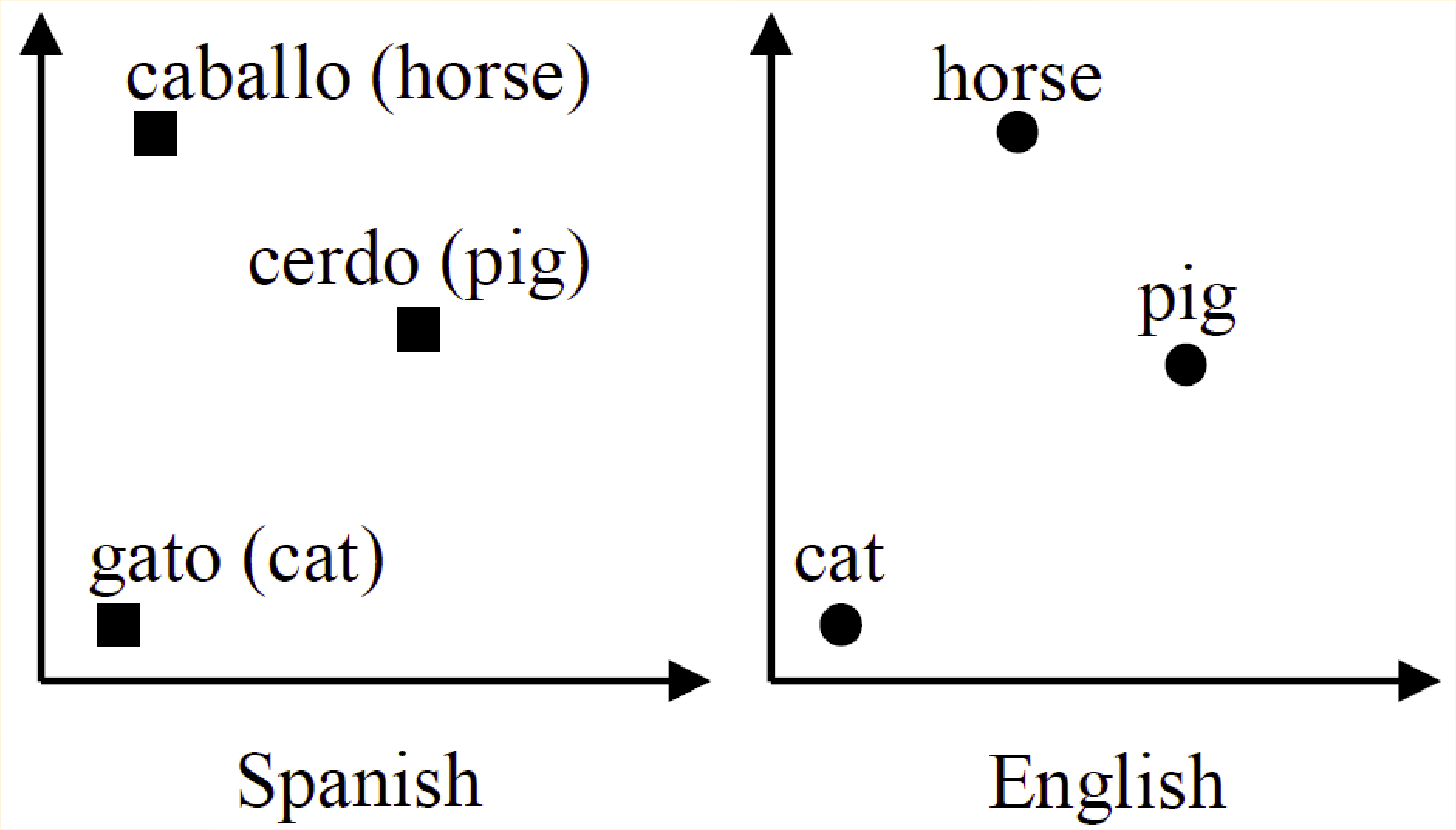
# Cross-lingual Word Embedding




Speaker: TIAN-Junfeng ( `@rgtjf` )

# References

[1] arXiv13-Mikolov et al.-**Exploiting Similarities among Languages for Machine Translation**
[2] ICLR17-Smith et al.-**Offline Bilingual Word Vectors, Orthogonal Transformations and the Inverted Softmax**
[3] ACL17-Meng Zhang et al.-**Adversarial Training for Unsupervis-ed Bilingual Lexicon Induction**

