RECOGNAI

COMBINING NLP AND KNOWLEDGE-BASED METHODS

A practical perspective

Daniel Vila Suero
recogn.ai
@dvilasuero

KCAP 2017 | Austin | Texas

Who am I

- Did my PhD thesis at Ontology Engineering Group, UPM 2016 on ontology-based data access for semi-structured data.
- Founded recogn.ai in 2017
- spaCy contributor and developer since spaCy 1.X. Contributed training code and Spanish Models (embeddings, NER, Parsing, POS)

Outline

- An introduction to **NLP tasks** and **modern tooling.**
- From **named entity recognition** to **entity linking.**
- Distributed word representations in practice: finding, training and using **word vectors**.
- **Knowledge graph** embeddings.

RECOGNAI


This repository
Search
Pull requests
Issues
Marketplace
Explore
recognai / kcap17-tutorial
Unwatch 1
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Settings
Material for tutorial "Hybrid techniques for knowledge-based NLP: Knowledge graphs meet machine learning and all their friends" at KCAP 2017, Austin (Texas) http://expertsystemlab.com/kcap2017/
Edit
Add topics
5 commits
1 branch
0 releases
1 contributor
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
dvsrepo Edit README
Latest commit 287a910 just now
lib
Include writing custom components notebook
7 minutes ago
.gitignore
Include writing custom components notebook
7 minutes ago
Get started with spaCy.ipynb
Add introductory notebook
an hour ago
README.md
Edit README
just now
Writing custom components and Entity Lin...
Include writing custom components notebook
7 minutes ago
README.md
KCAP 2017 Tutorial: NLP and Knowledge-based methods with spaCy
Material for tutorial @KCAP 2017 "Hybrid techniques for knowledge-based NLP: Knowledge graphs meet machine learning and all their friends"

RECOGNAI

NLP tasks

And modern tooling

Typical workflow

- Tokenize text
- Split into sentences: sentence boundary detection
- Annotate sentences, tokens and spans: lemmas, part-of-speech tags, named entities, text categorization, etc.
- Parse syntax: dependency parsing
- Extract structured data: triples, slots, intent, etc.

Typical issues

- Need to use **different tools for different tasks**: sentence segmentation, lemmatization, POS, etc.
- **Difficult integration** across tools: pipelines.
- **Training** new models is a **cumbersome process**.

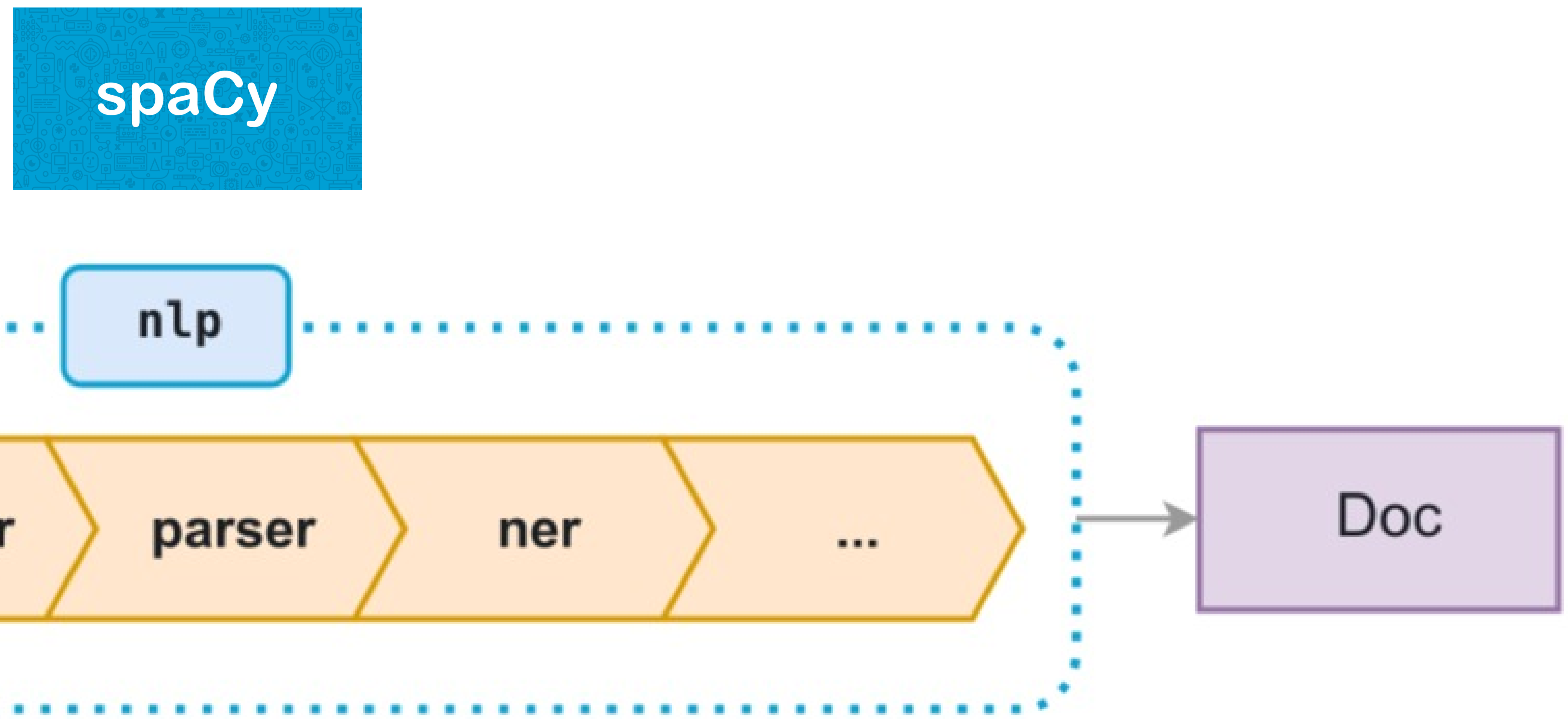
NLP toolkits

	SPACY	SYNTAXNET	NLTK	CORENLP
Programming language	Python	C++	Python	Java
Neural network models	✓	✓	✗	✓
Integrated word vectors	✓	✗	✗	✗
Multi-language support	✓	✓	✓	✓
Tokenization	✓	✓	✓	✓
Part-of-speech tagging	✓	✓	✓	✓
Sentence segmentation	✓	✓	✓	✓
Dependency parsing	✓	✓	✗	✓
Entity recognition	✓	✗	✓	✓
Coreference resolution	✗	✗	✗	✓

