Deep Learning for Natural Language Processing

Lecture 5 – Bilingual and Syntax-Based Word Embeddings

Dr. Ivan Habernal

May 11, 2021

Trustworthy Human Language Technologies
Department of Computer Science
Technical University of Darmstadt

www.trusthlt.org

This lecture

Multi-Lingual, Multi-Sense Word Embeddings

Syntactic Word Embeddings

Miscellaneous

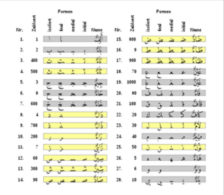
Word Senses

Words do not represent only one meaning

Problem is generally known as polysemy (or even homonymy): a word may have many different meanings:

- bank, **table**, fly, man, ...

A table (furniture)



A table of characters in the Arabic alphabet

Figure 1: Source: Wiktionary

Word Senses

Man

1. The human species (man vs. other organism)
2. Males of the human species (i.e. man vs. woman)
3. Adult males of the human species

This example shows the specific polysemy where the same word is used at different levels of a taxonomy.

Example 1 contains example 2, and example 2 contains example 3.

Sense-disambiguated word representations

Idea: Train word vectors on sense-disambiguated corpora

"a rush of panic caught Sarah"[1]

```
<s snum="132">
  <wf pos="DT">A</wf>
  <wf pos="NN" lemma="rush" wnsn="2">rush</wf>
  <wf pos="IN">of</wf>
  <wf pos="NN" lemma="panic" wnsn="1">panic</wf>
  <wf pos="VB" lemma="catch" wnsn="12">caught</wf>
  <wf pos="NNP" lemma="person" wnsn="1">Sarah</wf>
  <punc>.</punc>
</s>
```

```
=> A rush_2 of panic_1 caught_12 Sarah_1
```

[1] Shortened example from *SemCor* corpus. Not all words have different senses; function words and punctuation do not have senses

Sense-disambiguated word representations

bank$_1^n$ (geographical)	***bank***$_2^n$ (financial)	***number***$_4^n$ (phone)	***number***$_3^n$ (acting)	***hood***$_1^n$ (gang)	***hood***$_{12}^n$ (convertible car)
upstream$_1^r$	commercial_bank$_1^n$	calls$_1^n$	appearing$_6^v$	tortures$_5^n$	taillights$_1^n$
downstream$_1^r$	financial_institution$_1^n$	dialled$_1^v$	minor_roles$_1^n$	vengeance$_1^n$	grille$_2^n$
runs$_6^v$	national_bank$_1^n$	operator$_{20}^n$	stage_production$_1^n$	badguy$_1^n$	bumper$_2^n$
confluence$_1^n$	trust_company$_1^n$	telephone_network$_1^n$	supporting_roles$_1^n$	brutal$_1^a$	fascia$_2^n$
river$_1^n$	savings_bank$_1^n$	telephony$_1^n$	leading_roles$_1^n$	execution$_1^n$	rear_window$_1^n$
stream$_1^n$	banking$_1^n$	subscriber$_2^n$	stage_shows$_1^n$	murders$_1^n$	headlights$_1^n$

Table 1: Closest senses to two senses of three ambiguous nouns: *bank*, *number*, and *hood*

Figure 2: Result: different representations for each sense[2]

Note: subscript is sense-id superscript is pos-tag Number and bank could also appear as verbs (not illustrated here)

[2] I. Iacobacci, M. T. Pilehvar, and R. Navigli (2015). “SensEmbed: Learning Sense Embeddings for Word and Relational Similarity”. In: *Proceedings of ACL*. Beijing, China: Association for Computational Linguistics, pp. 95–105

Problems

How do you now train an NLP system with these sense-disambiguated embeddings?