# Recap

## How to measure the `similarity` between words?

e.g. `Dog` V.S. `Cat`

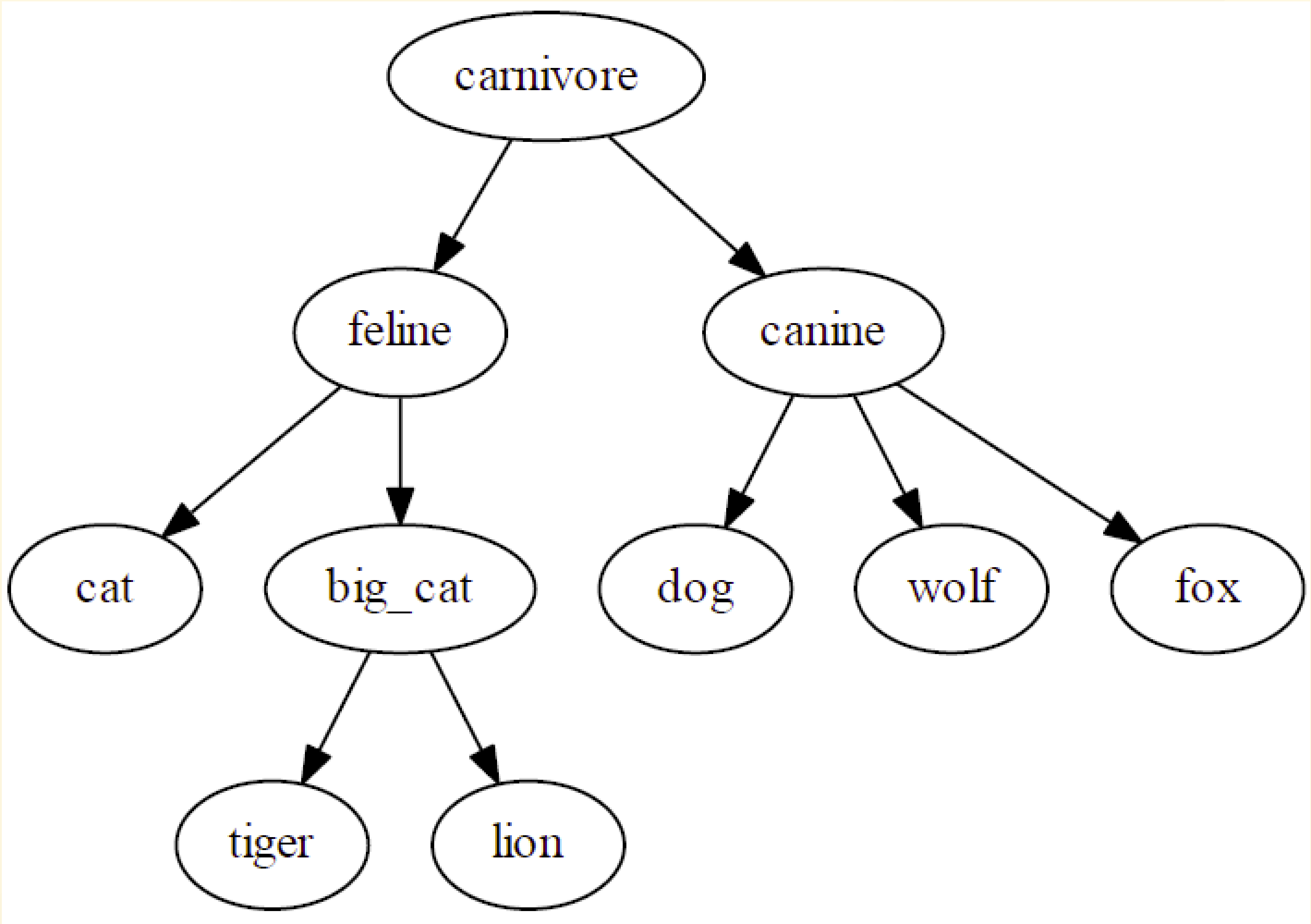
# Recap

**How to measure the `similarity` between words?**

e.g. `Dog` V.S. `Cat`

# Solution I

- **WordNet**




- carnivore [ˈkɑ:nɪvɔ:(r)] (肉食动物)
- feline [ˈfi:laɪn] (猫科动物), canine [ˈkeɪnaɪn] (.)

# Recap

**How to measure the `similarity` between words?**

e.g. `Dog` V.S. `Cat`

```
the cat is walking in the bedroom.
the dog is running in the kitchen.
```