Source: https://spacy.io/usage/facts-figures#benchmarks

- Other tools: IXApipes, Freeling, Gate, etc.

spaCy



- High quality **documentation**, focus on **production** (and research)
- Good integration with **deep learning libraries** (e.g., PyTorch, AllenNLP).
- **Training and fine-tuning** new models is a **not an after-thought.**

Tokenization

RECOGNAI

```
import spacy

text = "This is a sentence. And this is another sentence."
nlp = spacy.load('en')
doc = nlp(text)

for token in doc:
    print(token, token.pos_)

>>
   This DET
   is VERB
   a DET
   sentence NOUN
   . PUNCT
   ......
```

RECOGNAI

Sentence boundary detection

```
import spacy

text = "This is a sentence. And this is another sentence."
nlp = spacy.load('en')
doc = nlp(text)

for sent in doc.sents:
    print([(token, token.pos_) for token in sent])

>>
  [(This, 'DET'), (is, 'VERB'), (a, 'DET'), (sentence, 'NOUN'), (., ‘PUNCT')]

  [(And, 'CCONJ'), (this, 'DET'), (is, 'VERB'), (another, 'DET'), (sentence, 'NOUN'),
  (., 'PUNCT')]
```

Part-of-speech tagging and dependency parsing

RECOGNAI

```
import spacy

text = "This is a sentence. And this is another sentence."
nlp = spacy.load('en')
doc = nlp(text)

for token in doc:
    print(token, token.pos_, token.dep_, token.head)

>>
   This DET nsubj is
   is VERB ROOT is
   a DET det sentence
   sentence NOUN attr is
   . PUNCT punct is
```

RECOGNAI

Named Entity Recognition

```
import spacy

text = “Daniel Vila is visiting Austin, Texas”
nlp = spacy.load('en')
doc = nlp(text)

for ent in doc.ents:
    print(ent, ent.label_, [token.dep_ for token in ent])



>>
  Daniel Vila PERSON ['compound', 'nsubj']
  Austin GPE ['dobj']
  Texas GPE ['appos']
```

Hands-on

Get started with spaCy notebook

RECOGNAI

This repository | Search | Pull requests | Issues | Marketplace | Explore

recognai / kcap17-tutorial

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

Material for tutorial "Hybrid techniques for knowledge-based NLP: Knowledge graphs meet machine learning and all their friends" at KCAP 2017, Austin (Texas) http://expertsystemlab.com/kcap2017/

Edit

Add topics

5 commits | **1** branch | **0** releases | **1** contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

dvsrepo Edit README — Latest commit 287a910 just now

lib	Include writing custom components notebook	7 minutes ago
.gitignore	Include writing custom components notebook	7 minutes ago
Get started with spaCy.ipynb		an hour ago
README.md	Edit README	just now
Writing custom components and Entity Lin...	Include writing custom components notebook	7 minutes ago

HANDS-ON

README.md

KCAP 2017 Tutorial: NLP and Knowledge-based methods with spaCy

Material for tutorial @KCAP 2017 "Hybrid techniques for knowledge-based NLP: Knowledge graphs meet machine learning and all their friends"

RECOGNAI

Custom components for knowledge-based methods

What and how?

What to ask your tool?

1. How easy is to integrate **custom components and pipelines**?

→ TOKENIZATION → SENTENCE SPLIT → DEP PARSING → TRIPLE EXTRACTION →

- **Example:** use dependencies tree for extracting **subject-predicate-object triples**

What to ask your tool?

1. How easy is to integrate **custom components and pipelines**?

→ TOKENIZATION → SENTENCE SPLIT → NER → ENTITY LINKING →

- **Example:** use NER annotations to perform **entity linking**

RECOGNAI

What to ask your tool?

2. Is there a consistent **data model** to **annotate** at different levels?

→ ENTITY LINKING → RELATION EXTRACTION

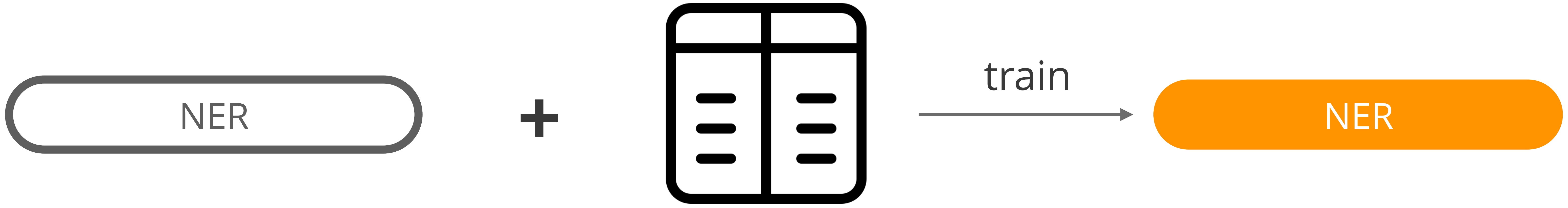
Annotations at different levels:
Tokens, Spans, Doc

- **Example:** add a **relation extraction classifier** and keep annotating, using **span-level** entity annotations and **token-level** dependency annotations

RECOGNAI

What to ask your tool?

3. How easy is to fine-tune existing models or train new ones?



BUILT-IN NER

DOMAIN CORPUS

DOMAIN NER

- **Other example:** DEPS/POS for new languages using Universal Dependencies

How? Pipelines

```
import spacy

text = “Daniel Vila is visiting Austin, Texas”
nlp = spacy.load(‘en’)

print(nlp.pipe_names) # Default processing components for en model


>>
   ['tagger', 'parser', 'ner']
```

How? Adding components

```
import spacy

def custom_processor(doc):
    # Do something with doc here: add annotations, merge spans, ...
    return doc

nlp = spacy.load('en')

nlp.add_pipe(custom_processor, name='silly_processor', first=True)

print(nlp.pipe_names)

>>
  ['tagger', 'parser', 'ner','silly_processor']
```

How?
Adding stateful components

RECOGNAI

```
class CustomComponent(object):
    name = 'still_silly'

    def __init__(self, config):
        # We can initialize this with settings
        self.config = config

    def  __call__(self, doc):
        # Do things
        return doc

nlp = spacy.load('en')
custom_component = CustomComponent({})
nlp.add_pipe(custom_component)
print(nlp.pipe_names)                >> [‘tagger’, 'parser', ‘ner’,'still_silly']
```

How?
Adding attributes (labels, annotations..)