A more parsimonious approach

Run word2vec on your data and compute embeddings

For each target word, represent its context as avg. or concatenated embedding

- … need to go to the *bank* to get some money …
- … debt by utilizing a credit line granted by a *bank* …
- … raw water is largely river *bank* filtrate (approximately 70 percent) …
- … runs from its idyllic river *bank* promenade under the Elbe to …

A more parsimonious approach

Run word2vec on your data and compute embeddings

For each target word, represent its **context** as avg. or concatenated embedding

- … need **to go to the** *bank* **to get some money** …
- … debt by utilizing a **credit line granted by a** *bank* …
- … raw **water is largely river** *bank* **filtrate (approximately 70 percent)** …
- … runs **from its idyllic river** *bank* **promenade under the Elbe** to …

A more parsimonious approach

- … need **to go to the** *bank* **to get some money** …
  $\rightarrow [.2, .8]$
- … debt by utilizing a **credit line granted by a** *bank* …
  $\rightarrow [.4, .6]$
- … raw **water is largely river** *bank* **filtrate (approximately 70 percent)** … $\rightarrow [-.2, -.8]$
- … runs **from its idyllic river** *bank* **promenade under the Elbe** to … $\rightarrow [-.9, -.3]$

Cluster the context representations (unsupervised!)

Assign each word's context to a cluster: the word has the sense corresponding to the cluster index

Run word2vec on sense-disambiguated corpus

Sense-disambiguated word representations

Promising approach to unsupervised sense-disambiguated word representation

On the other hand, the cost is much higher — one needs a sense-labeler or a more complicated model

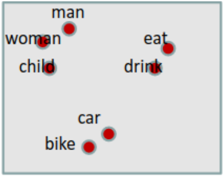
Hardly used in practice

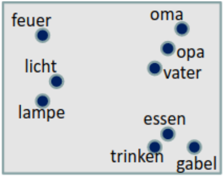
Before ELMo and BERT came around in 2018 with **contextualized word embeddings**

Bilingual Embeddings

Word representations for two languages:

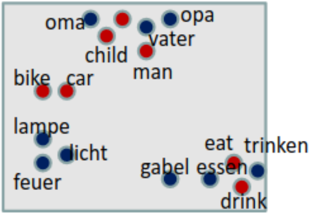
train on corpus from both languages





Bilingual Embeddings

Goal: represent words from different languages in the same space



Bilingual Embeddings – General idea

Can think of it as having two objectives we want to satisfy

- cross-lingual objective: words that are translations of each other should be close in the projected space
- mono-lingual objective: words that occur in monolingually similar contexts should be close to each other in vector space

Bilinguality – Why?

(1) Second language may act as an additional ”signal”

- Which may help to improve word embeddings even in the first language
  $\rightarrow$ *Make Monolingual Embeddings better*
- E.g. assume that some word like “opa” occurs very infrequently in the German corpus, thus it’s difficult to reliably estimate its word embedding
- If its English translation “grandfather” occurs frequently in the English corpus, the German word should get a more appropriate embedding in the bilingual space

Bilinguality – Why?

(2) If words are projected in a common space (“shared features”), this may allow for **direct transfer**

- Train a model in one language (usually resource-rich)
- Directly apply in another language (usually resource-poor)

Bilinguality – Example

(2) Example Direct Transfer: task is POS tagging

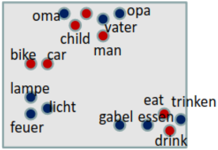
Goal / approach:

Train: *I may not drink this* → PRON VERB PARTICLE VERB DET

Test: *Es ist wichtig, ausreichend zu trinken* → ?

Training (idea):

Input: center words with their context words
Output: labels of center word
E.g. (not, **drink**, this) → VERB



Bilinguality – Example

(2) Example Direct Transfer: task is POS tagging

Direct transfer aka zero-shot transfer:

- train using bilingual embeddings in English (assume big labeled English dataset)

- then apply to German data

Problems with the Direct Transfer approach?

- ”OOV words”

- syntactic ordering

Bilingual Embeddings – Naive Approach

Given 1: Monolingual Embeddings (e.g. English, German)

Given 2: Dictionary EN $\Leftrightarrow$ DE

Translate German words to English words, assign them the embedding of the English word (or concatenate, average, ...)