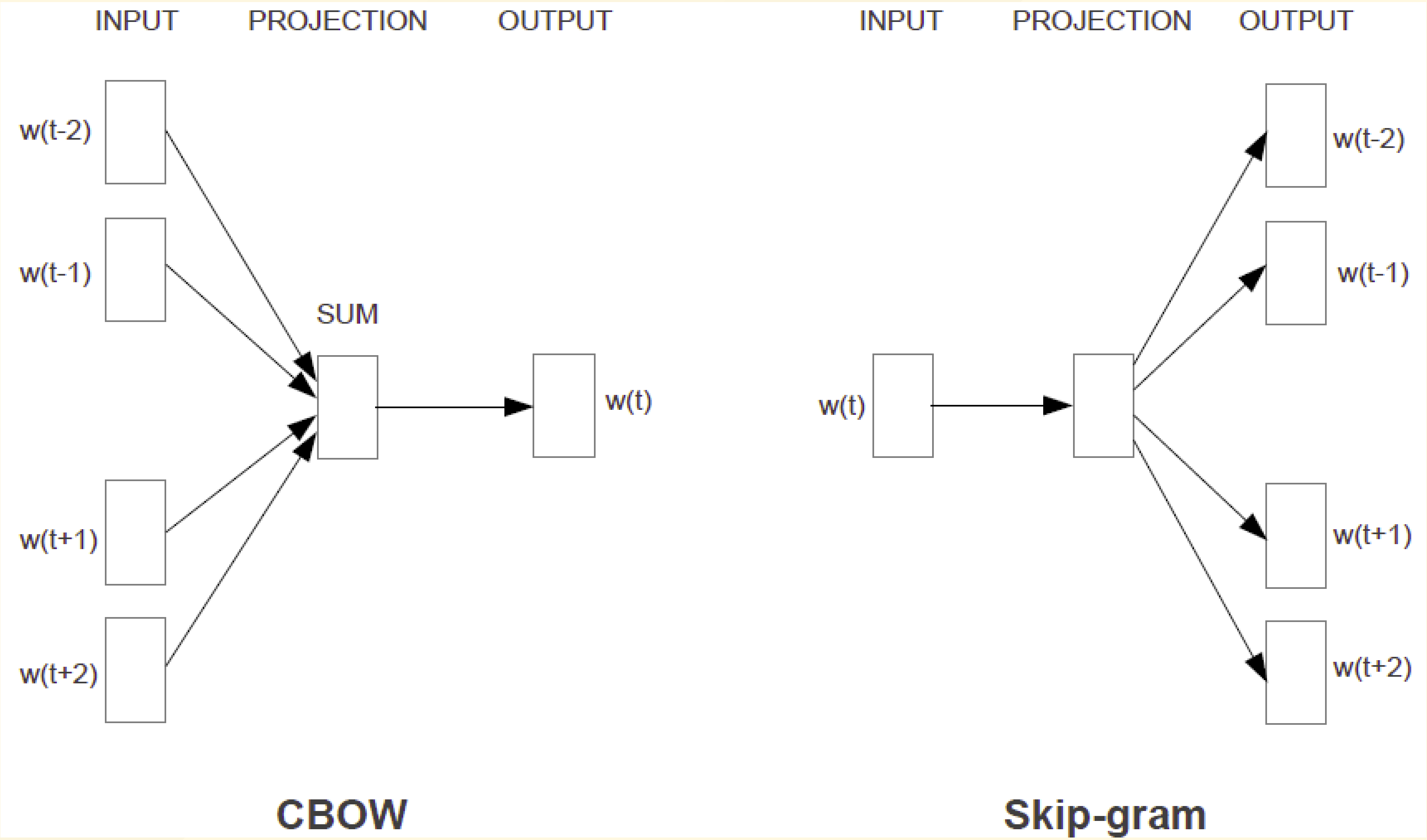
# Solution II

```
the cat is walking in the bedroom.
the dog is running in the kitchen.
```

- **Context Information (Co-occurrence Matrix)**
  - word-document (Topic Models)
  - word-word (word2vec)
- However,
  - sparsity
  - curse of dimensionality

## Don't count, predict!

CBoW, Skip-gram, etc.

