# Below and beyond the word level: Subword embeddings and embeddings for phrases, sentences, documents...

VL Embeddings

Uni Heidelberg

SS 2019

# Below and beyond words

- We can learn semantic representations for words
- But what about other linguistic units?
    - characters
    - morphemes
    - phrases
    - sentences
    - paragraphs
    - documents

# Subword embeddings

- Motivation:
  High-quality representations for rare or unknown words for
  - morphologically rich languages
  - low-resourced languages
  - languages with no clear word boundaries
  - noisy text (learner language, user-generated content)
  - text from new domains (with many unknown words)

# Subword embeddings

- Motivation:
  High-quality representations for rare or unknown words for
  - morphologically rich languages
  - low-resourced languages
  - languages with no clear word boundaries
  - noisy text (learner language, user-generated content)
  - text from new domains (with many unknown words)
- FastText:
  - word embeddings enriched with subword information

# Subword embeddings

- Motivation:
  High-quality representations for rare or unknown words for
  - morphologically rich languages
  - low-resourced languages
  - languages with no clear word boundaries
  - noisy text (learner language, user-generated content)
  - text from new domains (with many unknown words)
- FastText:
  - word embeddings enriched with subword information

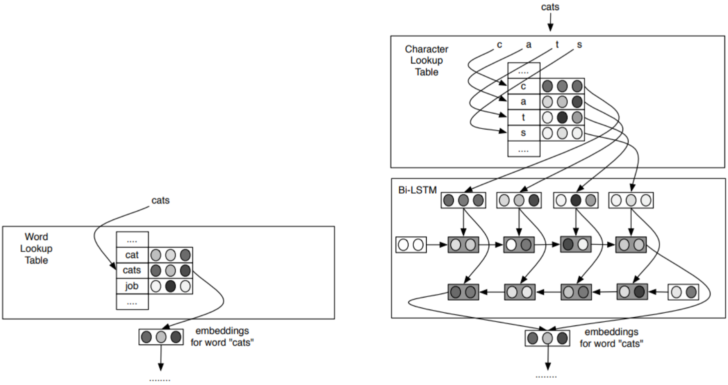
Why not training representations for subword units directly?

# Subword embedding types

- Character-based embeddings (characters or char-ngrams)
  - Ling et al. 2015; Luong and Manning 2016; Chiu and Nichols 2016
- Phonemes and Graphemes
  - Chaudhary et al. 2018
- Morphemes
  - Luong et al., 2013; Botha and Blunsom, 2014; Cotterell and Schütze, 2015; Chaudhary et al. 2018
- Byte-pair encoding
  - Sennrich et al. 2016; Heinzerling and Strube 2018
- Compound embeddings
  - Do et al. 2017

# Character-based embeddings

- Based on
  - recurrent neural networks (RNN) (Ling et al. 2015)
  - convolutional neural networks (CNN) (Chiu and Nichols, 2016)




from Ling et al. (2015)

# Character-based embeddings

- Often used in combination with word embeddings, e.g. for
    - POS/NER tagging (e.g. dos Santos and Zadrozny 2014; dos Santos et al. 2015; Ma and Hovy 2016; Lample et al. 2016)
    - dependency parsing (e.g. Ma et al. 2018)
    - text normalisation (Watson et al. 2018)
    - ...

# Byte-pair encoding (BPE)

- Variable-length encoding: text as a sequence of symbols
  - iteratively merge most frequent symbol pair into a new symbol
    e.g.: 1. iteration: `t h` $\rightarrow$ `th`
    2. iteration: `th e` $\rightarrow$ `the`

