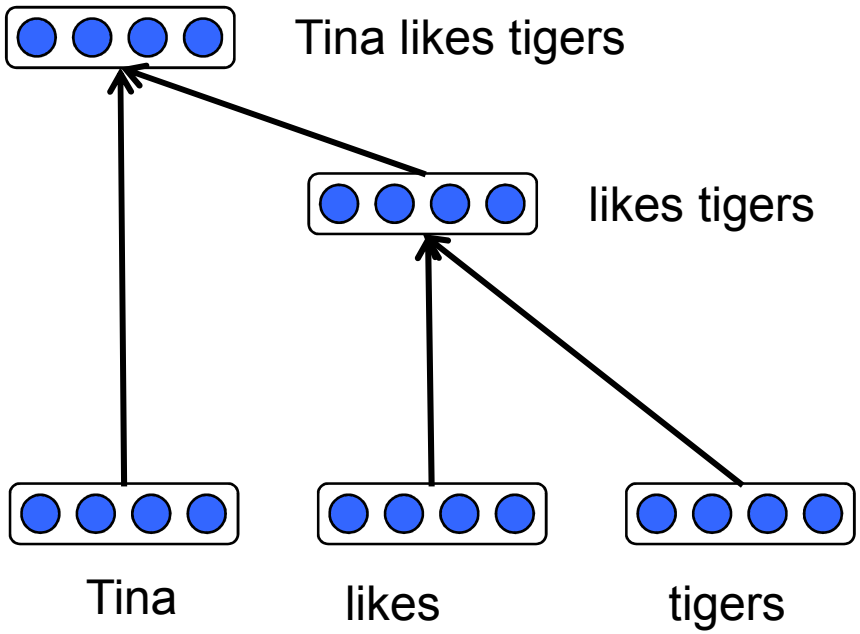
- DeSR parser

  https://sites.google.com/site/desrparser/

# Limits of Word Embeddings

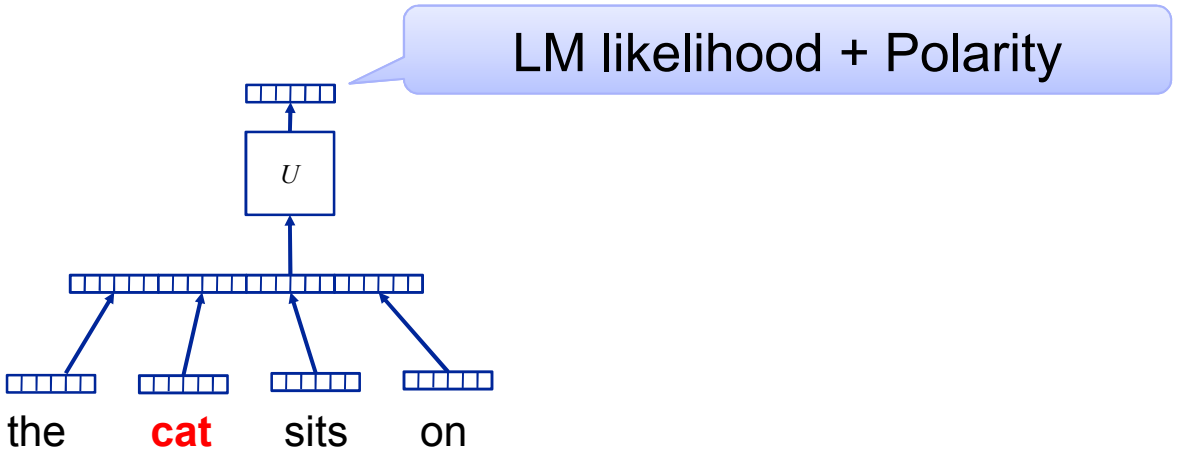
- Limited to words (neither MW nor phrases)
- Represent similarity: antinomies often appear similar
  - Not good for sentiment analysis
  - Not good for polysemous words
- Solution
  - Semantic composition
  - or …

