Machine Learning for Language Modelling

Practical

Marek Rei

Advanced LM approaches

- Use log-space for probabilities
- Discriminative models optimised for a specific task
- Pruning
  - Only store n-grams where count > threshold
  - Entropy-based pruning
- Efficient data structures
  - Bloom filters
  - Store words as indexes not strings
  - Quantise probabilities (~8 bits)

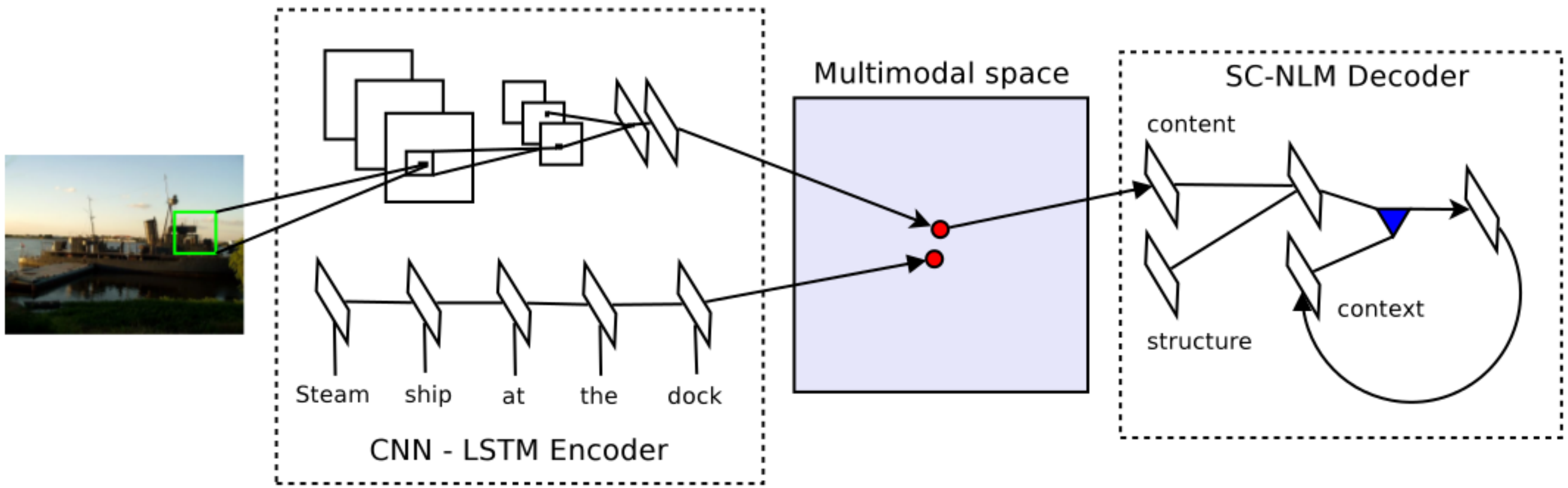
Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models

Ryan Kiros, Ruslan Salakhutdinov, Richard S. Zemel

University of Toronto

2014

System description



- First model learns to map images and text into the same vector space
- A neural language model learns to generate text descriptions based on a vector from that vector space

there is a cat sitting on a shelf .

a plate with a fork and a piece of cake .

a black and white photo of a window .

a young boy standing on a parking lot next to cars .

a wooden table and chairs arranged in a room .

a kitchen with stainless steel appliances .

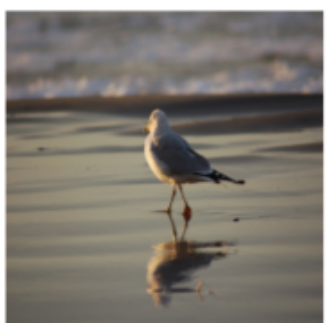
this is a herd of cattle out in the field .

a car is parked in the middle of nowhere .

a ferry boat on a marina with a group of people .

a little boy with a bunch of friends on the street .

a giraffe is standing next to a fence in a field .
(hallucination)



the two birds are trying to be seen in the water .
(counting)

a parked car while driving down the road .
(contradiction)

the handlebars are trying to ride a bike rack .
(nonsensical)

a woman and a bottle of wine in a garden .
(gender)

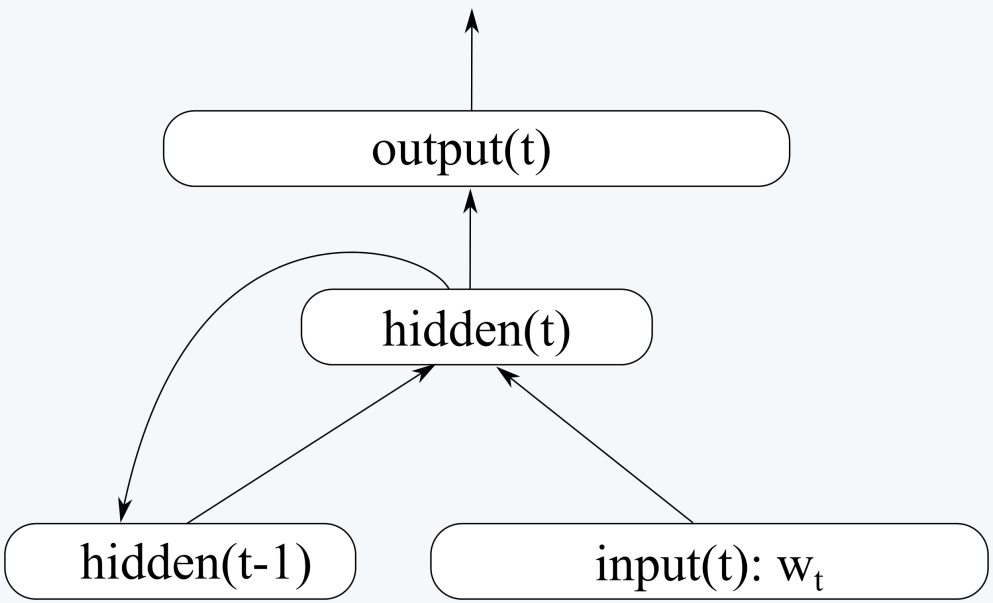
Recurrent neural network based language model

Tomas Mikolov, Martin Karafiat, Lukas Burget,
Jan “Honza” Cernocky, Sanjeev Khudanpur