RECOGNAI

```
from spacy.tokens import Doc, Span, Token

experiment_keywords = ['experiment', 'results', 'validation', 'experimental']

is_experiment_keyword =
   lambda token: token.lower_ in experiment_keywords

is_experiment_part =
   lambda text: any([token.lower_ in experiment_keywords for token in text])

Token.set_extension('is_experiment_keyword', getter=is_experiment_keyword)
Doc.set_extension(‘has_experiment_part', getter=is_experiment_part)
Span.set_extension('is_experiment_part', getter=is_experiment_part)
```

How?
Adding attributes (labels, annotations..)

RECOGNAI

```
doc = nlp(u"This section presents the experimental results.")

print(doc._.has_experiment_part)
print(doc[4:5]._.is_experiment_part)
print(doc[1:2]._.is_experiment_part)




>> True, True, False
```

RECOGNAI

Detecting and disambiguating entities

Hands on:From entity recognition to entity linking

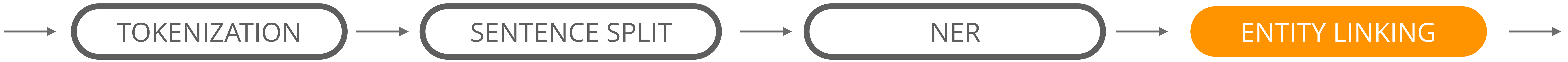
RECOGNAI


This repository
Search
Pull requests
Issues
Marketplace
Explore
recognai / kcap17-tutorial
Unwatch 1
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Settings
Material for tutorial "Hybrid techniques for knowledge-based NLP: Knowledge graphs meet machine learning and all their friends" at KCAP 2017, Austin (Texas) http://expertsystemlab.com/kcap2017/
Edit
Add topics
5 commits
1 branch
0 releases
1 contributor
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
dvsrepo Edit README
Latest commit 287a910 just now
lib
Include writing custom components notebook
7 minutes ago
.gitignore
Include writing custom components notebook
7 minutes ago
Get started with spaCy.ipynb
Add introductory notebook
an hour ago
README.md
Edit README
just now
Writing custom components and Entity Lin...
otebook
7 minutes ago
HANDS-ON
README.md
KCAP 2017 Tutorial: NLP and Knowledge-based methods with spaCy
Material for tutorial @KCAP 2017 "Hybrid techniques for knowledge-based NLP: Knowledge graphs meet machine learning and all their friends"

RECOGNAI

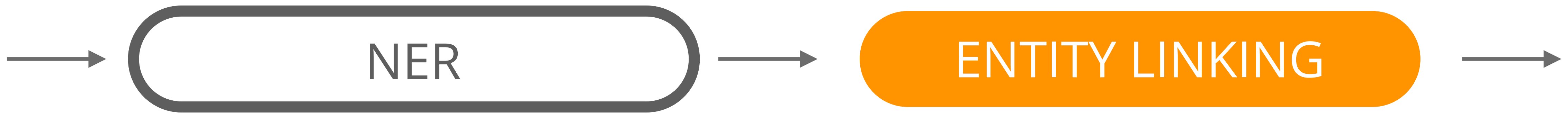
Basic idea

Create a **custom entity linking component** consuming NER annotations and extending them with URIs to LOD entities.



RECOGNAI

Basic steps



- **Init** the custom entity linking component with the service URL.
- **Get entity annotations** from built-in NER component.
- **Call the service** with NER annotations.
- **Get potential links** from service.
- **Annotate spaCy Doc** and **Spans** accordingly

Entity Linking highlighted methods

RECOGNAI

- Babelfy:
  - ***Entity Linking meets Word Sense Disambiguation: a Unified Approach. (Moro et al., TACL 2014)***
  - Joint identification of senses and entities in BabelNet
- Agdistis: Best paper award ISWC'14, new version to be presented at KCAP-2017!
  - ***AGDISTIS - Graph-Based Disambiguation of Named Entities using Linked Data (Usbeck et al. ISWC 2017)***
  - ***MAG: A Multilingual, Knowledge-base Agnostic and Deterministic Entity Linking Approach (Moussallem et al. KCAP 2017)***
  - Multilingual, knowledge-based agnostic (DBpedia, Wikidata)

Hands-on

RECOGNAI

Using AGDISTIS services for linking entities to DBpedia

- Check `lib/linkers.py`
- Use AgdistisEntityLinker
- Get similarities across surface forms of linked entities: which is the most dissimilar?

Custom linker design

RECOGNAI

```
from spacy.tokens import Doc, Span, Token

class AgdistisEntityLinker(object):

    name = 'agdistis_linker'

    def __init__(self, lang='en', param='text'):
        # Setup language and service
        # Setup custom extension attributes for Span and Doc

    def __call__(doc):
        # Use NER annotations from doc.ents to build endpoint query
        # Call service endpoint
        # Add entity linking annotations to doc and ent Spans
        return doc
```