- Bottleneck is the dictionary
- Cannot assign meanings to words that are not in the dictionary

Bilingual Embeddings

More sophisticated approaches have been suggested, relying on different kinds of (costly) information

Approach 1: Learning a transformation matrix

- One of the first and simplest approaches
  - Mikolov et al. 2013, Exploiting similarities among languages for machine translation
- Given: monolingual embeddings + dictionary
  - Dictionary: cat-Katze, table-Tisch, ...

x_i	z_i
cat	Katze
table	Tisch
...	...

Approach 1: Learning a transformation matrix

$\mathbf{x}_i$	$\mathbf{z}_i$
$[0.2, -0.3, 0.8]$	$[0.5, 0.9, -1]$
$[1, 2, -5]$	$[0.1, -0.1, 0.1]$
...	...

We estimate a linear transformation from this data:

$$\min_{\mathbf{W}} \sum_i \|\mathbf{x}_i \mathbf{W} - \mathbf{z}_i\|_2$$

- $\mathbf{x}_i$ and $\mathbf{z}_i$: monolingual word vectors from dictionary

Once $\mathbf{W}$ is learned, we can map any language $\mathbf{x}$ word into the space of language $\mathbf{z}$

Even words for which we do not have translations

More Bilingual Embeddings – Survey papers

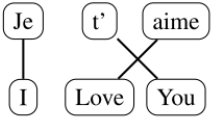
See S. Upadhyay et al. (2016). “Cross-lingual Models of Word Embeddings: An Empirical Comparison”. In: *Proceedings of ACL*. Berlin, Germany, pp. 1661–1670

And more recent G. Glavaš et al. (2019). “How to (Properly) Evaluate Cross-Lingual Word Embeddings: On Strong Baselines, Comparative Analyses, and Some Misconceptions”. In: *Proceedings of ACL*. Florence, Italy, pp. 710–721

Bilingual embeddings

BiSkip uses sentence and word aligned texts, then runs a skip-gram model whose contexts are words from both languages:

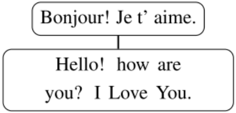
- E.g. on input *love* BiSkip wants to predict the context *je*, *I*, *you*, *t'*;
- similar for *aime*: *t'*, *you*
- $\rightarrow$ similar contexts are predicted $\rightarrow$ similar representations



Bilingual embeddings

BiVCD[3] is even simpler. Given aligned documents (e.g. Wikipedia articles)

- Merge them, then random shuffle all words
- Then run a Monolingual Model (e.g. CBOW, Glove, Skip-Gram) on it



[3] I. Vulić and M.-F. Moens (2015). "Bilingual Word Embeddings from Non-Parallel Document-Aligned Data Applied to Bilingual Lexicon Induction". In: *Proceedings of ACL (Volume 2: Short Papers)*. Beijing, China, pp. 719–725

Determining bi-lingual mappings (for BiSkip)

Dictionary

Inter-lingual links in Wikipedia

Word alignments learned from parallel corpora

Multilinguality

- We talked about mapping two languages in a common space
- How about 3, 5, 10 languages?
- Much less explored topic
- However, there is work on it, such as Ammar et al. (2016), Massively Multilingual word embeddings
    - They extend BiCCA to MultiCCA and BiSkip to MultiSkip

In recent years, Multilingual BERT (MBERT), which yields embeddings in a joint space for 100+ languages

Current trends

- Learn bilingual word embeddings from as few resources as possible
- E.g., only 10 aligned word pairs (can be punctuation)
- From there we can go to unsupervised machine translation

Current trends

M. Artetxe, G. Labaka, and E. Agirre (2017). "Learning bilingual word embeddings with (almost) no bilingual data". In: *Proceedings of ACL*. Vancouver, Canada, pp. 451–462

Main idea:

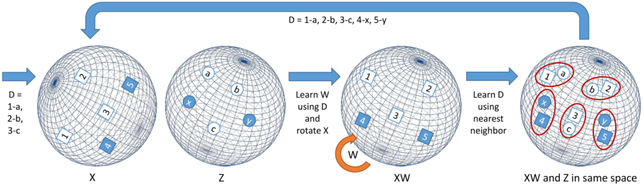
– If we had a dictionary, we can get bilingual embeddings

