Deep Learning in NLP

Horacio Rodríguez

Outline

- Introduction
- Short review of Distributional Semantics, Semantic spaces, VSM, …
- Embeddings
  - Embedding of words
  - Embedding of more complex units
- Simple Linear Models
- Neural Networks models for NLP
- Applications
- Conclusions

Applications

from Young et al, 2018

TABLE II: **POS tagging**

Paper	Model	WSJ-PTB (per-token accuracy %)
Giménez and Marquez [132]	SVM with manual feature pattern	97.16
Collobert et al. [5]	MLP with word embeddings + CRF	97.29
Santos and Zadrozny [32]	MLP with character+word embeddings	97.32
Huang et al. [133]	LSTM	97.29
Huang et al. [133]	Bidirectional LSTM	97.40
Huang et al. [133]	LSTM-CRF	97.54
Huang et al. [133]	Bidirectional LSTM-CRF	97.55
Andor et al. [134]	Transition-based neural network	97.45
Kumar et al. [97]	DMN	97.56

Applications

from Young et al, 2018

TABLE III: **Parsing** (UAS/LAS = Unlabeled/labeled Attachment Score; WSJ = The Wall Street Journal Section of Penn Treebank)

Parsing type	Paper	Model	WSJ
Dependency Parsing	Chen and Manning [135]	Fully-connected NN with features including POS	91.8/89.6 (UAS/LAS)
	Weiss et al. [136]	Deep fully-connected NN with features including POS	94.3/92.4 (UAS/LAS)
	Dyer et al. [137]	Stack-LSTM	93.1/90.9 (UAS/LAS)
	Zhou et al. [138]	Beam contrastive model	93.31/92.37 (UAS/LAS)
Constituency Parsing	Petrov et al. [139]	Probabilistic context-free grammars (PCFG)	91.8 (F1 Score)
	Socher et al. [10]	Recursive neural networks	90.29 (F1 Score)
	Zhu et al. [140]	Feature-based transition parsing	91.3 (F1 Score)
	Vinyals et al. [101]	seq2seq learning with LSTM+Attention	93.5 (F1 Score)

Applications

from Young et al, 2018

TABLE IV: Named-Entity Recognition

Paper	Model	CoNLL 2003 (F1 %)
Collobert et al. [5]	MLP with word embeddings+gazetteer	89.59
Passos et al. [142]	Lexicon Infused Phrase Embeddings	90.90
Chiu and Nichols [143]	Bi-LSTM with word+char+lexicon embeddings	90.77
Luo et al. [144]	Semi-CRF jointly trained with linking	91.20
Lample et al. [88]	Bi-LSTM-CRF with word+char embeddings	90.94
Lample et al. [88]	Bi-LSTM with word+char embeddings	89.15
Strubell et al. [145]	Dilated CNN with CRF	90.54

Applications

from Young et al, 2018

TABLE V: **Semantic Role Labeling**

Paper	Model	CoNLL2005 (F1 %)	CoNLL2012 (F1 %)
Collobert et al. [5]	CNN with parsing features	76.06	
Täckström et al. [146]	Manual features with DP for inference	78.6	79.4
Zhou and Xu [147]	Bidirectional LSTM	81.07	81.27
He et al. [148]	Bidirectional LSTM with highway connections	83.2	83.4

Applications

from Young et al, 2018

TABLE VI: **Sentiment Classification** (SST-1 = Stanford Sentiment Treebank, fine-grained 5 classes Socher et al. [4]; SST-2: the binary version of SST-1; Numbers are accuracies (%))

Paper	Model	SST-1	SST-2
Socher et al. [4]	Recursive Neural Tensor Network	45.7	85.4
Kim [45]	Multichannel CNN	47.4	88.1
Kalchbrenner et al. [44]	DCNN with k-max pooling	48.5	86.8
Tai et al. [111]	Bidirectional LSTM	48.5	87.2
Le and Mikolov [149]	Paragraph Vector	48.7	87.8
Tai et al. [111]	Constituency Tree-LSTM	51.0	88.0
Yu et al. [150]	Tree-LSTM with refined word embeddings	54.0	90.3
Kumar et al. [97]	DMN	52.1	88.6

Applications

from Young et al, 2018

TABLE VII: **Machine translation** (Numbers are BLEU scores)