# Byte-pair encoding (BPE)

- Variable-length encoding: text as a sequence of symbols
  - iteratively merge most frequent symbol pair into a new symbol
    e.g.: 1. iteration: `t h` → `th`
    2. iteration: `th e` → `the`

aaabdaaabac

# Byte-pair encoding (BPE)

- Variable-length encoding: text as a sequence of symbols
  - iteratively merge most frequent symbol pair into a new symbol

    e.g.: 1. iteration: `t h` $\rightarrow$ `th`

    2. iteration: `th e` $\rightarrow$ `the`

aaabdaaabac Z=aa

# Byte-pair encoding (BPE)

- Variable-length encoding: text as a sequence of symbols
  - iteratively merge most frequent symbol pair into a new symbol
    e.g.: 1. iteration: `t h` $\rightarrow$ `th`
    2. iteration: `th e` $\rightarrow$ `the`

aaabdaaabac Z=aa
ZabdZabac

# Byte-pair encoding (BPE)

- Variable-length encoding: text as a sequence of symbols
  - iteratively merge most frequent symbol pair into a new symbol

    e.g.: 1. iteration: `t h` → `th`

    2. iteration: `th e` → `the`

| | |
|---|---|
| aaabdaaabac | Z=aa |
| ZabdZabac | Y=ab |

# Byte-pair encoding (BPE)

- Variable-length encoding: text as a sequence of symbols
  - iteratively merge most frequent symbol pair into a new symbol

    e.g.: 1. iteration: `t h` $\rightarrow$ `th`

    2. iteration: `th e` $\rightarrow$ `the`

| | |
|---|---|
| aaabdaaabac | Z=aa |
| ZabdZabac | Y=ab |
| ZYdZYac | |

# Byte-pair encoding (BPE)

- Variable-length encoding: text as a sequence of symbols
  - iteratively merge most frequent symbol pair into a new symbol
    e.g.: 1. iteration: `t h` $\rightarrow$ `th`
    2. iteration: `th e` $\rightarrow$ `the`

| | |
|---|---|
| aaabdaaabac | Z=aa |
| ZabdZabac | Y=ab |
| ZYdZYac | X=ZY |

# Byte-pair encoding (BPE)

- Variable-length encoding: text as a sequence of symbols
  - iteratively merge most frequent symbol pair into a new symbol
    e.g.: 1. iteration: `t h` → `th`
    2. iteration: `th e` → `the`

| | |
|---|---|
| aaabdaaabac | Z=aa |
| ZabdZabac | Y=ab |
| ZYdZYac | X=ZY |
| XdXac | |

Example from:
https://howlingpixel.com/i-en/Byte_pair_encoding

# Byte-pair encoding (BPE)

- Variable-length encoding: text as a sequence of symbols
  - iteratively merge most frequent symbol pair into a new symbol
    e.g.: 1. iteration: `t h` → `th`
    2. iteration: `th e` → `the`

| | |
|---|---|
| aaabdaaabac | Z=aa |
| ZabdZabac | Y=ab |
| ZYdZYac | X=ZY |
| XdXac | |

Example from: https://howlingpixel.com/i-en/Byte_pair_encoding

  - Parameter $o$: number of merge operations
  - $o$ determines if resulting encoding mostly creates short character sequences (e.g. $o = 1000$) or if it includes symbols for many frequently occurring words, e.g. $o = 30,000$

# Byte-pair encoding (BPE)

Heinzerling and Strube (2018): Collection of pre-trained subword embeddings in 275 languages

- `https://github.com/bheinzerling/bpemb`

  - Based on Byte-Pair Encoding (BPE)
  - Trained on Wikipedia:
    1. iterate over Wikipedia to create byte-pairs
    2. pretrain embeddings for resulting BPE symbol using GloVe

# Byte-pair encoding (BPE)

Heinzerling and Strube (2018): Collection of pre-trained subword embeddings in 275 languages

- https://github.com/bheinzerling/bpemb
  - Based on Byte-Pair Encoding (BPE)
  - Trained on Wikipedia:
    1. iterate over Wikipedia to create byte-pairs
    2. pretrain embeddings for resulting BPE symbol using GloVe
  - Advantages of BPE:
    - competetive performance to other types of embeddings for entity typing
    - more compact representations
    - no tokenisation required

# Beyond word embeddings: phrase vectors

Mikolov et al. (2013c): Distributed representations of words and phrases and their compositionality

**New York Times** ⇒ newspaper
(not combination of new and york and times)

# Beyond word embeddings: phrase vectors

Mikolov et al. (2013c): Distributed representations of words and phrases and their compositionality