Bnro University of Technology

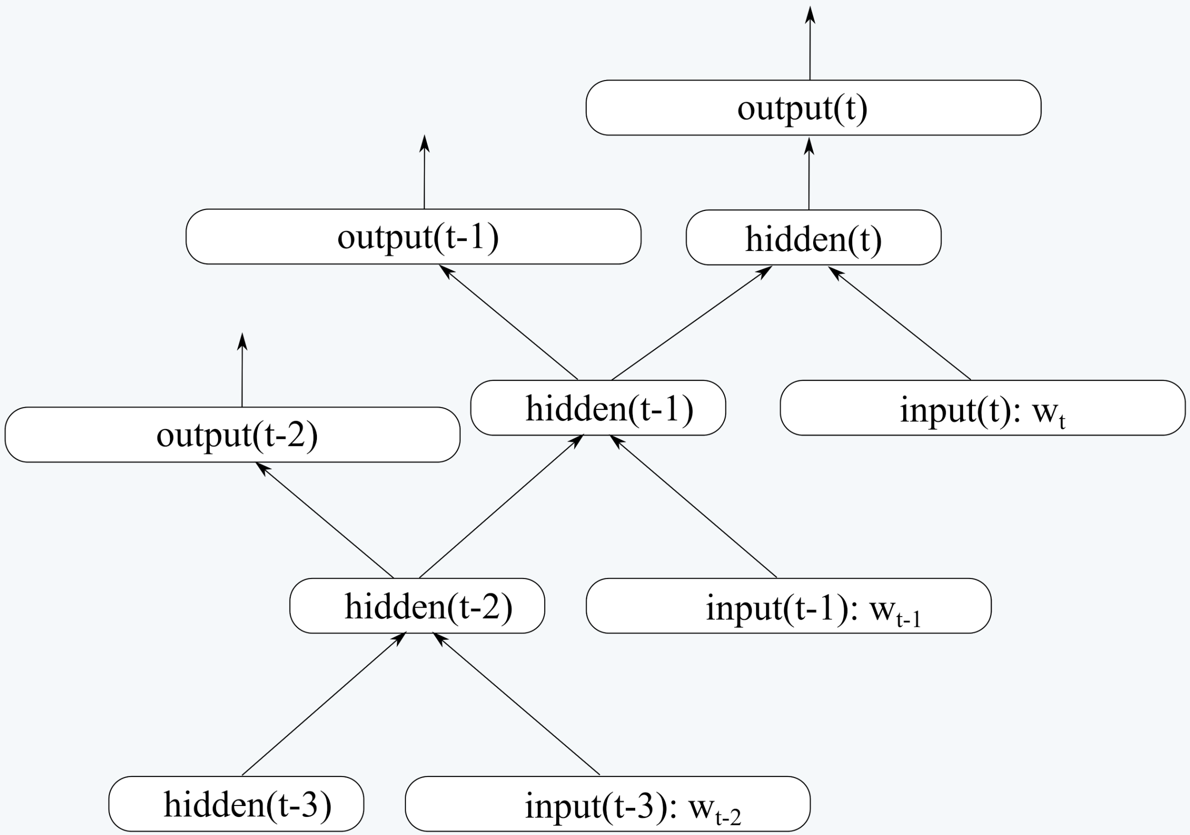
2010

RNNLM



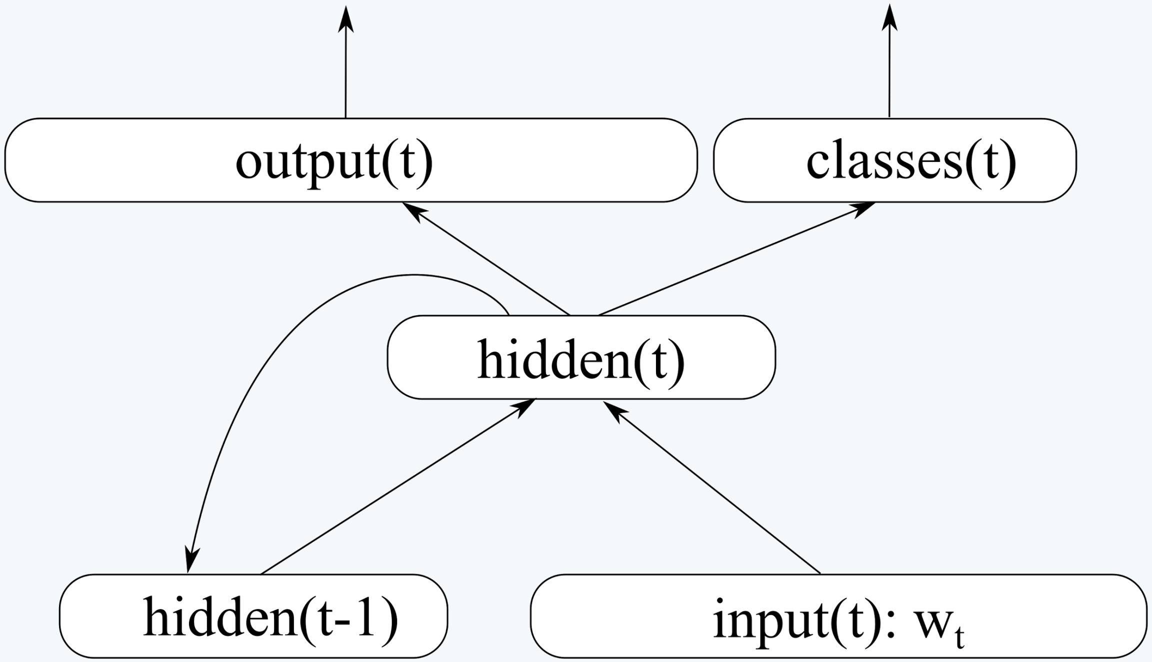
- One input word is added at every time step
- The hidden vector from the previous time step is used as input for the next time step

Backpropagation through time



To train the RNNLM, we “unfold” it over previous time steps

Class-based output



- Most of the computation is done in the output layer (V*H)
- Can break it down into two separate steps:

$$P(\text{word} \mid \text{context}) = P(\text{word}|\text{class, context}) * P(\text{class}|\text{context})$$