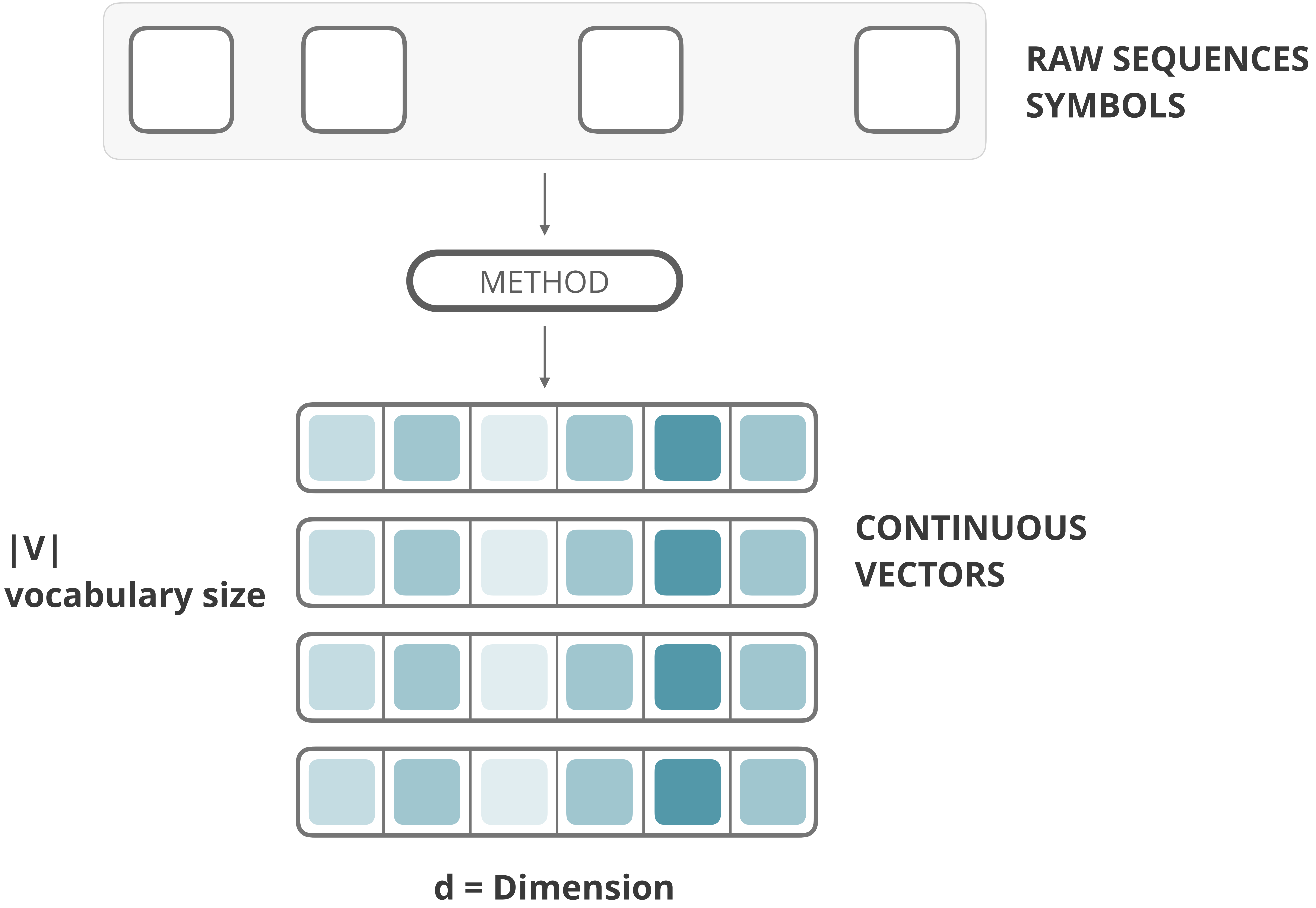
RECOGNAI


This repository
Search
Pull requests
Issues
Marketplace
Explore
recognai / kcap17-tutorial
Unwatch 1
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Settings
Material for tutorial "Hybrid techniques for knowledge-based NLP: Knowledge graphs meet machine learning and all their friends" at KCAP 2017, Austin (Texas) http://expertsystemlab.com/kcap2017/
Edit
Add topics
5 commits
1 branch
0 releases
1 contributor
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
dvsrepo Edit README
Latest commit 287a910 just now
lib
Include writing custom components notebook
7 minutes ago
.gitignore
Include writing custom components notebook
7 minutes ago
Get started with spaCy.ipynb
Add introductory notebook
an hour ago
README.md
Edit README
just now
Writing custom components and Entity Lin...
Include writing custom components notebook
7 minutes ago
README.md
KCAP 2017 Tutorial: NLP and Knowledge-based methods with spaCy
Material for tutorial @KCAP 2017 "Hybrid techniques for knowledge-based NLP: Knowledge graphs meet machine learning and all their friends"

Distributed word representations

Intuitions and practice

Components


RAW SEQUENCES
SYMBOLS
METHOD
|V|
vocabulary size
CONTINUOUS
VECTORS
d = Dimension

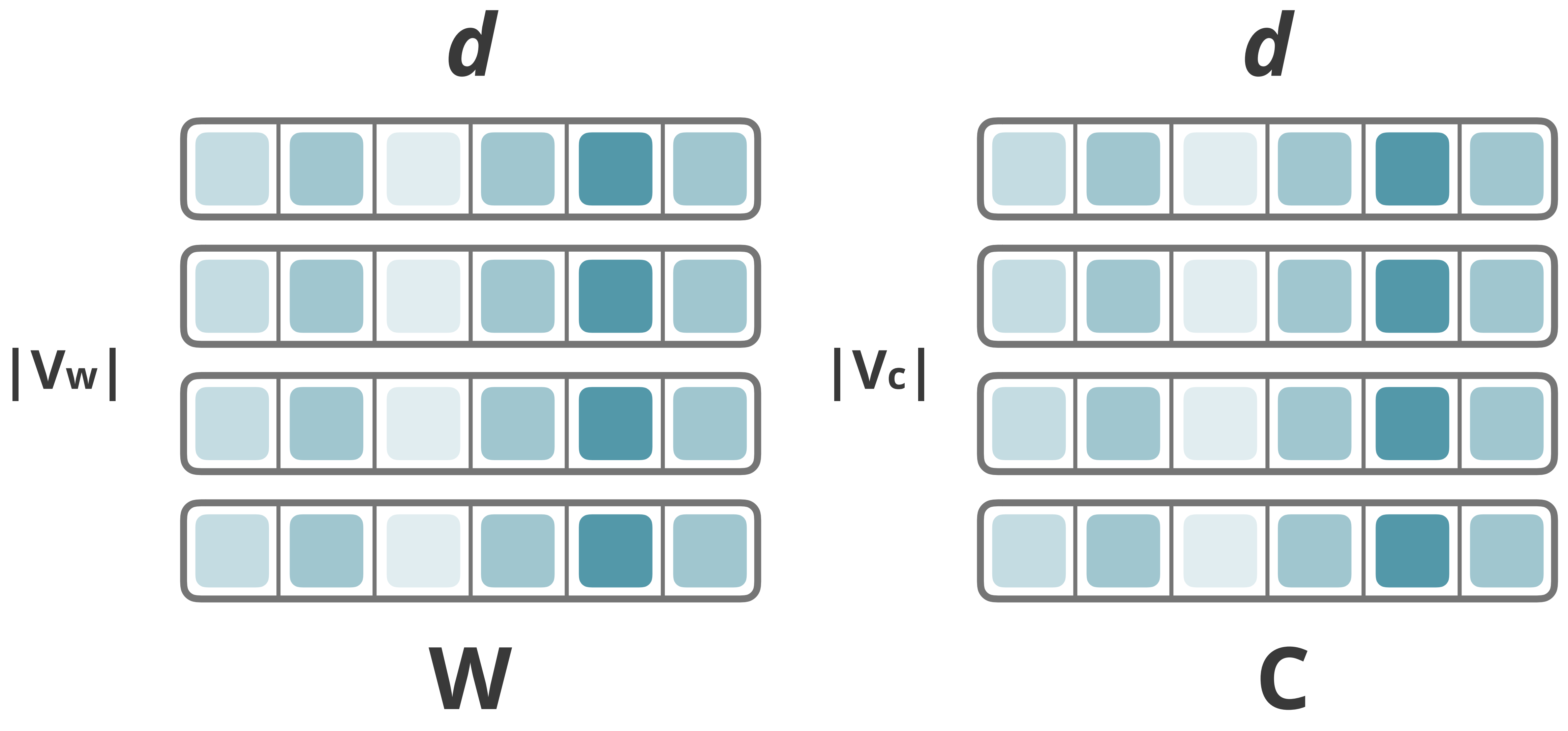
word2vec

- *Distributed representations of words and phrases and their compositionality (Mikolov et al. NIPS 2013)*
- Widely used **C implementation** and **pre-trained vectors**
- Two **training methods**:
    - **Negative sampling**
    - Hierarchical softmax
- Two **context representations**:
    - Continuous Bag of Words (CBOW)
    - **Skip-grams**