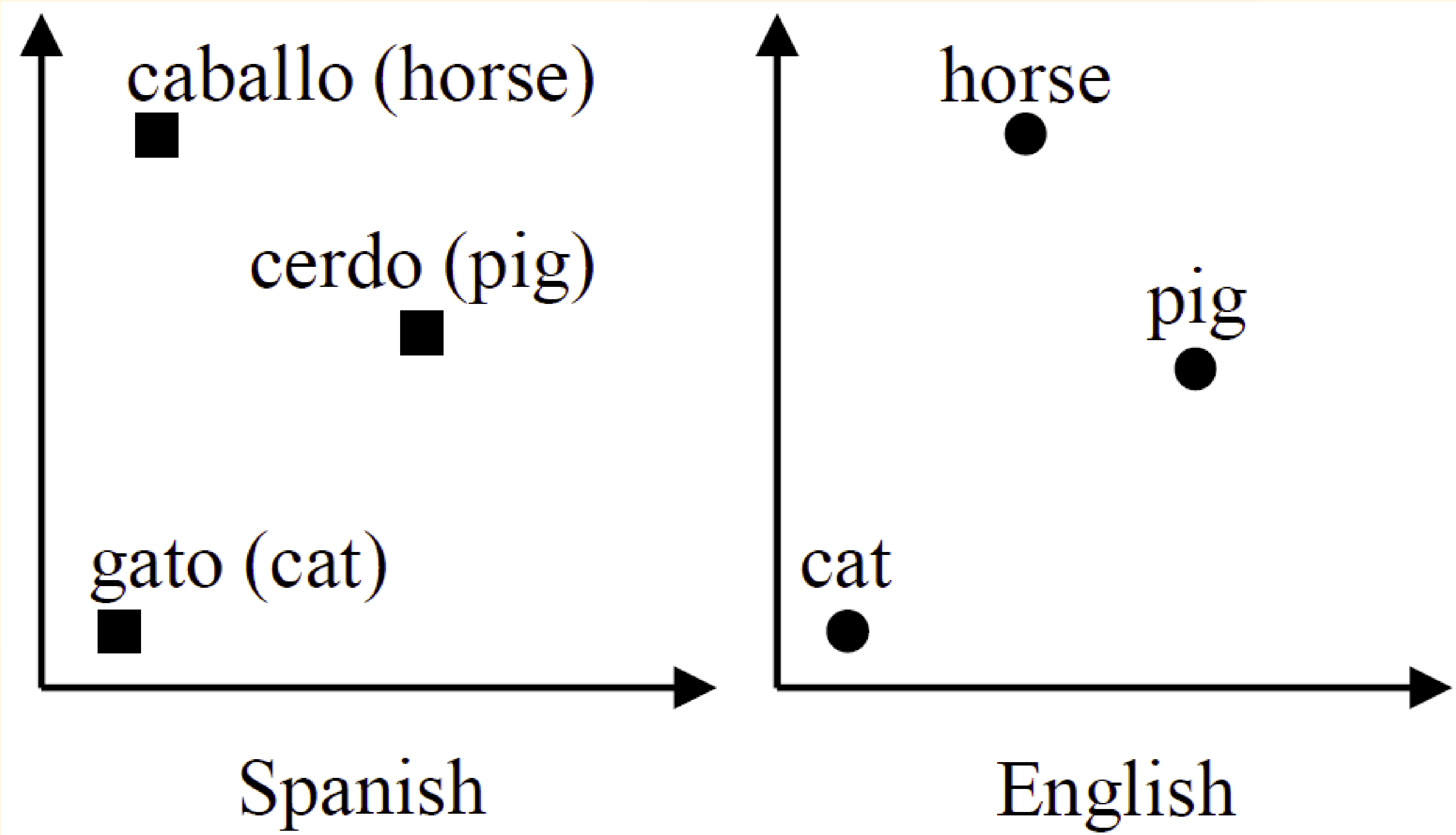


# Cross-lingual Word Embeddings

**How to measure the `similarity` between words from `different languages`?**

## Solution I

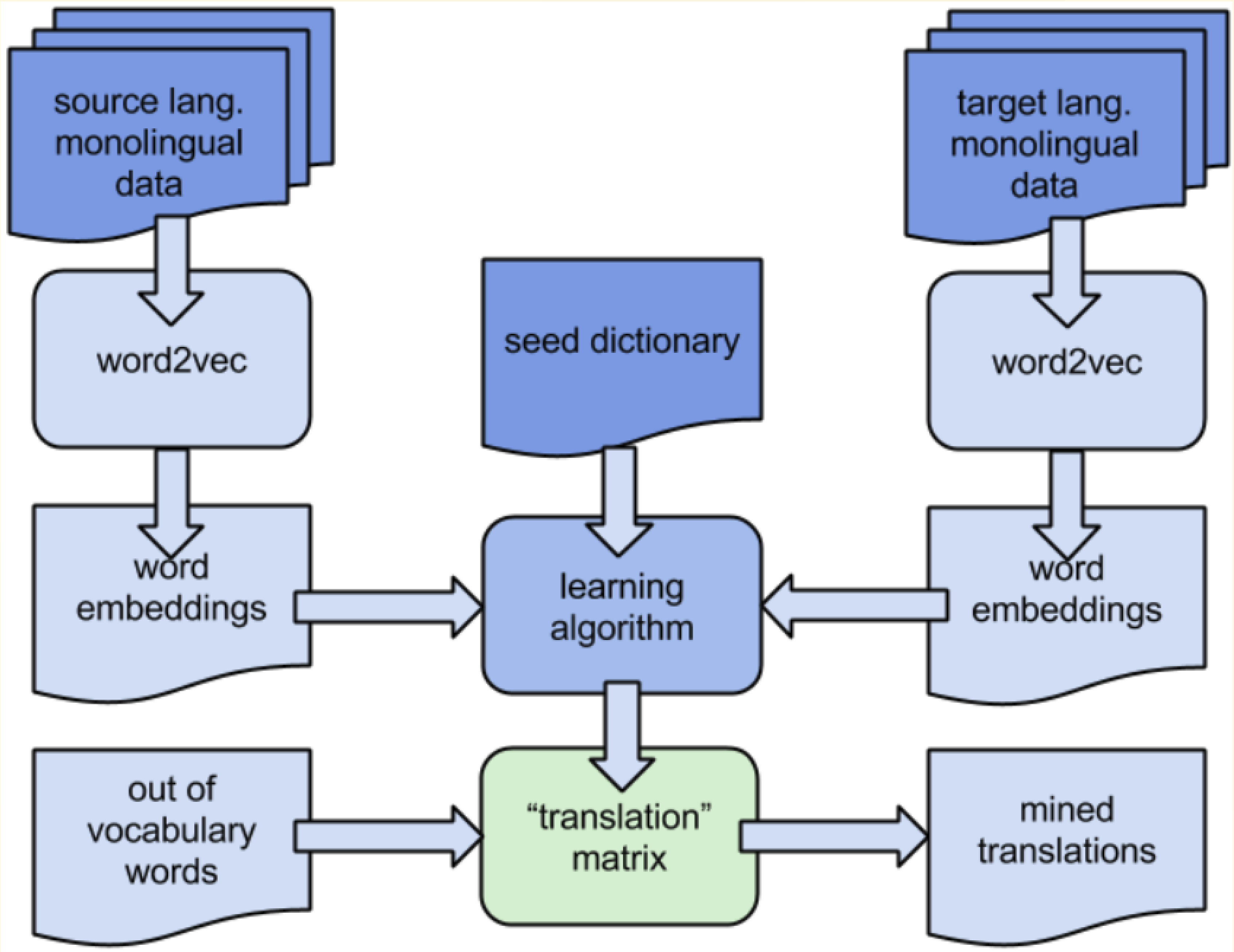
- **Bi-lingual Lexicon** or **Machine Translation**
  - expensive, time-consuming, and need experts
  - require large parallel data to build translation model