– If we had bilingual embeddings, we can get a dictionary

Current trends

Artetxe, Labaka, and Agirre's (2017) method:

1. Use a lexicon (seed lexicon is easy to get automatically)
2. Learn bilingual embeddings using current lexicon ($\rightarrow$ Mikolov's method, i.e., "Approach 1")
3. Get a better lexicon using bilingual embeddings
4. Go back to 1)



Syntactic word embeddings

More syntactically oriented embeddings

Syntactic relations between words should also be represented in the vectors

– Problem: word order matters

Dog bites man. vs. *Man bites dog.*

Position Information

Remember: The `word2vec` models do not consider position information:

- No distinction between left and right context
- No distinction between close and far contexts

Skip-gram: ____ bites ____
$\rightarrow$ (bites, man) , (bites, dog)

Position Information

dog bites man vs. *man bites dog*

`(bites, dog-1), (bites, man+1)`

vs.

`(bites, man-1), (bites, dog+1)`

This is “intuitively” what we want (although we don’t add indices to words)

Skip-gram model

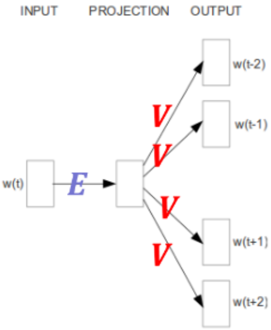


Figure 3: SkipGram model

How can we predict different words when V is always the same?

Structured Skip-gram model

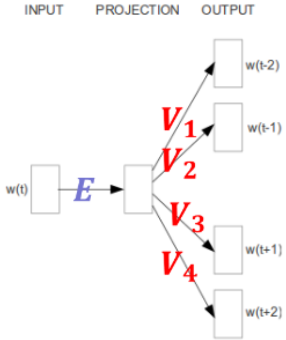


Figure 4: Structured SkipGram model

Results

Nearest neighbours for *breaking*

Skip-gram	Structured Skip-gram
breaks	putting
turning	turning
broke	sticking
break	pulling
stumbled	picking

Word representations with positional information work slightly better for syntactic tasks like POS-tagging and parsing.

W. Ling et al. (2015). "Two/Too Simple Adaptations of Word2Vec for Syntax Problems". In: *Proceedings of NAACL*. Denver, Colorado, pp. 1299–1304

Long-distance dependencies

Words can be similar with respect to verb selection preferences

– tea/milk/beer/coffee can all be an object of the verb *drink*

Words that share syntactic relations might be distant in a sentence:

I would like to ***drink*** *a very hot tall decaf half-soy (...insert any other thousand options ...) white chocolate* ***mocha***

Dependency parsing in one slide

Grammatical relationships between words in a sentence

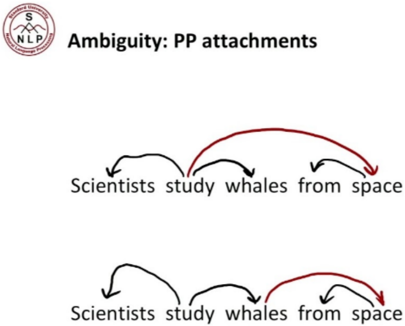


Figure 5: Prepositional phrase (PP) attachment. Image courtesy of Stanford NLP lab

Towards dependency-embeddings

Idea: apply dependency parsing first

*I would like to **drink** a very hot tall decaf half-soy (...) white chocolate **mocha***

Output of Stanford dependency parser:

```
nsubj(like-3, I-1)   nsubj(drink-5, I-1)   aux(like-3, would-2)
root(ROOT-0, like-3) mark(drink-5, to-4)   xcomp(like-3, drink-5)
det(mocha-14, a-6)   advmod(hot-8, very-7) amod(mocha-14, hot-8)
amod(mocha-14, tall-9) amod(mocha-14, decaf-10) amod(mocha-14, half-soy-11)
amod(mocha-14, white-12) compound(mocha-14, chocolate-13)
dobj(drink-5, mocha-14)
```

`dobj(drink-5, mocha-14)`: direct object

The direct object of a verb phrase is the noun phrase which is the (accusative) object of the verb.