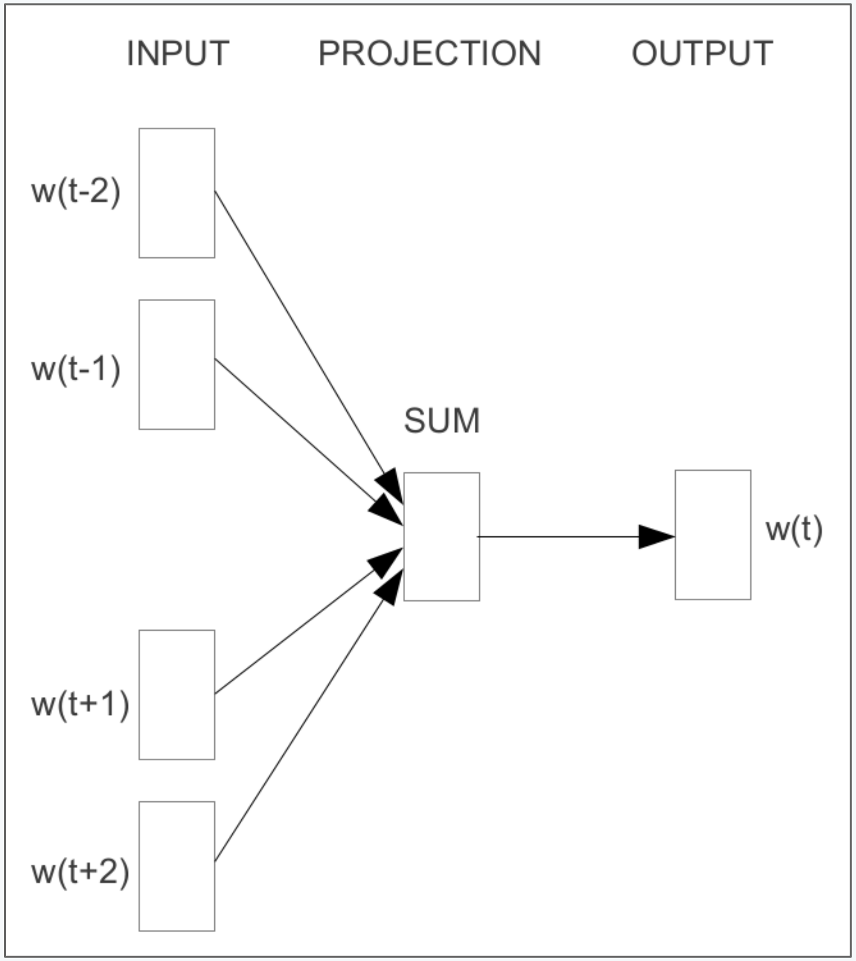
Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean

Google

2013

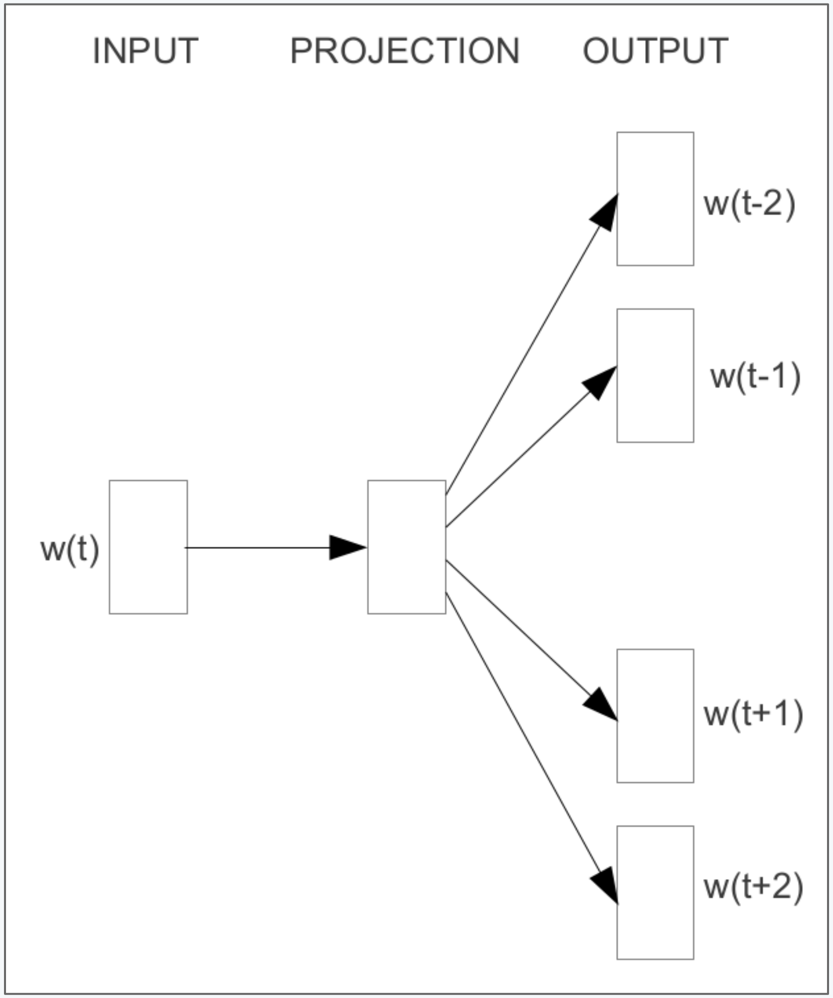
Word2vec: CBOW



CBOW

Predict the current word based on the surrounding words

Word2vec: skip-gram



Skip-gram

Predict the surrounding words based on the current word

Linguistic regularities

France - Paris + Italy = ?

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Extra materials

Don’t count, predict!
A systematic comparison of context-counting vs. context-predicting semantic vectors