RECOGNAI

word2vec

- Represent each word as ***d*** dimensional vector.
- Represent each context as ***d*** dimensional vector.
- Init all vectors to random weights.
- Arrange vectors in two matrices.



RECOGNAI

word2vec: SGNS

- while(text):
  - Extract a window:

    `Turing was [educated at Hazelhurst Preparatory School]`

    c_1 c_2 w c_3 c_4

  - Try setting the vector values such that:

    $p(wc_1) + p(wc_2) + p(wc_3) + p(wc_4)$ is **high**

  - Create a corrupt example by picking a random word ***w'***

    `Turing was [educated at banana Preparatory School]`

    c_1 c_2 w' c_3 c_4

  - Try setting the vector values such that:

    $p(w'c_1) + p(w'c_2) + p(w'c_3) + p(w'c_4)$ is **low**

word2vec

RECOGNAI

- Result intuitions:
  - ***wc*** for **good** word-context pairs is **high**
  - ***wc*** for **bad** word-context pairs is **low**
  - **Words that share many contexts** get **close** to each other
  - **Contexts that share many words** get **close** to each other

Interesting reads

- Similar to Distributional lexical semantics methods:
  - ***Neural Word Embeddings as Implicit Matrix Factorization (Levy and Goldberg, NIPS 2014):***
    - SGNS very similar to factorizing traditional word-context PMI matrix.
- Big **impact of algorithmic and hyper-parameter choices:**
  - ***Improving Distributional Similarity with Lessons Learned from Word Embeddings (Levy, Goldberg, Dagan, ACL 2015)***
- Sebastian Ruder's blog post series **"On Word Embeddings"**

Other methods

RECOGNAI

- GloVe:
  - *Glove: Global vectors for word representation (Pennington, Socher, Manning, ACL 2014)*
  - Fast and implementation + pre-trained English vectors available
- Facebook's Fasttext:
  - *Enriching Word Vectors with Subword Information (Bojanowski, Grave, Joulin, Mikolov, 2016)*
  - Fast and implementation + pre-trained multi-language vectors available
- Google's Swievel
  - *Swivel: Improving Embeddings by Noticing What's Missing (Shazeer, et al. 2016)*
  - Tensorflow implementation available

RECOGNAI

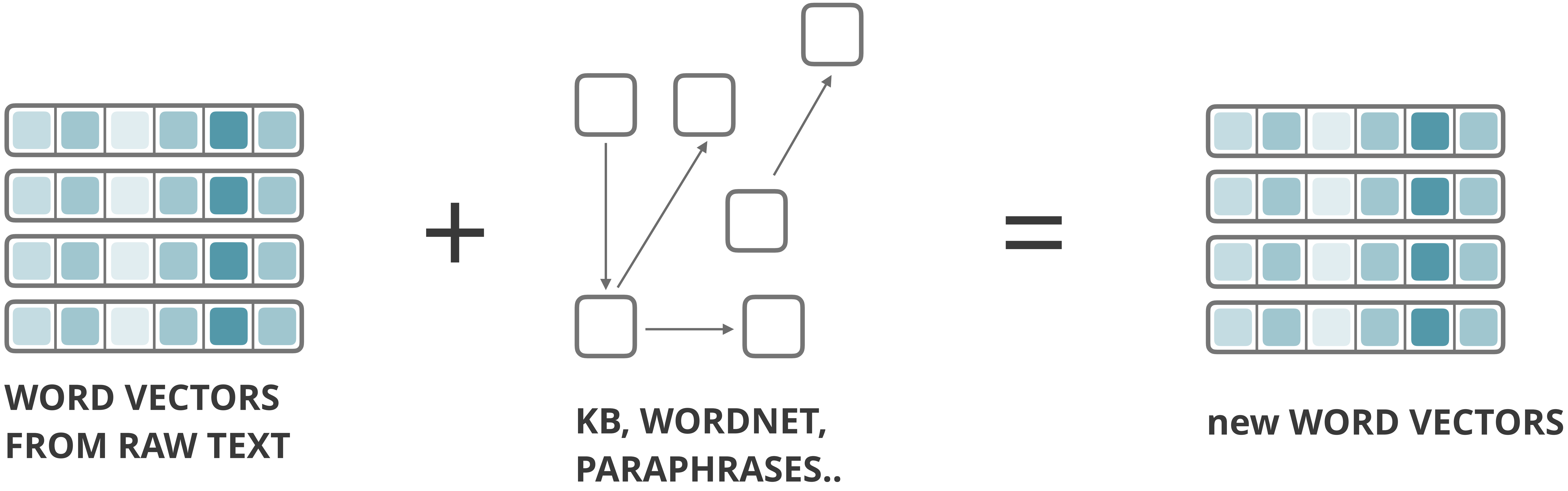
Enriching word representations

Using external knowledge

RECOGNAI

Enriching representations with external knowledge

- Introduce knowledge from external sources to enforce external semantics within learned word representations



Enriching representations with external knowledge

- Two types of approach (Joint learning and retrofitting):

1. Modify training process and objective: Jointly learning word representations and semantic relations together.

   - ***Improving lexical embeddings with semantic knowledge (Yu and Dredze, ACL 2014).*** Uses PPDB and Wordnet and modifies word2vec training with a linear combination of objectives (CBOW + RCM). Code available at https://github.com/Gorov/JointRCM

   - ***Rc-net: A general framework for incorporating knowledge into word representations (Xu et al. 2014).*** Relational and categorical knowledge.