- **approximate** `isomorphism` [ˌaɪsəʊ'mɔ:fɪzəm] 同形
- **it is possible to learn an accurate** `linear mapping` **from one space to another**

# How it works

- **Acquire**:
  - monolingual data (hundreds of millions of words)
  - seed dictionary (hundreds to thousands of words)
- **Learn**:
  - translation matrix (SGD or other learning algorithm)
  - distributed representations of words (CBoW or Skip-gram)


source lang. monolingual data
target lang. monolingual data
word2vec
seed dictionary
word2vec
word embeddings
learning algorithm
word embeddings
out of vocabulary words
“translation” matrix
mined translations

# Translation Matrix

Let $x$, $y$ be distributed word representations for two words in a translation pair.

Then we want to learn a matrix $W$ such that:

$$y = Wx$$

This gives an optimization problem over translation pairs in the seed dictionary:

$$\arg\min_{W} \sum ||Wx_i - y_i||^2$$

# Results

Dataset

- monolingual data

| Language | Training tokens | Vocabulary size |
|---|---|---|
| English | 575M | 127K |
| Spanish | 84M | 107K |
| Czech | 155M | 505K |

- seed dictionary (Google Translator)
  most frequent 5K words / 1k test
- Cz: Czech [tʃek] 捷克语

# Results

Baselines

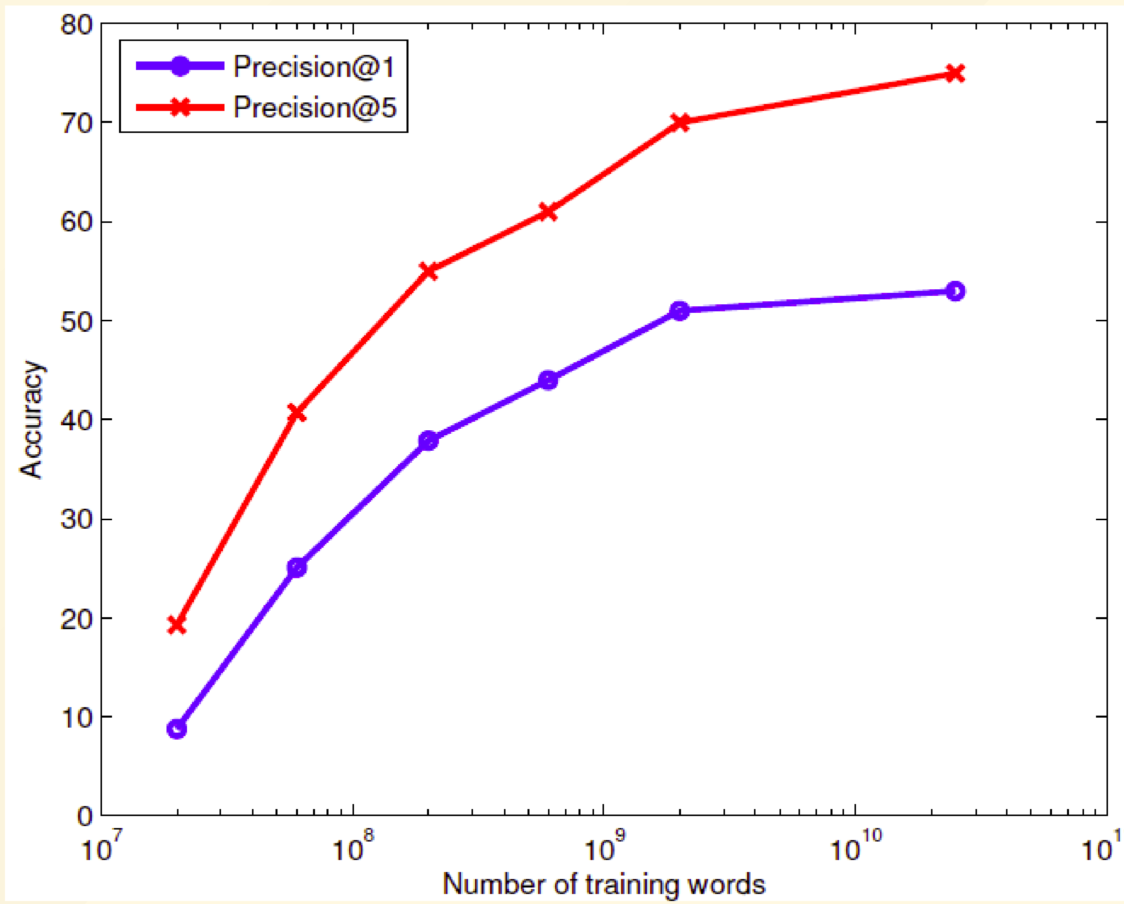
- Edit Distance (morphological structure)
- Word Co-occurrence
  [1]. form count-based word vectors (dictionary size)
  [2]. count occurrrence of in-dictionary words with a short window (up to 10 words) for each test word in source language / target language
  [3]. using the dictionary, map the word count vectors from source to target language
  [4]. for each test word, search for the most similar vector in the targe language