**New York Times** ⇒ newspaper
(not combination of new and york and times)

- **Goal**: Learn vectors that represent phrases instead of words

# Beyond word embeddings: phrase vectors

Mikolov et al. (2013c): Distributed representations of words and phrases and their compositionality

**New York Times** ⇒ newspaper
(not combination of new and york and times)

- **Goal**: Learn vectors that represent phrases instead of words
- **Approach**:
  1. find words that occur frequently together, and infrequently in other context
  2. merge those into an atomic representation, e.g.:

     New York Times ⇒ New_York_Times

  3. train word vectors on the modified corpus
     where phrases are now new atomic words

# Phrase Vectors

## Evaluation

- Analogical reasoning task: `https://code.google.com/archive/p/word2vec/source/default/source/source-archive.zip` (file: questions-phrases.txt)

- Test set with both words and phrases
  Steve_Jobs : Apple :: Bill_Gates : ?
  - correct if nearest representation to
    vec("Apple") - vec("Steve_Jobs") + vec("Bill_Gates")
    is vec("?")
  - 5 different categories of analogies

# Phrase Vectors

## Evaluation

- Analogical reasoning task:
  `https://code.google.com/archive/p/word2vec/source/default/source/source-archive.zip` (file: questions-phrases.txt)

- Test set with both words and phrases
  Steve_Jobs : Apple :: Bill_Gates : Microsoft
  - correct if nearest representation to
    vec("Apple") - vec("Steve_Jobs") + vec("Bill_Gates")
    is vec("Microsoft")
  - 5 different categories of analogies

# Phrase Vectors

## Evaluation

- Analogical reasoning task: `https://code.google.com/archive/p/word2vec/source/default/source/source-archive.zip` (file: questions-phrases.txt)

| | | | |
|---|---|---|---|
| Newspapers | | | |
| New York<br>San Jose | New York Times<br>San Jose Mercury News | Baltimore<br>Cincinnati | Baltimore Sun<br>Cincinnati Enquirer |
| NHL Teams | | | |
| Boston<br>Phoenix | Boston Bruins<br>Phoenix Coyotes | Montreal<br>Nashville | Montreal Canadiens<br>Nashville Predators |
| NBA Teams | | | |
| Detroit<br>Oakland | Detroit Pistons<br>Golden State Warriors | Toronto<br>Memphis | Toronto Raptors<br>Memphis Grizzlies |
| Airlines | | | |
| Austria<br>Belgium | Austrian Airlines<br>Brussels Airlines | Spain<br>Greece | Spainair<br>Aegean Airlines |
| Company executives | | | |
| Steve Ballmer<br>Samuel J. Palmisano | Microsoft<br>IBM | Larry Page<br>Werner Vogels | Google<br>Amazon |

# Phrase Vectors

## Evaluation

- Train different SkipGram models with dimensions = 300 and context size=5 on news data
    - Hierarchical Softmax versus Negative Sampling
    - with/without subsampling of frequent tokens

| Method | Dimensionality | no subsampling [%] | $10^{-5}$ subsampling [%] |
|---|---|---|---|
| NEG-5 | 300 | 24 | 27 |
| NEG-15 | 300 | 27 | 42 |
| HS-Huffman | 300 | 19 | **47** |

Table : Accuracies of SkipGram models on phrase analogy dataset.

# Phrase Vectors

## Evaluation

- Train different SkipGram models with dimensions = 300 and context size=5 on news data
  - Hierarchical Softmax versus Negative Sampling
  - with/without subsampling of frequent tokens
- Maximise accuracy by increasing amount of training data
  
  $\Rightarrow$ dataset with about 33 billion words
  - Hierarchical Softmax, dimension = 1000, context size = entire sentence
- increased accuracy of **72%**

# Phrase Vectors

## Evaluation

- Train different SkipGram models with dimensions = 300 and context size=5 on news data
  - Hierarchical Softmax versus Negative Sampling
  - with/without subsampling of frequent tokens

- Maximise accuracy by increasing amount of training data

  ⇒ dataset with about 33 billion words
  - Hierarchical Softmax, dimension = 1000, context size = entire sentence
- increased accuracy of **72%**