# Recursive Neural Networks



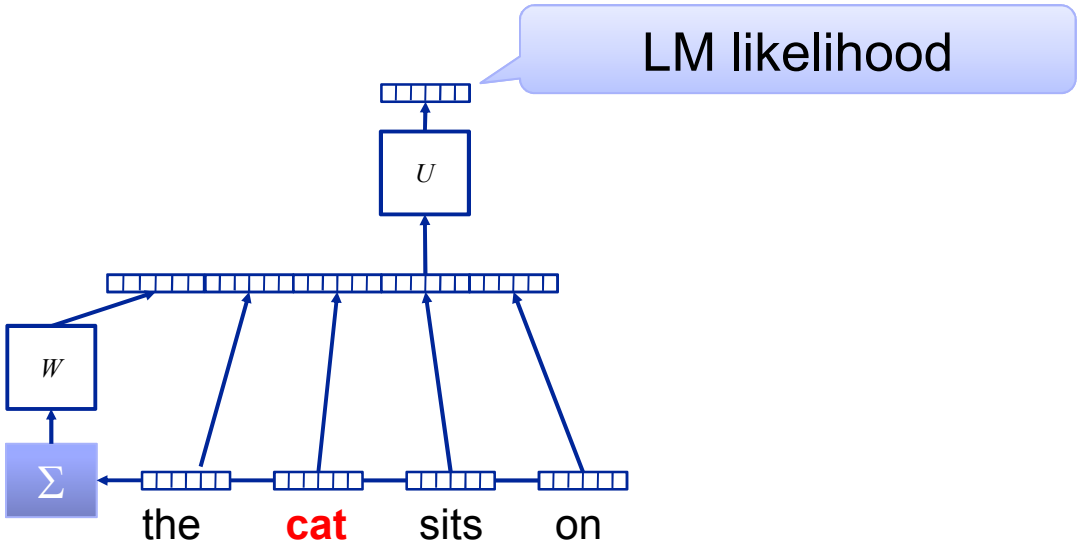
# Sense Specific Word Embeddings

- Sentiment Specific Word Embeddings




- Uses an annotated corpus with polarities (e.g. tweets)
- SS Word Embeddings achieve SOTA accuracy on tweet sentiment classification

# Context Aware Word Embeddings




- Applications:
  - Word sense disambiguation
  - Document Classification
  - Polarity detection
  - Adwords matching

# Multiword Expressions

# Identifying Idiomatic MWE

- Simple idea: idiomatic expressions must be surprising, or else people would not recognize them
- Formally: non-substitutivity, i.e. replacing near synonyms produces weird phrases
- Word embeddings provide near synonyms
- Language model provides weirdness estimate

# Experiment

- Use WE from English Wikipedia
  - Built by WikiExtractor and word2vec
- Use LM on the same corpus:
  - Built with kenlm from moses
- Use WikiMwe for evaluation
- Algorithm:
  - Given phrase, build variants by replacing words with 5 closest words in embeddings space
  - Evaluate variants according to measures:
    - LM, ngram counts

# Examples

LM prob.

OK MWE,art of being,0,-2.9

OK COL,protest against the war,0,-2.0

OK COL,way to Damascus,0,-3.7

OK MWE,androgenic alopecia,0,0.0

KO COL,financial service

n-gram coun

OK MWE,dual gauge,232,-2.1

| | Precision | Recall | F1 |
|---|---|---|---|
| collocation | 53 | 58 | 55 |
| MWE | 51 | 66 | 58 |

# demo

Italian MWE Identification

# References


1. R. Collobert et al. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12, 2461–2505.
2. Q. Le and T. Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proc. of the 31st International Conference on Machine Learning*, Beijing, China, 2014. JMLR:W&CP volume 32.
3. Rémi Lebret and Ronan Collobert. 2013. Word Embeddings through Hellinger PCA. *Proc. of EACL 2013*.
4. O. Levy and Y. Goldberg. 2014. Neural Word Embeddings as Implicit Matrix Factorization. In *Advances in Neural Information Processing Systems* (NIPS).
5. T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of Workshop at ICLR*, 2013.
6. Tang et al. 2014. Learning Sentiment-SpecificWord Embedding for Twitter Sentiment Classification. In *Proc. of the 52nd Annual Meeting of the ACL*, 1555–1565,