# Results

| Translation | Edit Distance | | Word Co-occurrence | | Translation Matrix | | ED + TM | | Coverage |
|---|---|---|---|---|---|---|---|---|---|
| | P@1 | P@5 | P@1 | P@5 | P@1 | P@5 | P@1 | P@5 | |
| En → Sp | 13% | 24% | 19% | 30% | 33% | 51% | 43% | 60% | 92.9% |
| Sp → En | 18% | 27% | 20% | 30% | 35% | 52% | 44% | 62% | 92.9% |
| En → Cz | 5% | 9% | 9% | 17% | 27% | 47% | 29% | 50% | 90.5% |
| Cz → En | 7% | 11% | 11% | 20% | 23% | 42% | 25% | 45% | 90.5% |

- related spellings languages (English and Spanish)
- distant language pairs (English and Czech)
- the word vectors trained on source language should be several times (`around 2x-4x`) larger than that on the target language.

# With more data...




Performance doubles if the amount of data increases by two orders of magnitude.

| Spanish word | Computed English Translations | Dictionary Entry |
|---|---|---|
| emociones | emotions<br>emotion<br>feelings | emotions |
| protegida | wetland<br>undevelopable<br>protected | protected |
| imperio | dictatorship<br>imperialism<br>tyranny | empire |
| determinante | crucial<br>key<br>important | determinant |
| preparada | prepared<br>ready<br>prepare | prepared |
| millas | kilometers<br>kilometres<br>miles | miles |
| hablamos | talking<br>talked<br>talk | talk |
| destacaron | highlighted<br>emphasized<br>emphasised | highlighted |

# More Findings

## Orthogonal Transformations

- define similarity matrix $S = YWX^T$

$$S_{ij} = y_i^T W x_j = y_i \cdot (W x_j)$$

- define a second similarity matrix $S' = XQY^T$

$$S'_{ji} = x_j^T Q y_i = x_j \cdot (Q y_i)$$

$$S' = S^T, S^T = XW^TY^T \Rightarrow Q = W^T$$

- $x \backsim W^T y, y \backsim W x \Rightarrow x \backsim W^T W x$

# How it works

under orthogonality ($W^TW = I$)

$\arg\min_W \sum_i ||y_{i*} - Wx_{i*}||$

$= \arg\min_W \sum_i (|y_{i*}|^2 + |x_{i*}|^2 - 2y_{i*}^T W x_{i*})$

$= \arg\max_W \sum_i y_{i*}^T W x_{i*}$

$= \arg\max_W Tr(XWY^T)$

$= \arg\max_W Tr(Y^T XW)(Tr(AB) = Tr(BA))$

Perform SVD on $Y^T X = U\Sigma V^T$

$= \arg\max_W Tr(U\Sigma V^T W)$

$= \arg\max_W Tr(\Sigma V^T W U)$

$\Rightarrow V^T W U = I$

$\Rightarrow W = VU^T$

# TLDR, just tell me what to do!

```
def learn_transformation(source_matrix, target_matrix, nor
    """
    Source and target matrices are numpy arrays, shape
    (dictionary_length, embedding_dimension). These contai
    word vectors from the bilingual dictionary.
    """

    # optionally normalize the training vectors
    if normalize_vectors:
        source_matrix = normalized(source_matrix)
        target_matrix = normalized(target_matrix)

    # perform the SVD
    product = np.matmul(source_matrix.transpose(), target_
    U, s, V = np.linalg.svd(product)

    # return orthogonal transformation which aligns source
    return np.matmul(U, V)
```

# Results

Table 1: Translation performance using the expert training dictionary, English into Italian.