Paper	Model	WMT2014 English2German	WMT2014 English2French
Cho et al. [77]	Phrase table with neural features		34.50
Sutskever et al. [69]	Reranking phrase-based SMT best list with LSTM seq2seq		36.5
Wu et al. [151]	Residual LSTM seq2seq + Reinforcement learning refining	26.30	41.16
Gehring et al. [152]	seq2seq with CNN	26.36	41.29
Vaswani et al. [153]	Attention mechanism	28.4	41.0

Applications

from Young et al, 2018

TABLE VIII: **Question answering**

Paper	Model	bAbI (Mean accuracy %)	Farbes (Accuracy %)
Fader et al. [157]	Paraphrase-driven lexicon learning		0.54
Bordes et al. [158]	Weekly supervised embedding		0.73
Weston et al. [107]	Memory networks	93.3	0.83
Sukhbaatar et al. [131]	End-to-end memory networks	88.4	
Kumar et al. [97]	DMN	93.6	

Applications

from Young et al, 2018

TABLE IX: **Dialogue systems**

Paper	Model	Twitter Conversation Triple Dataset (BLEU)	Ubuntu Dialogue Dataset (recall 1@10 %)
Ritter et al. [159]	SMT	3.60	
Sordoni et al. [160]	SMT+neural reranking	4.44	
Li et al. [161]	LSTM seq2seq	4.51	
Li et al. [161]	LSTM seq2seq with MMI objective	5.22	
Lowe et al. [92]	Dual LSTM encoders for semantic matching		55.22
Dodge et al. [162]	Memory networks		63.72
Zhou et al. [163]	Sentence-level CNN-LSTM encoder		66.15

Applications

from from Wang et al, 2014
Different embedding models of relations and triples

Model	Score function $f_r(\mathbf{h}, \mathbf{t})$	# Parameters
TransE (Bordes et al. 2013b)	$\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_{\ell_{1/2}}, \mathbf{r} \in \mathbb{R}^k$	$O(n_e k + n_r k)$
Unstructured (Bordes et al. 2012)	$\|\mathbf{h} - \mathbf{t}\|_2^2$	$O(n_e k)$
Distant (Bordes et al. 2011)	$\|W_{rh}\mathbf{h} - W_{rt}\mathbf{t}\|_1, W_{rh}, W_{rt} \in \mathbb{R}^{k \times k}$	$O(n_e k + 2n_r k^2)$
Bilinear (Jenatton et al. 2012)	$\mathbf{h}^\top W_r \mathbf{t}, W_r \in \mathbb{R}^{k \times k}$	$O(n_e k + n_r k^2)$
Single Layer	$\mathbf{u}_r^\top f(W_{rh}\mathbf{h} + W_{rt}\mathbf{t} + \mathbf{b}_r)$ $\mathbf{u}_r, \mathbf{b}_r \in \mathbb{R}^s, W_{rh}, W_{rt} \in \mathbb{R}^{s \times k}$	$O(n_e k + n_r(sk + s))$
NTN (Socher et al. 2013)	$\mathbf{u}_r^\top f(\mathbf{h}^\top \mathbf{W}_r \mathbf{t} + W_{rh}\mathbf{h} + W_{rt}\mathbf{t} + \mathbf{b}_r)$ $\mathbf{u}_r, \mathbf{b}_r \in \mathbb{R}^s, \mathbf{W}_r \in \mathbb{R}^{k \times k \times s}, W_{rh}, W_{rt} \in \mathbb{R}^{s \times k}$	$O(n_e k + n_r(sk^2 + 2sk + 2s))$
TransH (this paper)	$\|(\mathbf{h} - \mathbf{w}_r^\top \mathbf{h}\mathbf{w}_r) + \mathbf{d}_r - (\mathbf{t} - \mathbf{w}_r^\top \mathbf{t}\mathbf{w}_r)\|_2^2$ $\mathbf{w}_r, \mathbf{d}_r \in \mathbb{R}^k$	$O(n_e k + 2n_r k)$

Applications

- Some applications with more details: QA
  - Bordes 2014's approach is based on converting questions to (uninterpretable) embeddings which require no pre-defined grammars or lexicons and can query any KB independent of its schema.
  - He focuses on answering simple factual questions on a broad range of topics, more specifically, those for which single KB triples stand for both the question and an answer.
    - automatically generating questions from KB triples and treating this as training data
    - Supplementing this with a data set of question collaboratively marked as paraphrases but with no associated answers.