RC-net

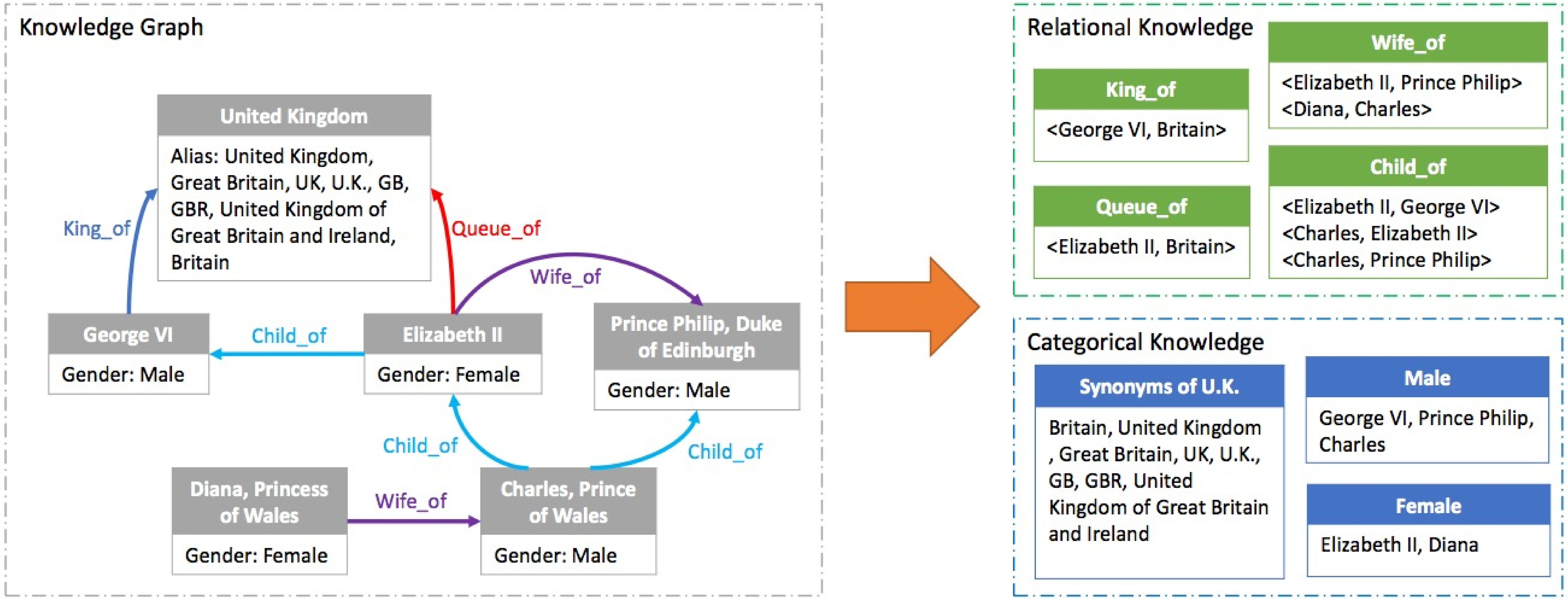


Figure 1: Knowledge graph contains two forms of knowledge: relational knowledge and categorical knowledge.

Rc-net: A general framework for incorporating knowledge into word representations (Xu et al. 2014)

RECOGNAI

RC-net

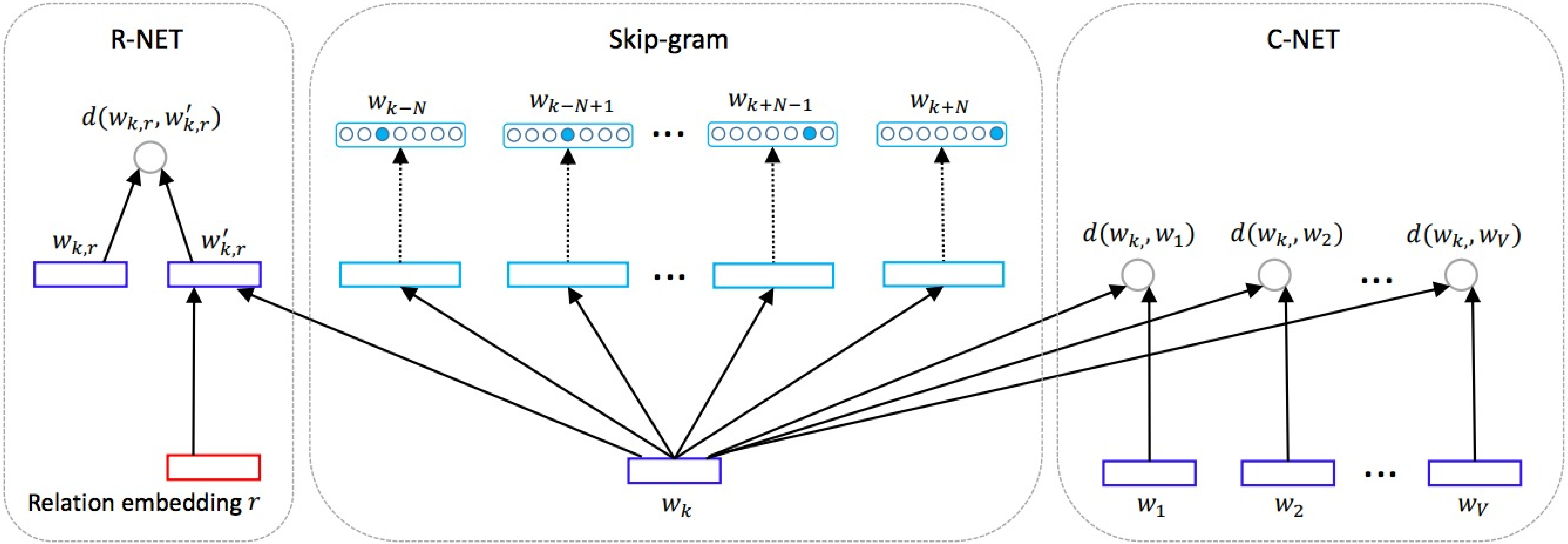


Figure 4: The architecture of RC-NET. The objective is to learn word representations and relation representations based on text stream, relational knowledge, and categorical knowledge.