Marco Baroni, Georgiana Dinu, German Kruszewski

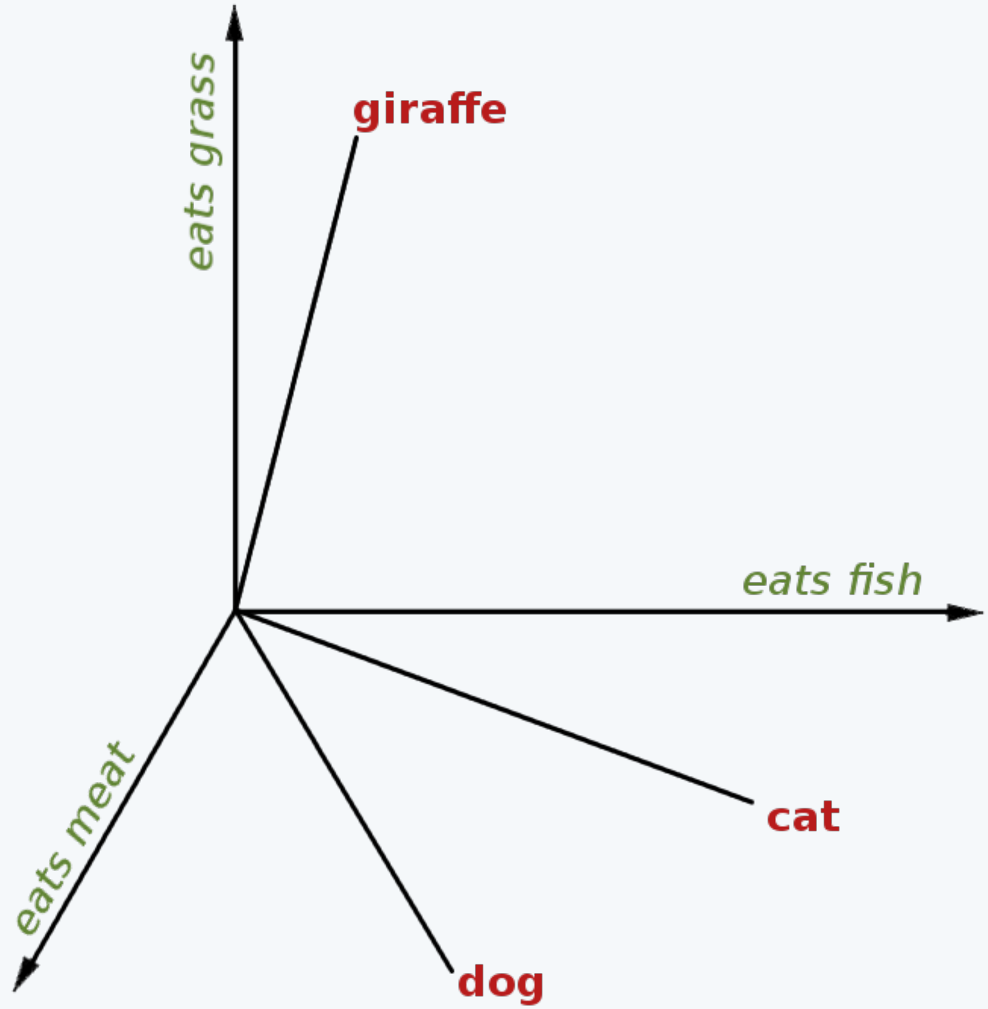
University of Trento

Motivation

We can represent words as vectors

Words with similar meaning have similar vectors

What is the best way to construct these vectors?



Distributional hypothesis

Words which are similar in meaning occur in similar contexts

(Rubenstein & Goodenough, 1965).

I was reading a **magazine** today	I was reading a **newspaper** today
The **magazine** published an article	The **newspaper** published an article
He buys this **magazine** every day	He buys this **newspaper** every day

The counting model

One way of creating a vector for a word:

Let's count how often it occurs together with specific other words

"He buys this **newspaper** every day"
"I read a **newspaper** every day"

buys	this	every	day	read	a
1	1	2	2	1	1

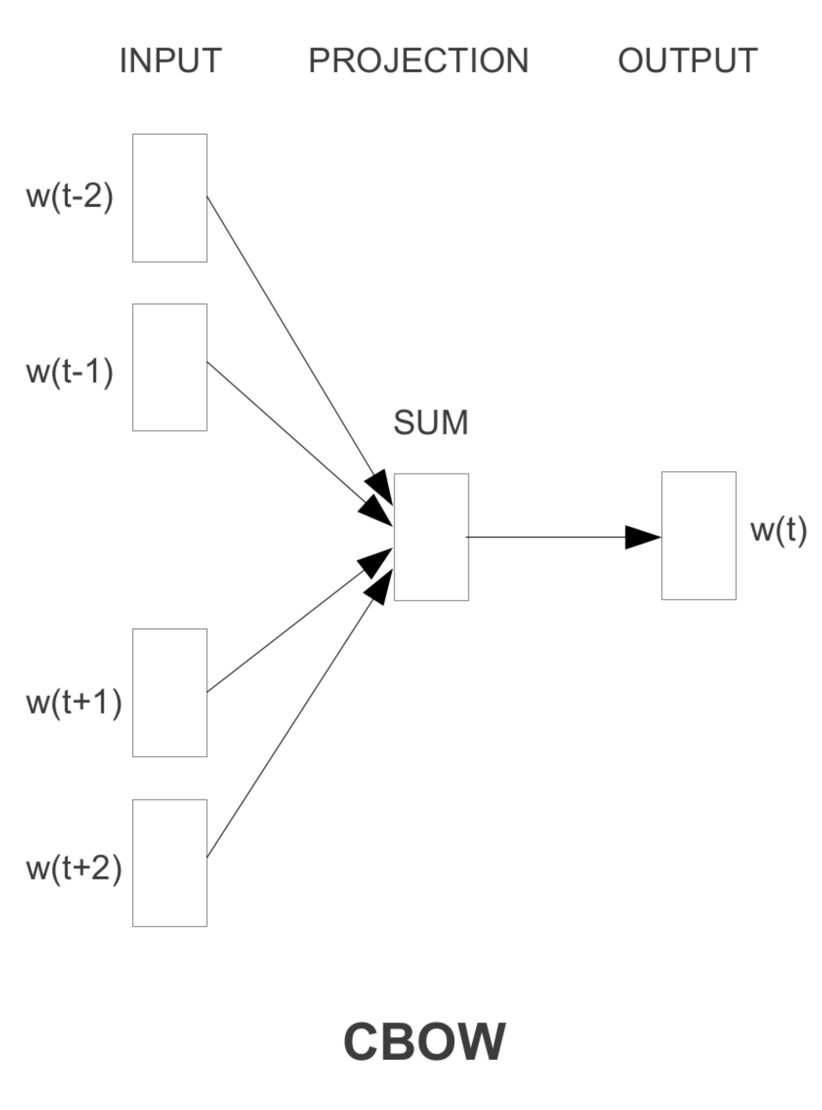
The predicting model

Second option:

Learn vectors with a neural network

Train it to predict the target word, given the context

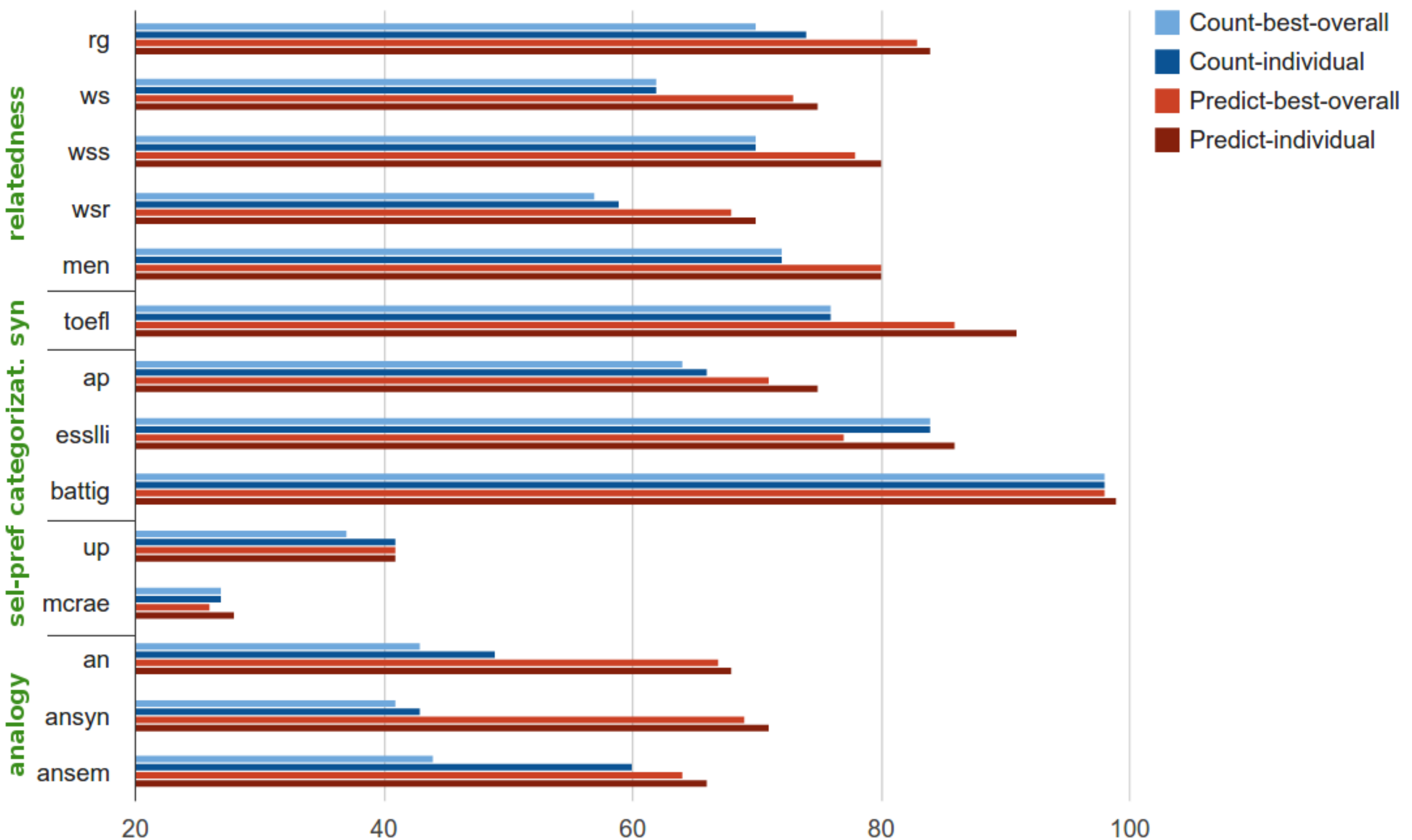
Implemented in the word2vec toolkit
https://code.google.com/p/word2vec/



Evaluation

1. **Semantic relatedness**
   a. **rg**: 65 noun pairs
   b. **ws**: Wordsim353, 353 word pairs
   c. **wss**: Subset of Wordsim353 focused on similarity
   d. **wsr**: Subset of Wordsim353 focused on relatedness
   e. **men**: 1000 word pairs
2. **Synonym detection**
   a. **toefl**: 80 multiple-choice questions with 4 synonym candidates
3. **Concept categorization**
   a. **ap**: 402 concepts in 21 categories
   b. **esslli**: 44 concepts in 6 categories
   c. **battig**: 83 concepts in 10 categories
4. **Selectional preferences**
   a. **up**: 221 word pairs
   b. **mcrae**: 100 noun-verb pairs
5. **Analogy recovery**
   a. **an**: ~19,500 analogy questions
   b. **ansyn**: Subset of the analogy questions, focused on syntactic analogies
   c. **ansem**: Subset of the analogy questions, focused on semantic analogies

Results



Count-best-overall
Count-individual
Predict-best-overall
Predict-individual
relatedness
syn
categorizat.
sel-pref
analogy
rg
ws
wss
wsr
men
toefl
ap
esslli
battig
up
mcrae
an
ansyn
ansem
20
40
60
80
100

Best configurations

The best parameter choices for counting models:

- window size 2 (bigger is not always better)
- weighted with PMI, not LMI
- no dimensionality reduction (not using SVD or NNMF)

The best parameters for the neural network model:

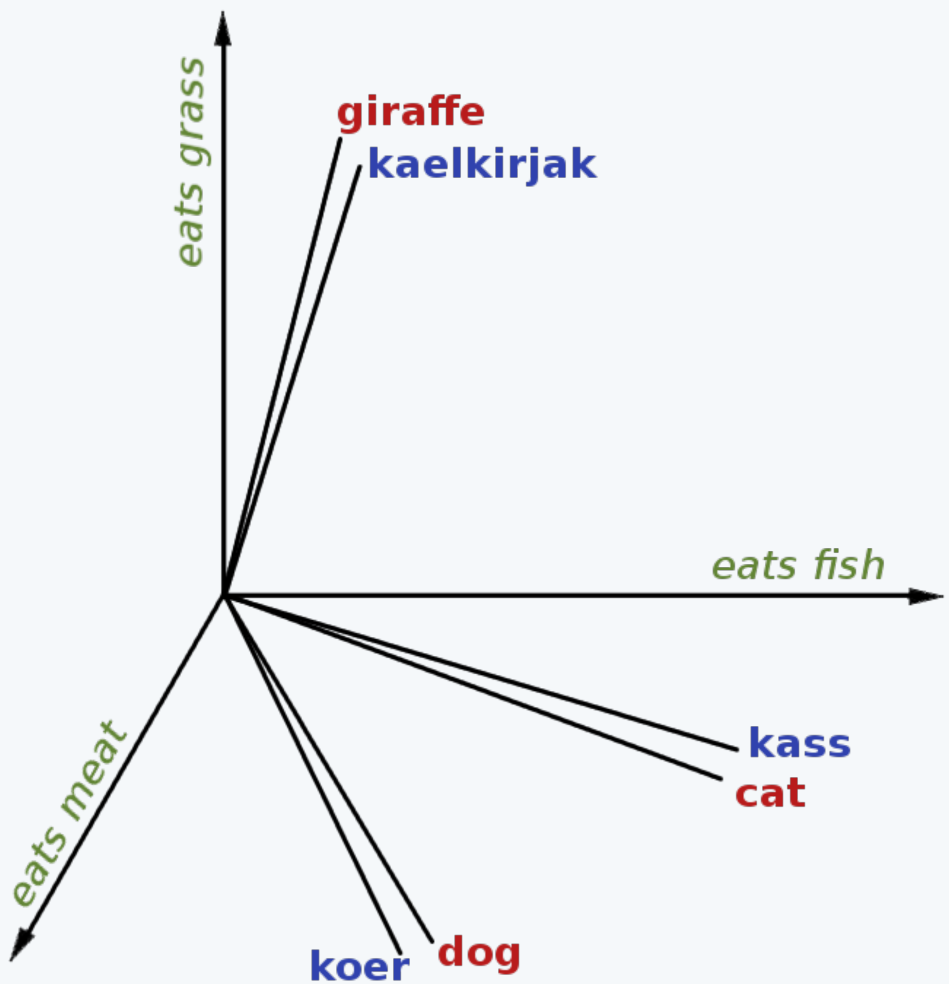
- window size 5
- negative sampling (not hierarchical softmax)
- subsampling of frequent words
- dimensionality 400

Multilingual Models for Compositional Distributed Semantics

Karl Moritz Hermann, Phil Blunsom
University of Oxford

Motivation

Motivation



eats grass
giraffe
kaelkirjak
eats fish
kass
cat
eats meat
koer
dog

The Idea

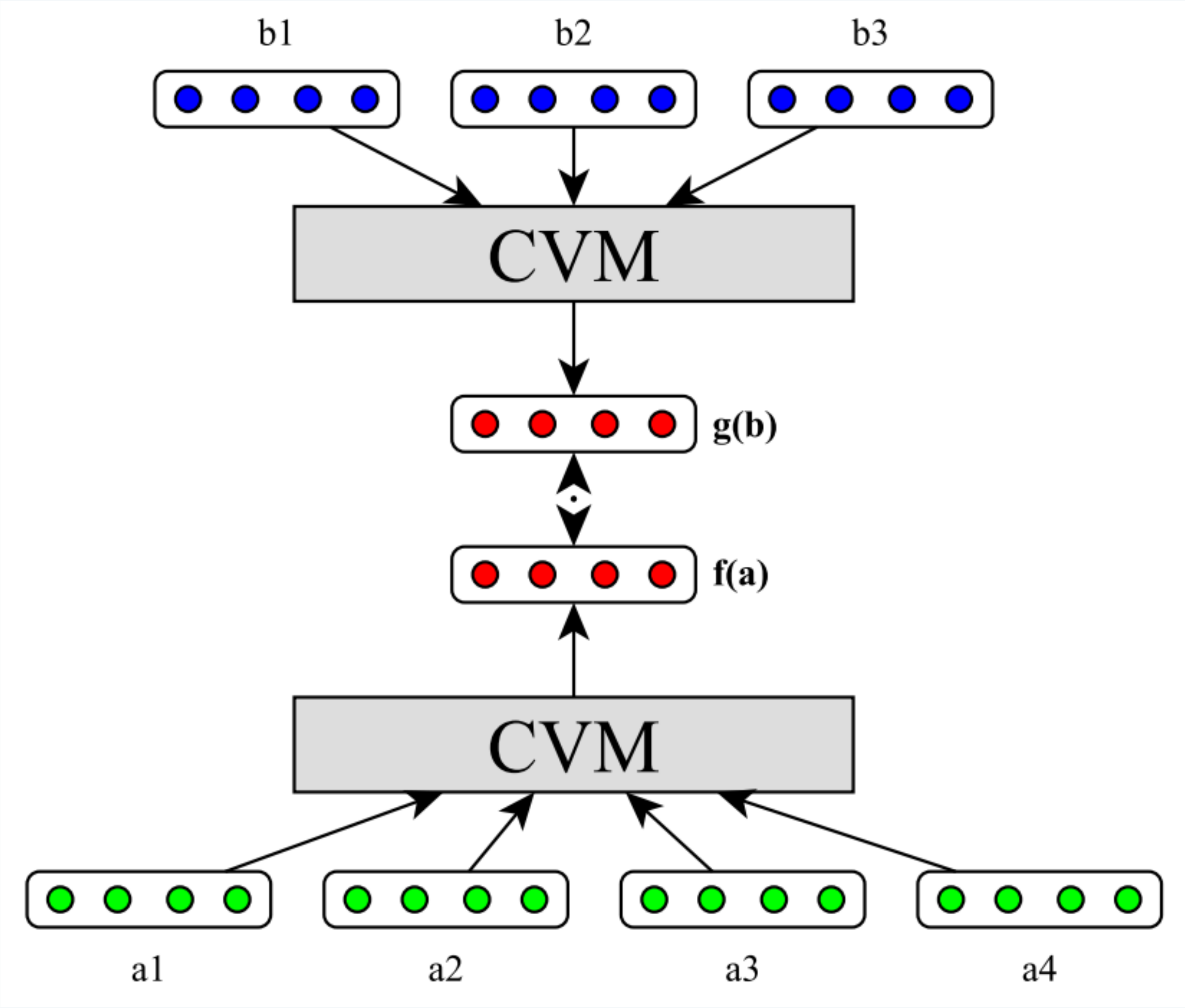
We have **sentence *a* in one language**, and function *f(a)* which maps that sentence into a vector representation.

We then have **sentence *b***, the same sentence **in a different language**, and function *g(b)* for mapping that into a vector representation.

Goal: **have *f(a)* and *g(b)* be identical**, because both of these sentences have the same meaning.

Training: **Process a series of parallel sentences *a* and *b***, and each time we **adjust the functions *f(a)* and *g(b)*** so that they would produce more similar vectors.