Applications

Patterns for generating questions from ReVerb triples

KB Triple	Question Pattern
(?, r, e)	*who* r e *?*
(?, r, e)	*what* r e *?*
(e, r, ?)	*who does* e r *?*
(e, r, ?)	*what does* e r *?*
(?, r, e)	*what is the* r *of* e *?*
(?, r, e)	*who is the* r *of* e *?*
(e, r, ?)	*what is* r *by* e *?*
(?, r, e)	*who is* e*'s* r *?*

KB Triple	Question Pattern
(?, r, e)	*what is* e*'s* r *?*
(e, r, ?)	*who is* r *by* e *?*
(e, r-in, ?)	*when did* e r *?*
(e, r-on, ?)	*when did* e r *?*
(e, r-in, ?)	*when was* e r *?*
(e, r-on, ?)	*when was* e r *?*
(e, r-in, ?)	*where was* e r *?*
(e, r-in, ?)	*where did* e r *?*

Applications

- QA
  - Embedding Reverb
  - The model ends up learning embeddings of symbols, either for entities or relationships from ReVerb, or for each word of the vocabulary. The embeddings are used for scoring the similarities of a question q and a triple t, i.e. learning the function S(q,t).
  - It consists of projecting questions, treated as a bag of words (and possibly n-grams as well), on the one hand, and triples on the other hand, into a shared embedding space and then computing a similarity measure (as the dot product) between both projections.

Applications

- QA
  - Scoring function: $S(q, t) = f(q)^T g(t)$
    - $f(\cdot)$ a function mapping words from questions into R^k, $f(q) = V^T \Phi(q)$.
    - V is the matrix of $R^{nv \times k}$ containing all word embeddings v that will be learned,.
    - $\Phi(q)$ is the (sparse) binary representation of q ($\in \{0, 1\}^{nv}$) indicating absence or presence of words.
    - Similarly, $g(\cdot)$ is a function mapping entities and relationships from KB triples into R^k, $g(t) = W^T \Psi(t)$.

Applications

- QA
  - Scoring function: $S(q, t) = f(q)^T g(t)$
    - W is the matrix of $R^{ne \times k}$ containing all entity and relationship embeddings w, that will also be learned.
    - $\Psi(t)$ is the (sparse) binary representation of t ($\in \{0, 1\}^{ne}$) indicating absence or presence of entities and relationships.
    - An entity does not have the same embedding when appearing in the left-hand or in the right-hand side of a triple.

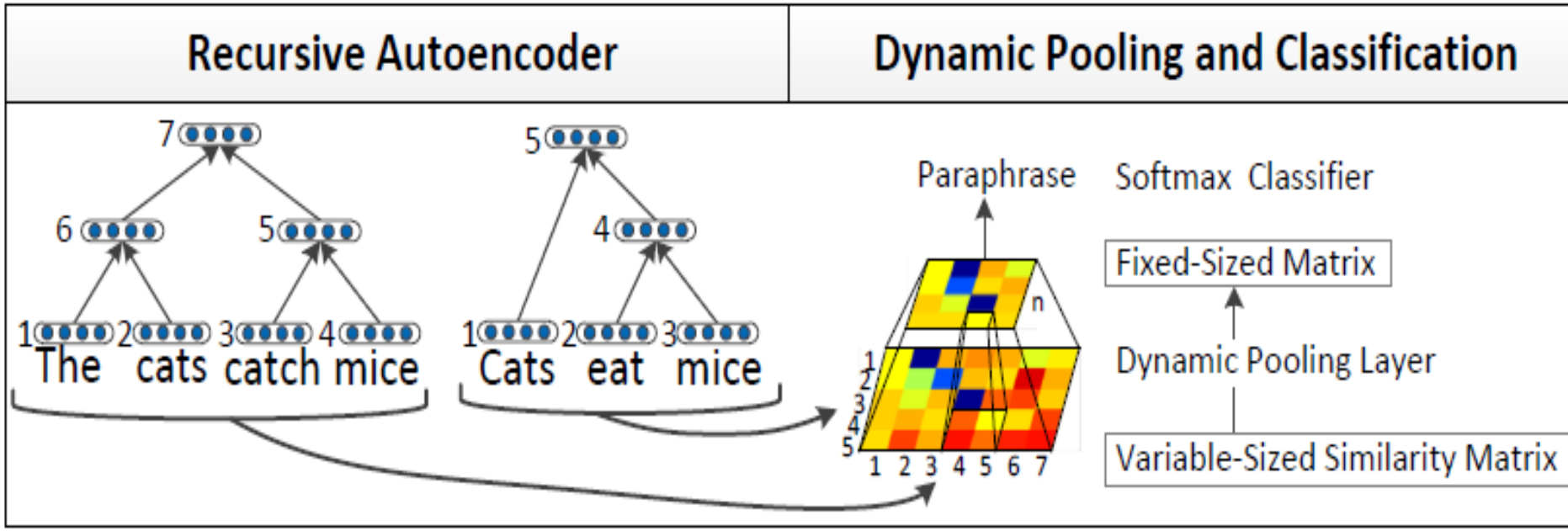
$$\hat{t}(q) = \arg\max_{t' \in \mathcal{K}} S(q, t') = \arg\max_{t' \in \mathcal{K}} \left(\boldsymbol{f}(q)^\top \boldsymbol{g}(t') \right).$$