Rc-net: A general framework for incorporating knowledge into word representations (Xu et al. 2014)

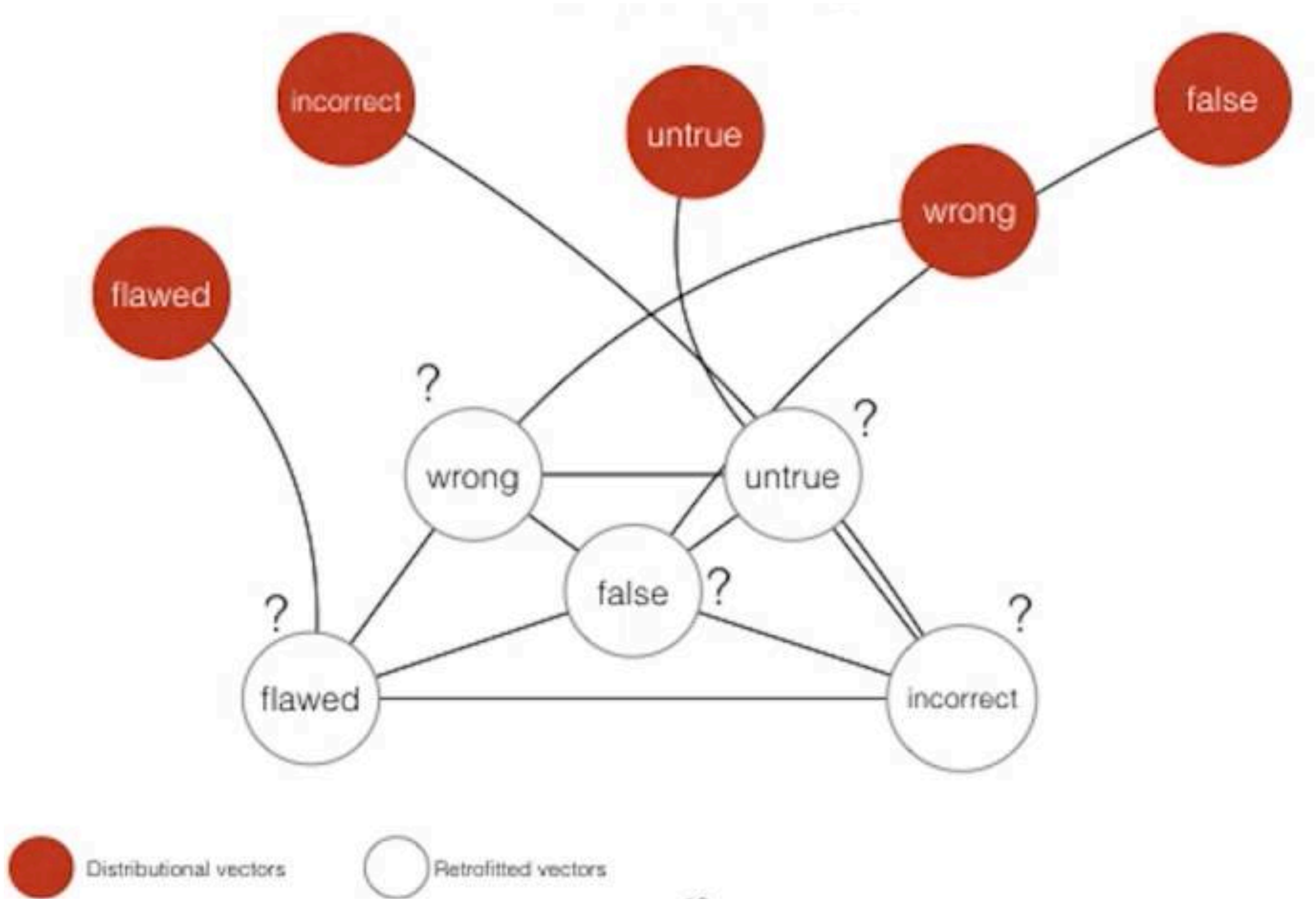
Enriching representations with external knowledge: Retrofitting

RECOGNAI

- *Retrofitting Word Vectors to Semantic Lexicons (Faruqui et al, 2011)*
- Adjust learned word vectors with Semantic Lexicons
- Advantage: Applicable to word vectors trained with any other method.
- Intuition: word reps learned are still close to each other but also close to neighbours in semantic lexicon.
- Code available: https://github.com/mfaruqui/retrofitting

Enriching representations with external knowledge

RECOGNAI



Retrofitting Word Vectors to Semantic Lexicons (Faruqui et al, 2011)

Retrofitting

```
def retrofit(wordVecs, lexicon, numIters):
  newWordVecs = deepcopy(wordVecs)
  wvVocab = set(newWordVecs.keys())
  loopVocab = wvVocab.intersection(set(lexicon.keys()))
  for it in range(numIters):
    # loop through every node also in ontology (else just use data estimate)
    for word in loopVocab:
      wordNeighbours = set(lexicon[word]).intersection(wvVocab)
      numNeighbours = len(wordNeighbours)
      #no neighbours, pass - use data estimate
      if numNeighbours == 0:
        continue
      # the weight of the data estimate if the number of neighbours
      newVec = numNeighbours * wordVecs[word]
      # loop over neighbours and add to new vector (currently with weight 1)
      for ppWord in wordNeighbours:
        newVec += newWordVecs[ppWord]
      newWordVecs[word] = newVec/(2*numNeighbours)
  return newWordVecs
```

RECOGNAI

Using word vectors

What, where and how

Training new vectors

- C libraries:
  - **GloVe**.
  - Facebook's **Fasttext**.
- **Gensim**: Python library for unsupervised methods over large corpora
  - Includes bindings and re-implementations of most common algorithms
  - Extra tooling for visualization, corpus management, evaluation.

Where to find raw text

- *N-gram counts and language models from the CommonCrawl (LREC 2014):*
  - *Deduped multiple langs: http://data.statmt.org/ngrams/deduped/*
- CommonCrawl:
  - ***LanguageCrawl: A Generic Tool for Building Language Models Upon Common-Crawl, (Szymon Roziewski, Wojciech Stokowiec, LREC 2016)***

Where to find vectors

- ***spaCy*** *medium and large models ship with* ***pre-trained vectors.***
- ***Fasttext*** *trained on Wikipedia for* ***294 languages****:*
  - *https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md*

RECOGNAI

Tutorial:
Using fastText vectors for tweet classification

RECOGNAI

Embedding knowledge graphs

Introduction

Introduction

- Similar intuition as word2vec
- Embed a knowledge graph in a latent space, where:
  - Entities and relations are represented as continuous vectors and similar entities/relations are close to each other.
- Overview of methods:
  - ***A Review of Relational Machine Learning for Knowledge Graphs (Nickel et al. 2015)***

Tasks

- **Link prediction** for knowledge base completion: Predict probability of missing facts.
- **Automatic knowledge base construction**: Combined with text-based information extraction and relation extraction to build KGs from raw text.
- Others: **Entity resolution, link-based clustering, question answering**

RECOGNAI

Example application

- *Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion (Dong et al. KDD 2014)*