The Multilingual Model



Composition

1. The **additive model** (ADD)

$$f_{ADD}(a) = \sum_{i=1}^{n} a_i$$

2. The **bigram model** (BI)

$$f_{BI}(a) = \sum_{i=1}^{n} tanh(a_{i-1} + a_i)$$

Optimization

The **error function** we try to optimize during training:

$$E(a,b) = ||f(a) - g(b)||^2$$

$$E_{nc}(a,b,c) = [m + E(a,b) - E(a,c)]_+$$

Evaluation

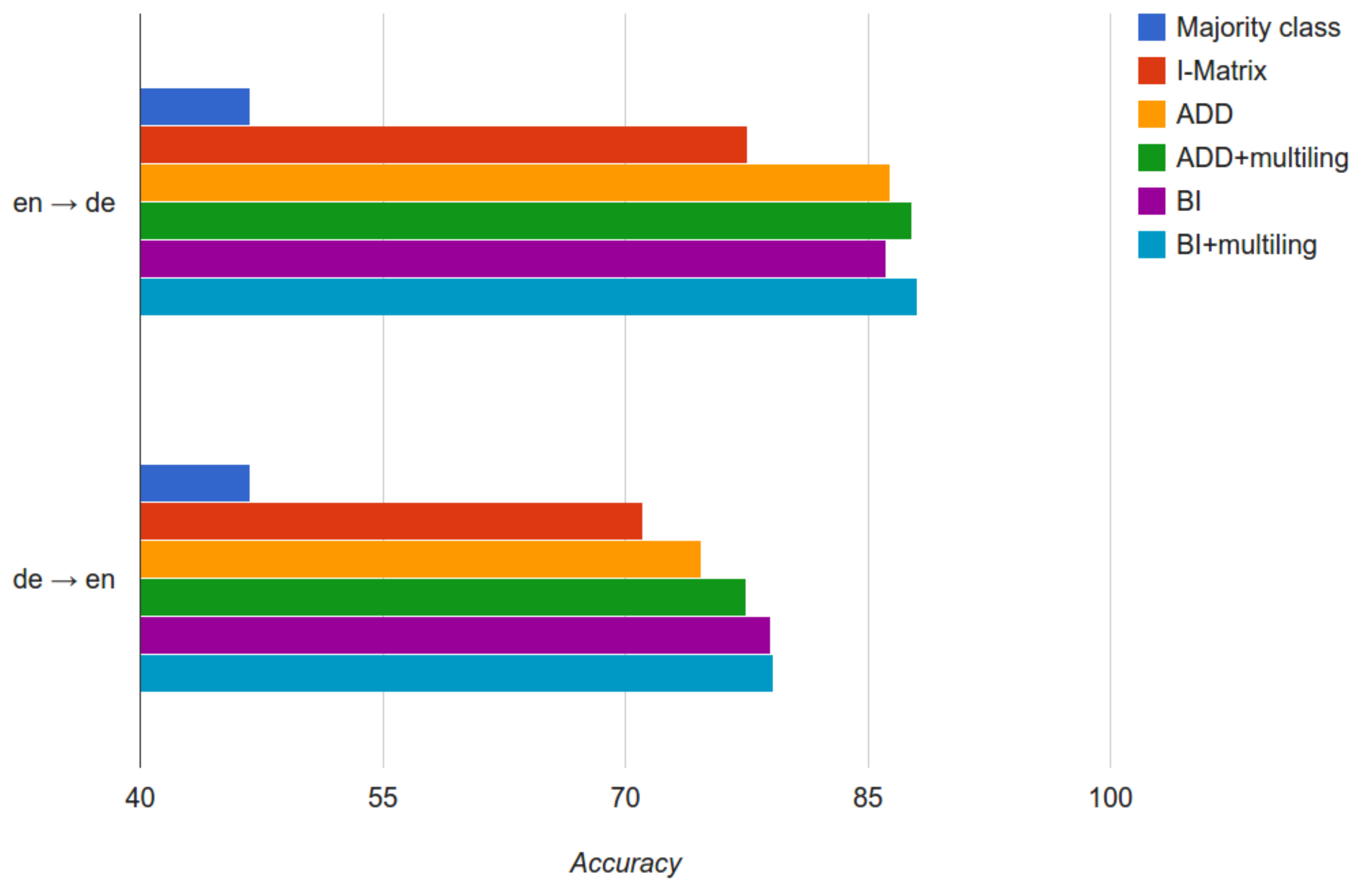
The system is evaluated on the **task of topic classification**.

The classifier is **trained on one language** (eg English) and then **tested on another language** (eg German) without training data.

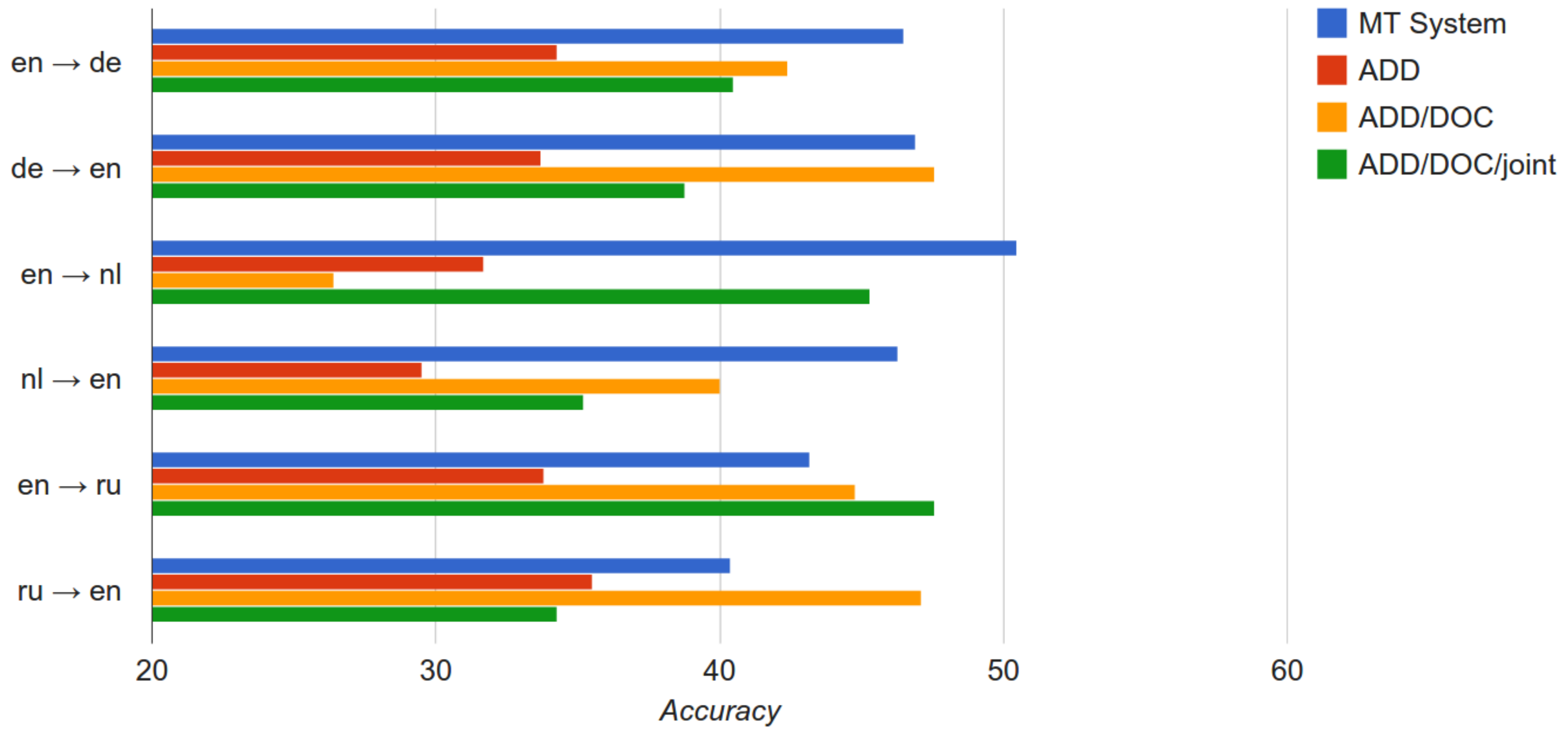
Two (main) datasets:

1. The **cross-lingual document classification (CLDC)** task. Trained on the parallel Europarl corpus, and tested on Reuters RCV1/RCV2. English-German and English-French.
2. A new corpus from parallel **subtitles of TED talks**. Each talk also has topic tags assigned to them, and the task is to assign a correct tag to every talk, using the document-level vector.

Results (CLDC)

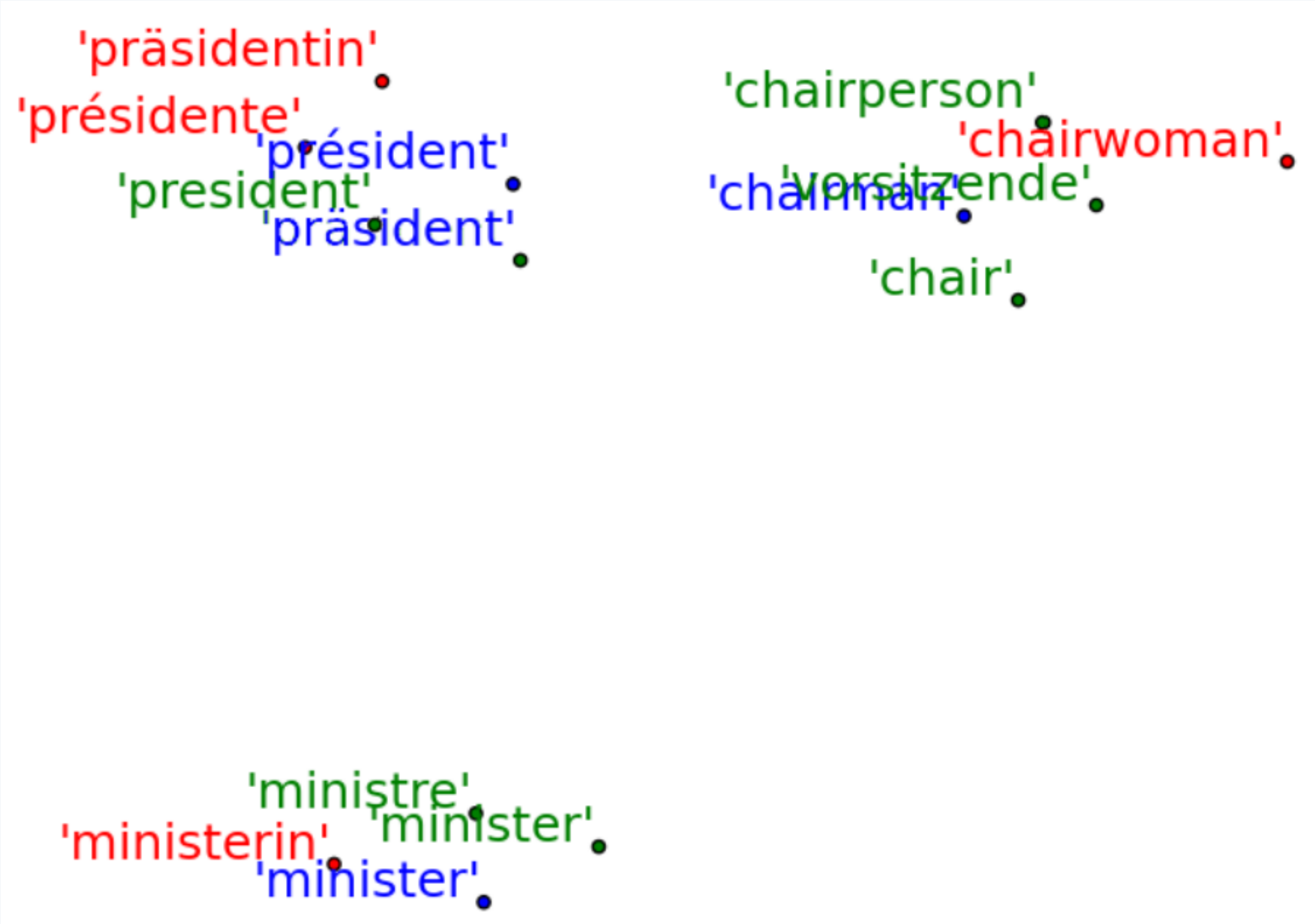


Results (TED)



en → de
de → en
en → nl
nl → en
en → ru
ru → en
20
30
40
50
60
Accuracy
MT System
ADD
ADD/DOC
ADD/DOC/joint

Example



'präsidentin'
'présidente'
'président'
'president'
'präsident'
'chairperson'
'chairwoman'
'vorsitzende'
'chairman'
'chair'
'ministre'
'minister'
'ministerin'
'minister'