Dependency-based embeddings

I would like to ***drink*** *a very hot tall decaf half-soy (...) white chocolate* ***mocha***

```
nsubj(like-3, I-1)   nsubj(drink-5, I-1)   aux(like-3, would-2)
root(ROOT-0, like-3) mark(drink-5, to-4)   xcomp(like-3, drink-5)
...
```

O. Levy and Y. Goldberg (2014). "Dependency-Based Word Embeddings". In: *Proceedings of ACL*. Baltimore, MD, USA, pp. 302–308

Word	Dependency Context
like	I/nsubj, would/aux, drink/xcomp
drink	I/nsubj, to/mark, mocha/dobj, like/xcomp-1
hot	very/advmod, mocha/amod-1
...	...

Dependency-based embeddings

`Word2Vec` finds words that **associate with** other words, while DepEmbeddings finds words **behave like** others – Domain similarity vs. functional similarity

Target Word	BoW5	BoW2	Deps
batman	nightwing aquaman catwoman superman manhunter	superman superboy aquaman catwoman batgirl	superman superboy supergirl catwoman aquaman
hogwarts	dumbledore hallows half-blood malfoy snape	evernight sunnydale garderobe blandings collinwood	sunnydale collinwood calarts greendale millfield
turing	nondeterministic non-deterministic computability deterministic finite-state	non-deterministic finite-state nondeterministic buchi primality	pauling hotelling heting lessing hamming
florida	gainesville fla jacksonville tampa lauderdale	fla alabama gainesville tallahassee texas	texas louisiana georgia california carolina
object-oriented	aspect-oriented smalltalk event-driven prolog domain-specific	aspect-oriented event-driven objective-c dataflow 4gl	event-driven domain-specific rule-based data-driven human-centered
dancing	singing dance dances dancers tap-dancing	singing dance dances breakdancing clowning	singing rapping breakdancing miming busking

Miscellaneous

Embeddings of other things than words

Embed other stuff than words:

- **Characters**: *i n s i g h t f u l*
- **Syllables**: *in + sight + ful*
- **Morphemes**:
  - *insightful = insight + ful*
  - *helping = help + ing*
  - *greedily = greedy + ly*
  - *Dampfschifffahrt = Dampf+Schiff+Fahrt*
  - Useful particulary for morphologically rich languages like German, French, Czech, etc.
  - Rarely find Dampfschifffahrt in a corpus, but its three morphemes are quite likely
- Embed **postags**, **synsets**, **lexemes**, **supersenses**, ...

Embeddings of other things than words

Embed `n-grams` – the `FastText` approach[4]

Words are represented as bags of character n-grams (n=3,4,5,6)

n=3: *where* = (*>wh* , *whe, her, ere* , *re<*)

Learn embeddings for all n-grams, represent a word by averaging over its n-gram embeddings

Big advantage:

– Can embed OOV words, spelling mistakes: "lenght", "spellling"

– Naturally works for morphologically rich languages

[4] P. Bojanowski et al. (2017). "Enriching Word Vectors with Subword Information". In: *Transactions of the ACL* 5, pp. 135–146

Using word embeddings in a task

Training word vectors to the task

Option 1: fixed word representations

- map word into id and get the vector from the embedding matrix
- only train the weights of the hidden layers

Option 2: adjust the word representations to the task

- word vectors are parameters and are updated in each epoch
- Example: sentiment classification, train vectors to represent positive/negative polarity for each word

Problem: Adaptation to the training data

Representations for words that are seen in the training data move in vector space, but words that are not seen remain where they were

- "TV", "telly" and "television" all indicate negative sentiment in the dataset
- Due to pre-training, they have similar vectors
- "TV" and "telly" occur in the training data, "television" in the test data