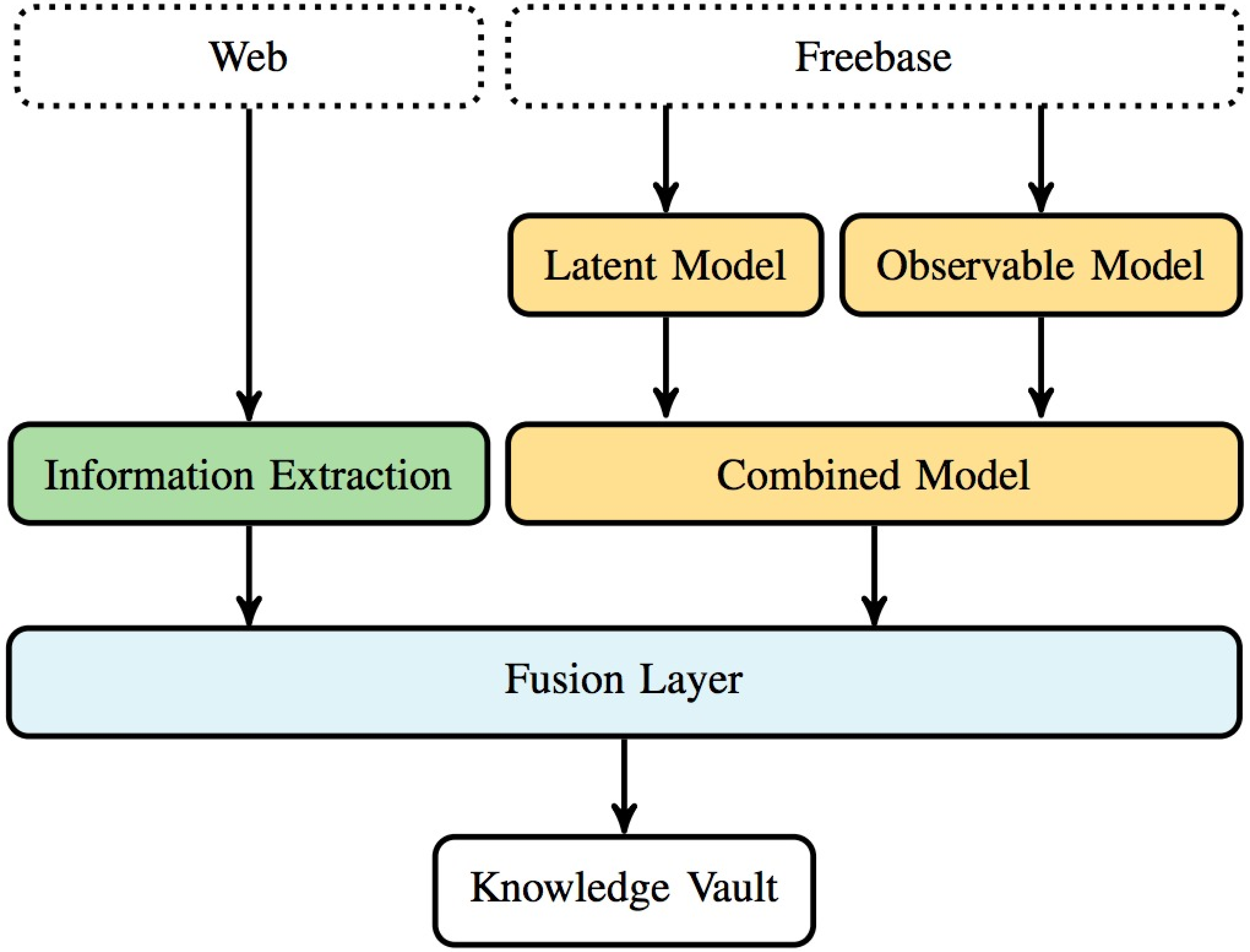


Fig. 7. Architecture of the Knowledge Vault.

RECOGNAI

Methods

Method	WN18		FB15k	
	raw	filtered	raw	filtered
TransE (Bordes et al., 2013)	75.4	89.2	34.9	47.1
Rescal (Nickel et al., 2012)	-	92.8	-	58.7
Fast-TransR (Lin et al., 2015)	81.0	94.6	48.8	69.8
HolE (Nickel et al., 2016)	-	94.9	-	73.9
TransE++ (Nickel et al., 2016)	-	94.3	-	74.9
Fast-TransD (Lin et al., 2015)	78.5	91.9	49.9	75.2
ReverseModel (Dettmers et al., 2017)	-	96.9	-	78.6
HolE+Neg-LL (Trouillon and Nickel, 2017)	-	94.7	-	82.5
Complex (Trouillon et al., 2017)	-	94.7	-	84.0
R-GCN (Schlichtkrull et al., 2017)	-	96.4	-	84.2
ConvE (Dettmers et al., 2017)	-	95.5	-	87.3
DistMul (Kadlec et al., 2017)	-	94.6	-	89.3
Ensemble DistMul (Kadlec et al., 2017)	-	95.0	-	90.4
IRN (Shen et al., 2016)	-	95.3	-	92.7
`fastText` - train	80.6	94.9	52.3	86.5
`fastText` - train+valid	83.2	97.6	53.4	89.9

Table 1: Raw and filtered Hit@10 on WN18 and FB15k. All the numbers are taken from their paper. Above, methods that should achieve better performance with a finer hyper-parameter grid, below, methods that were properly tuned. Higher the better.

Fast Linear Model for Knowledge Graph Embeddings (Joulin et al. AKBC WS-NIPS 2017)

TransE

- Inspired by word2vec
- Represent relations as translation in the embedding space:
  - Subject + Relation type ≈ object
- Training method: Negative sampling, **corrupted triples**.
- *Translating Embeddings for Modeling Multi-relational Data (Bordes et al. NIPS 2013)Fast Linear Model for Knowledge Graph Embeddings*

RECOGNAI

Implementations

- scikit-kge library by Nickel: https://github.com/mnick/scikit-kge
- ConvE (Convolutional 2D Knowledge Graph Embeddings) in PyTorch by Dettmers: https://github.com/TimDettmers/ConvE
- KGE-server (**V. Fernández Rico** Ontology Engineering Group), extends scikit-kge with:
    - Build train/test data from SPARQL queries
    - Scalable training with different methods (GloVE, TransE, etc.)
    - Index embedding vectors with Spotify Annoy for similarity endpoints.

RECOGNAI

KGE-server

RECOGNAI

KGE-server

RECOGNAI

KGE-server

KGE-server

RECOGNAI

RECOGNAI

THANK YOU!

daniel@recogn.ai
@dvilasuero