Applications

- RAE for paraphrase detection
  - From Socher et al, 2011
  - RAE learns feature representations for each node in the tree such that the word vectors underneath each node can be recursively reconstructed.
  - These feature representations are used to compute a similarity matrix that compares both the single words as well as all nonterminal node features in both sentences.
  - In order to keep as much of the resulting global information of this comparison as possible and deal with the arbitrary length of the two sentences, a new dynamic pooling layer which outputs a fixed-size representation. Any classifier such as a softmax classifier can then be used to classify whether the two sentences are paraphrases or not.

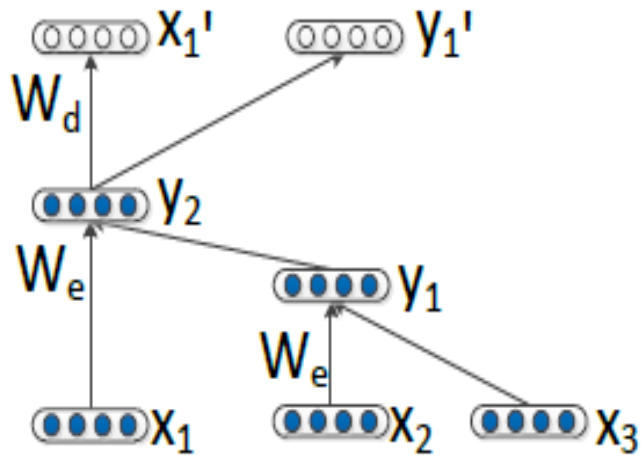
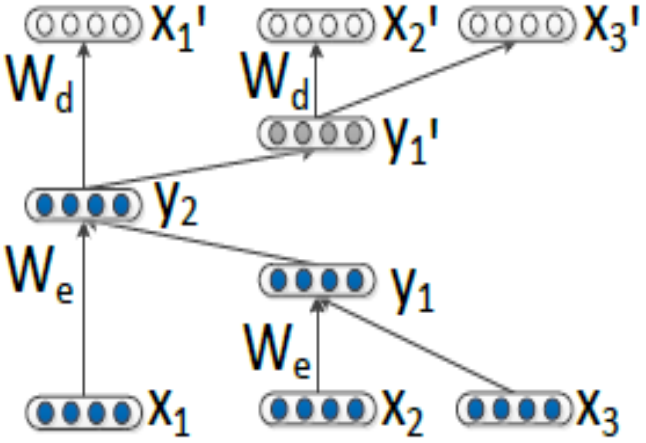
Applications

Paraphrase detection with RAE



Applications

Paraphrase detection with RAE

Recursive Autoencoder	Unfolding Recursive Autoencoder
	

Applications

Paraphrase detection with RAE

Representing a sentence as an ordered list of these vectors (x_1, …, x_m)
This word representation is better suited for RAEs than the binary number
representations used in previous related models.
A tree is given for each sentence by a parser.

Applications

Paraphrase detection with RAE

The binary parse tree for this input is in the form of branching triplets of parents with children: $(p \rightarrow c_1 c_2)$.
Each child can be either an input word vector x_i or a nonterminal node in the tree.
For both examples in last slide, we have the following triplets:
$((y_1 \rightarrow x_2 x_3), (y_2 \rightarrow x_1 y_1))$, $\forall x,\ y \in R^n$.