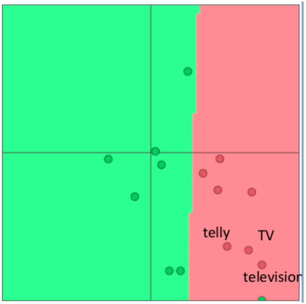


Figure 6: Courtesy of Richard Socher

Problem: Adaptation to the training data

"TV" and "telly" have been updated

"television" stayed the same -> synonym information is lost

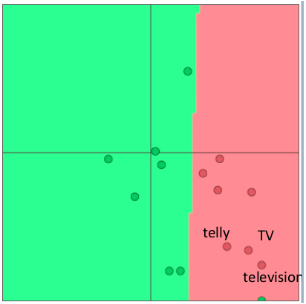


Figure 7: Before training

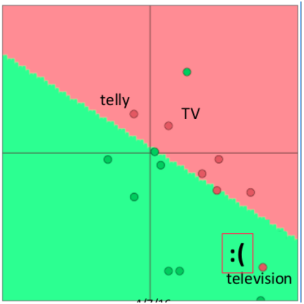


Figure 8: After training

Practical tips

Only train word vectors to the task if you have a large training corpus.

Even then, it might not be useful (depends on the task).

Common practice:

– Train the vectors only for a few epochs and then keep them fixed

If in doubt
Keep your embeddings fixed

Summary

Summary: Embedding approaches

What do all the embedding approaches have in common?

– Represent natural language input with real-valued vectors

Differences

Unit of representation
characters, morphemes, words, senses, phrases, windows, sentences, documents, ...

Definition of context for training
CBOW, Skip-gram, Glove, positional, dependency-based, ...

Towards contextualized embeddings

Static word embeddings – huge impact on adoption of DL in NLP

Becoming extinct now

Deplaced by contextualized embeddings (BERT, etc.)

License and credits

Credits

Ivan Habernal, Steffen Eger

Akir (https://commons.wikimedia.org/wiki/File:Table.png, CC-BY-SA)

Baba66 (https://commons.wikimedia.org/wiki/File:Arabicalphabet.svg, CC-BY)

Content from ACL Anthology papers licensed under CC-BY https://www.aclweb.org/anthology

References

- Artetxe, M., G. Labaka, and E. Agirre (2017). “Learning bilingual word embeddings with (almost) no bilingual data”. In: *Proceedings of ACL*. Vancouver, Canada, pp. 451–462.
- Bojanowski, P., E. Grave, A. Joulin, and T. Mikolov (2017). “Enriching Word Vectors with Subword Information”. In: *Transactions of the ACL* 5, pp. 135–146.

References ii

Glavaš, G., R. Litschko, S. Ruder, and I. Vulić (2019). "How to (Properly) Evaluate Cross-Lingual Word Embeddings: On Strong Baselines, Comparative Analyses, and Some Misconceptions". In: *Proceedings of ACL*. Florence, Italy, pp. 710–721.

Iacobacci, I., M. T. Pilehvar, and R. Navigli (2015). "SensEmbed: Learning Sense Embeddings for Word and Relational Similarity". In: *Proceedings of ACL*. Beijing, China: Association for Computational Linguistics, pp. 95–105.

Levy, O. and Y. Goldberg (2014). “Dependency-Based Word Embeddings”. In: *Proceedings of ACL*. Baltimore, MD, USA, pp. 302–308.

Ling, W., C. Dyer, A. W. Black, and I. Trancoso (2015). “Two/Too Simple Adaptations of Word2Vec for Syntax Problems”. In: *Proceedings of NAACL*. Denver, Colorado, pp. 1299–1304.

Upadhyay, S., M. Faruqui, C. Dyer, and D. Roth (2016). “Cross-lingual Models of Word Embeddings: An Empirical Comparison”. In: *Proceedings of ACL*. Berlin, Germany, pp. 1661–1670.

References iv

Vulić, I. and M.-F. Moens (2015). “Bilingual Word Embeddings from Non-Parallel Document-Aligned Data Applied to Bilingual Lexicon Induction”. In: *Proceedings of ACL (Volume 2: Short Papers)*. Beijing, China, pp. 719–725.