Best model for analogy task: hierarchical softmax and subsampling of frequent words

# Additive compositionality

- Word and phrase representations exhibit a linear structure that makes it possible to perform analogical reasoning using simple vector arithmetics

  vec(Berlin) - vec(Germany) + vec(France) = Paris

# Additive compositionality

- Word and phrase representations exhibit a linear structure that makes it possible to perform analogical reasoning using simple vector arithmetics

  vec(Berlin) - vec(Germany) + vec(France) = Paris

- Word vectors also show additive compositionality:
  - combine words by an element-wise addition of their vector representations, e.g.:

# Additive compositionality

- Word and phrase representations exhibit a linear structure that makes it possible to perform analogical reasoning using simple vector arithmetics

  vec(Berlin) - vec(Germany) + vec(France) = Paris

- Word vectors also show additive compositionality:
  - combine words by an element-wise addition of their vector representations, e.g.:

    vec(Vietnam) + vec(capital) = Hanoi

# Additive compositionality

- Word and phrase representations exhibit a linear structure that makes it possible to perform analogical reasoning using simple vector arithmetics

  vec(Berlin) - vec(Germany) + vec(France) = Paris

- Word vectors also show additive compositionality:
  - combine words by an element-wise addition of their vector representations, e.g.:

    vec(German) + vec(airlines) = Lufthansa

# Additive compositionality

- Word and phrase representations exhibit a linear structure that makes it possible to perform analogical reasoning using simple vector arithmetics

  vec(Berlin) - vec(Germany) + vec(France) = Paris

- Word vectors also show additive compositionality:
  - combine words by an element-wise addition of their vector representations, e.g.:

    vec(French) + vec(actress) = Juliette_Binoche

# Beyond Words: Sentence and Document Representations

Le and Mikolov (2014): Distributed Representations of Sentences and Documents

- Paragraph Vector
  - learns fixed-length feature representations from variable-length pieces of texts (sentences, paragraphs, documents)

# Beyond Words: Sentence and Document Representations

## Motivation

- Standard features for many **text classification** tasks: BoW
  - text is represented by fixed-length vectors of bag-of-words or bag-of-ngrams
  - simple, efficient, hard-to-beat baseline

# Beyond Words: Sentence and Document Representations

## Motivation

- Standard features for many **text classification** tasks: BoW
  - text is represented by fixed-length vectors of bag-of-words or bag-of-ngrams
  - simple, efficient, hard-to-beat baseline
- Disadvantages
  - word order is lost (or only preserved for short contexts) $\rightarrow$ semantically different sentences can have the same (or very similar) representations:

    When Mary started singing, everybody went home.

    When everybody went home, Mary started singing.

# Beyond Words: Sentence and Document Representations

## Motivation

- How can we get meaningful representations for sequences of words?
- Two very simple approaches:
  1. Phrase vectors (Mikolov et al. 2013c)
     $\Rightarrow$ merge word collocations into a new, atomic string and train embeddings for that new "word"

  2. Combine word vectors by **concatenating** them or by taking the **average** of two vectors, then use resulting vector to predict other words in the context
     (Bengio et al., 2006; Collobert & Weston, 2008; Mnih & Hinton, 2008; Turian et al., 2010; Mikolov et al., 2013a,b)

# Beyond Words

## Le and Mikolov (2014): Distributed Representations of Sentences and Documents

- Learn representations for whole sentences, paragraphs, documents... ⇒ vector representation is trained to predict words in a paragraph
  1. concatenate paragraph vector with several word vectors from the paragraph
  2. predict the following word in the given context
  3. train both, word and paragraph vectors, using stochastic gradient descent and backpropagation (Rumelhart et al., 1986)
- Paragraph vectors are unique among paragraphs
- Word vectors are shared across all paragraphs