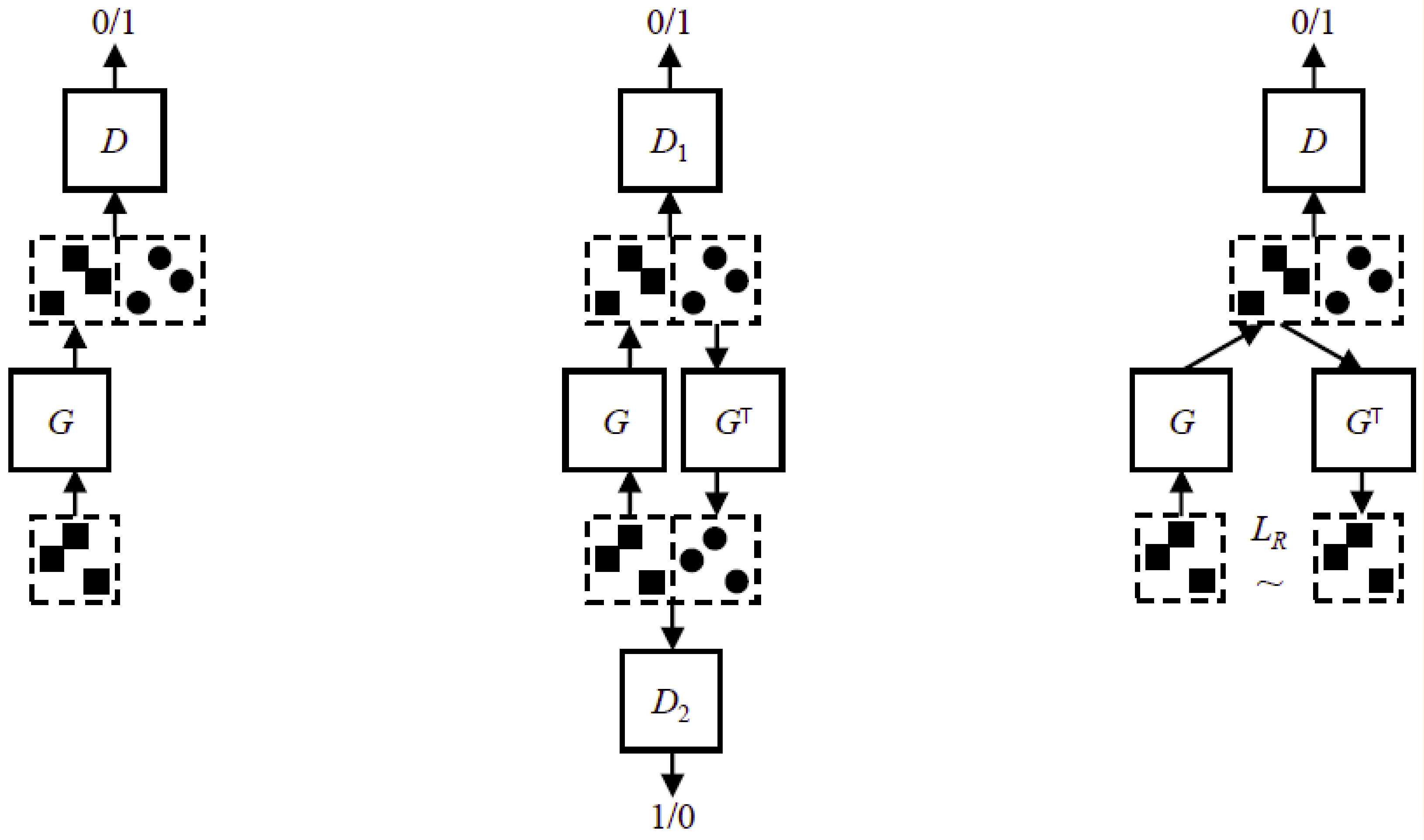
| Precision | *Mikolov et al.* | *Dinu et al.* | CCA | SVD | + inverted softmax | + dimensionality reduction |
|---|---|---|---|---|---|---|
| @1 | 0.338 | 0.385 | 0.361 | 0.369 | 0.417 | **0.431** |
| @5 | 0.483 | 0.564 | 0.527 | 0.527 | 0.587 | **0.607** |
| @10 | 0.539 | 0.639 | 0.581 | 0.579 | 0.655 | **0.664** |

Table 2: Translation performance using the expert training dictionary, Italian into English.

| Precision | *Mikolov et al.* | *Dinu et al.* | CCA | SVD | + inverted softmax | + dimensionality reduction |
|---|---|---|---|---|---|---|
| @1 | 0.249 | 0.246 | 0.310 | 0.322 | 0.373 | **0.380** |
| @5 | 0.410 | 0.454 | 0.499 | 0.496 | 0.577 | **0.585** |
| @10 | 0.474 | 0.541 | 0.570 | 0.557 | 0.631 | **0.636** |

# And if, without any cross-lingual signals as supervision

# Adversarial Game




(a) (b) (c)

# Adversarial Game

- $G$ is a mapping function: $f\ :\ \ \mathbb{R}^d \rightarrow \mathbb{R}^d$
- $D$ is a binary classfier to distiinguish between fake target word embedding $G(x)$ and real ones $y$

# Adversarial Game

- Model (a).

$$L_D = -\log D(y) - \log\left(1 - D(G(x))\right)$$

$$L_G = -\log D(G(x))$$

- Model (b).

$$L_G = -\log D_1(G(x)) - \log D_2(G^T(x))$$

- Model (c).