# Beyond Words

## Le and Mikolov (2014)

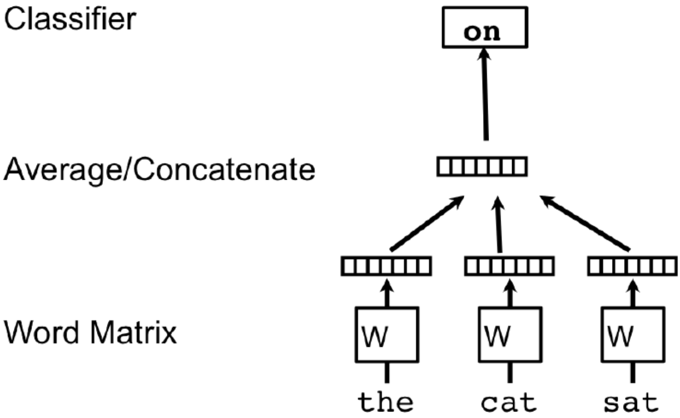
### Intuition

- Word vectors:
  - contribute to predicting words in sentence context
- Paragraph vectors:
  - contribute to predicting words sampled from whole paragraph

# Beyond Words

Le and Mikolov (2014)

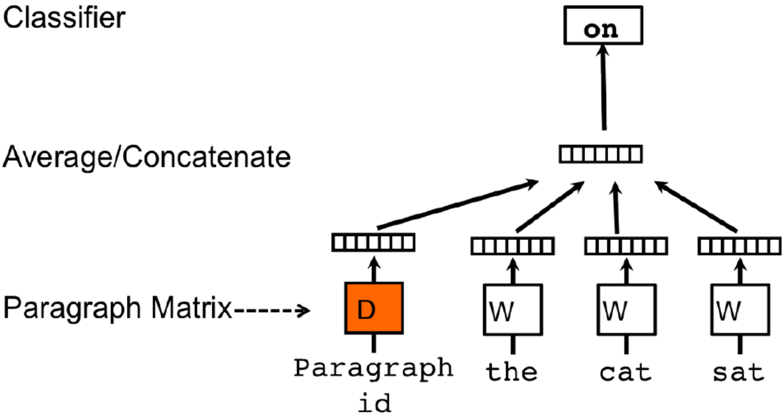
## Word vector model



# Beyond Words

Le and Mikolov (2014)

## Paragraph vector model



# Beyond Words

## Le and Mikolov (2014)

- Technical details:
  - Sample fixed-lenght contexts from a sliding window over the paragraph
  - Paragraph vector is shared across all contexts generated from the same paragraph
  - Word vector matrix is shared across paragraphs

# Beyond Words

## Le and Mikolov (2014)

- Technical details:
  - Sample fixed-lenght contexts from a sliding window over the paragraph
  - Paragraph vector is shared across all contexts generated from the same paragraph
  - Word vector matrix is shared across paragraphs
- Training with SGD and backpropagation
- In every iteration
  1. sample a fixed-length context from a random paragraph,
  2. compute the error gradient from the network
  3. use gradient to update parameters of the model

# Beyond Words

Le and Mikolov (2014)

- Advantages of the paragraph vectors
  - inherit properties of word vectors
  - sensitive to word order (at least in a small context)
  - less sparse than bag-of-ngram models
- Extension of the model: Distributed bag of words version of Paragraph Vector (PV-DBOW)
  - similar to SkipGram (not shown here, see paper)

# References


- Maas, Andrew L., Daly, Raymond E., Pham, Peter T., Huang, Dan, Ng, Andrew Y., and Potts, Christopher. Learning word vectors for sentiment analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, 2011.
- Socher, Richard, Perelygin, Alex, Wu, Jean Y., Chuang, Jason, Manning, Christopher D., Ng, Andrew Y., and Potts, Christopher. Recursive deep models for semantic compositionality over a sentiment treebank. In Conference on Empirical Methods in Natural Language Processing, 2013.
- Quoc Le and Tomas Mikolov (2014): Distributed Representations of Sentences and Documents. Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.
- Matteo Pagliardini, Prakhar Gupta, Martin Jaggi (2018): Unsupervised Learning of Sentence Embeddings Using Compositional n-Gram Features. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. New Orleans, Louisiana.
- Ledell Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes and Jason Weston (2018): StarSpace: Embed All The Things! Conference on Artificial Intelligence (AAAI).
- Bengio, Yoshua, Schwenk, Holger, Senecal, Jean-Sebastien, Morin, Frederic, and Gauvain, Jean-Luc (2006): Neural probabilistic language models. In Innovations in Machine Learning, pp. 137–186. Springer, 2006.
- Collobert, Ronan and Weston, Jason (2008): A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th International Conference on Machine Learning, pp. 160–167.
- Mnih, Andriy and Hinton, Geoffrey E. A scalable hierarchical distributed language model (2008): In Advances in Neural Information Processing Systems, pp. 1081–1088.
- Turian, Joseph, Ratinov, Lev, and Bengio, Yoshua (2010): Word representations: a simple and general method for semi-supervised learning. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 384–394. Association for Computational Linguistics.
- Mikolov, Tomas, Chen, Kai, Corrado, Greg, and Dean, Jeffrey (2013a): Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

# References


- Mikolov, Tomas, Le, Quoc V., and Sutskever, Ilya (2013b): Exploiting similarities among languages for machine translation. CoRR, abs/1309.4168.
- Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg, and Dean, Jeffrey (2013c): Distributed representations of phrases and their compositionality. In Advances on Neural Information Processing Systems 2013, pages 3111–3119.
- Rumelhart, David E, Hinton, Geoffrey E, and Williams, Ronald J. (1986): Learning representations by back-propagating errors. Nature, 323(6088):533–536.
- Marco Baroni and Alessandro Lenci (2010): Distributional memory: A general framework for corpus-based semantics. Computational Linguistics, 36(4):673–721.
- Yoav Goldberg and Omer Levy (2014): word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. arXiv preprint arXiv:1402.3722.
- Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In Proceedings of the 36th Annual Meeting of the Association for Computational Lin- guistics and 17th International Conference on Computational Linguistics - Volume 2, ACL '98, pages 768–774, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean (2013): Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States, pages 3111–3119.
- Sebastian Padó and Mirella Lapata (2007): Dependency-based construction of semantic space models. Computational Linguistics, 33(2):161–199.
- Omer Levy and Yoav Goldberg (2014): Dependency-based word embeddings. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 302–308, Baltimore, Maryland, USA
- Benjamin Heinzerling and Michael Strube (2018): BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. The 11th International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan.

# References


- Sennrich, R., Haddow, B., and Birch, A. (2016): Neural machine translation of rare words with subword units. The 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016), pages 1715–1725, Berlin, Germany.
- Luong, M.-T. and Manning, C. D. (2016): Achieving open vocabulary neural machine translation with hybrid word-character models. The 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016), pages 1054–1063. Berlin, Germany.
- Bich-Ngoc Do, Ines Rehbein and Anette Frank (2017): What do we need to know about an unknown word when parsing German. 1st Workshop on Subword and Character Level Models in NLP. Copenhagen, Denmark.
- Cicero dos Santos and Bianca Zadrozny (2014): Learning character-level representations for part-ofspeech tagging. The 31st International Conference on Machine Learning. pages 1818–1826.
- Cicero dos Santos, Victor Guimaraes, RJ Niteroi, and Rio de Janeiro. 2015. Boosting named entity recognition with neural character embeddings. The 5th Named Entities Workshop.
- Xuezhe Ma and Eduard Hovy (2016): End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. The 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016).
- Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. (2016): Neural architectures for named entity recognition. NAACL-HLT 2016, pages 260–270, San Diego, California.
- Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig and Eduard Hovy (2018): Stack-Pointer Networks for Dependency Parsing. ACL 2018.
- Daniel Watson, Nasser Zalmout and Nizar Habash (2018): Utilizing Character and Word Embeddings for Text Normalization with Sequence-to-Sequence Models. Empirical Methods in Natural Language Processing, pages 837–843. Brussels, Belgium.
- Aditi Chaudhary, Chunting Zhou, Lori Levin, Graham Neubig, David R. Mortensen, Jaime G. Carbonell (2018): Adapting Word Embeddings to New Languages with Morphological and Phonological Subword Representations. EMNLP 2018. Brussles, Belgium.
- Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis (2015): Finding function in form: Compositional character models for open vocabulary word representation. Empirical Methods in Natural Language Processing. Lisbon, Portugal, pages 1520–1530.