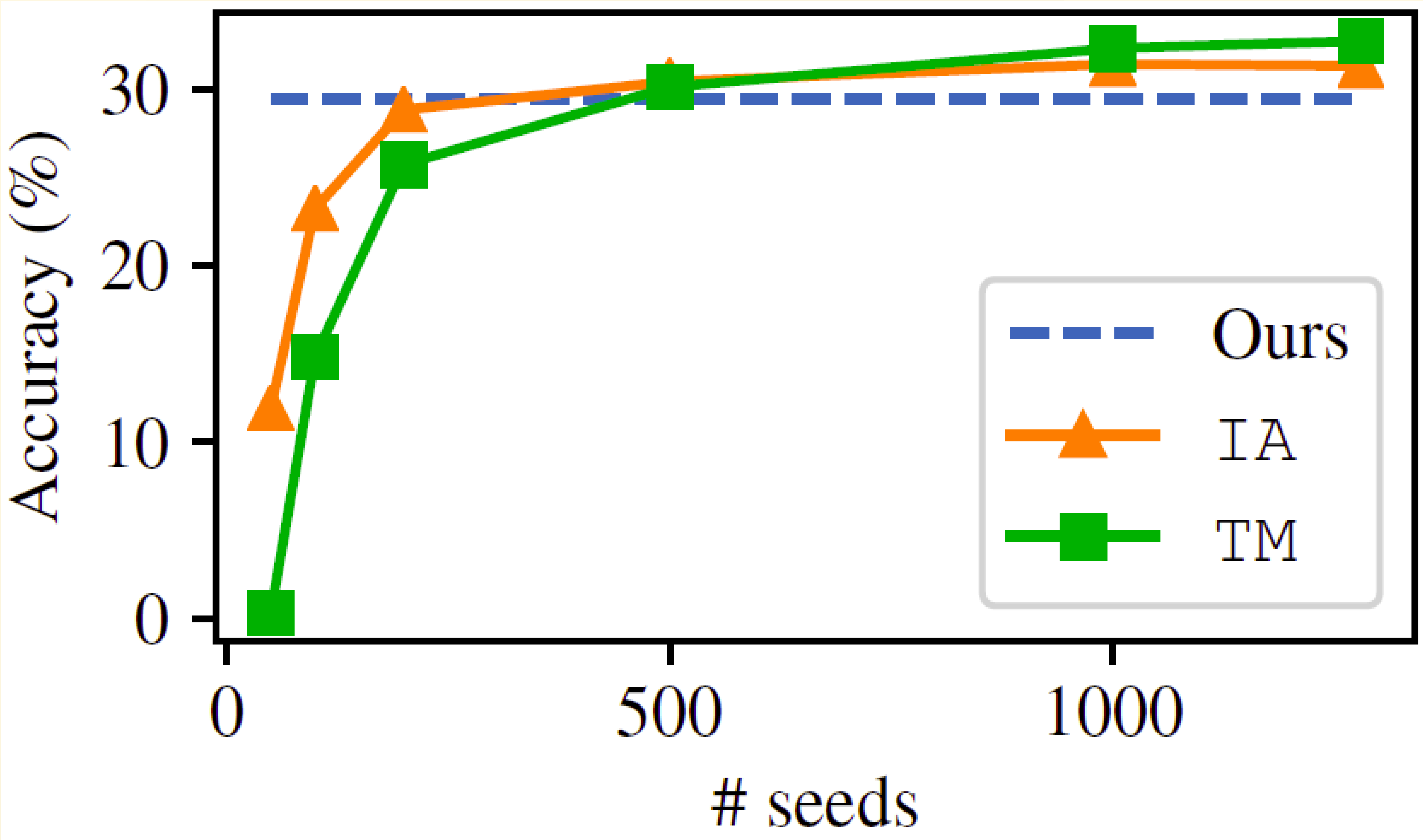
$$L_R = -\cos\left(x, G^T Gx\right)$$

# Results

| method | # seeds | accuracy (%) |
|---|---|---|
| MonoGiza w/o emb. | 0 | 0.05 |
| MonoGiza w/ emb. | 0 | 0.09 |
| TM | 50 | 0.29 |
| | 100 | 21.79 |
| IA | 50 | 18.71 |
| | 100 | 32.29 |
| Model 1 | 0 | 39.25 |
| Model 1 + ortho. | 0 | 28.62 |
| Model 2 | 0 | 40.28 |
| Model 3 | 0 | 43.31 |

Table 2: Chinese-English top-1 accuracies of the MonoGiza baseline and our models, along with the translation matrix (TM) and isometric alignment (IA) methods that utilize 50 and 100 seeds.

# Results



# Conslustion

1. Translation Matrix
2. Orthogonal Transformations
3. Adversarial Game

# Future work

1. distant language pairs
2. multilingual task
3. low-resource language