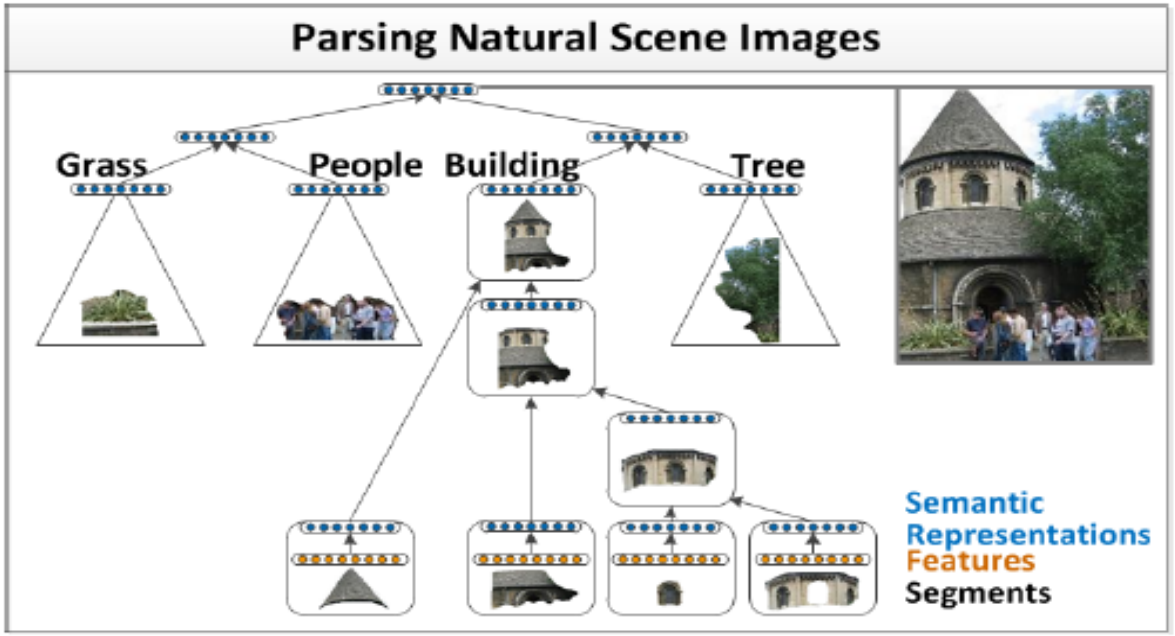
Applications

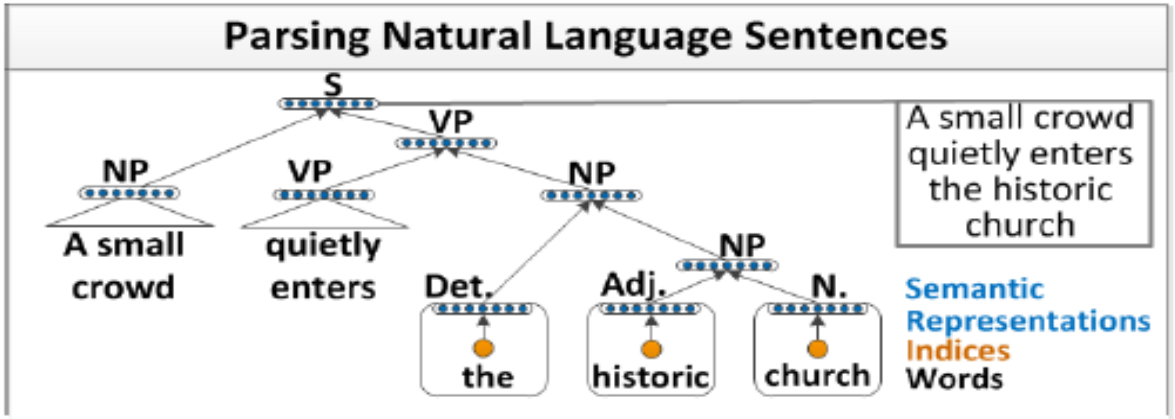
Paraphrase detection with RAE

Compute the parent representations.
$p = y_1$ is computed from the children $(c_1, c_2) = (x_2, x_3)$ by one standard neural network layer: $p = f(W_e[c_1; c_2] + b)$,
where $[c_1; c_2]$ is simply the concatenation of the two children,
f an element-wise activation function and $W_e \in R^{nX2n}$ (the encoding matrix).
how well this n-dimensional vector represents its direct children?
decode their vectors in a reconstruction layer and then compute the Euclidean distance between the original input and its reconstruction.

Applications

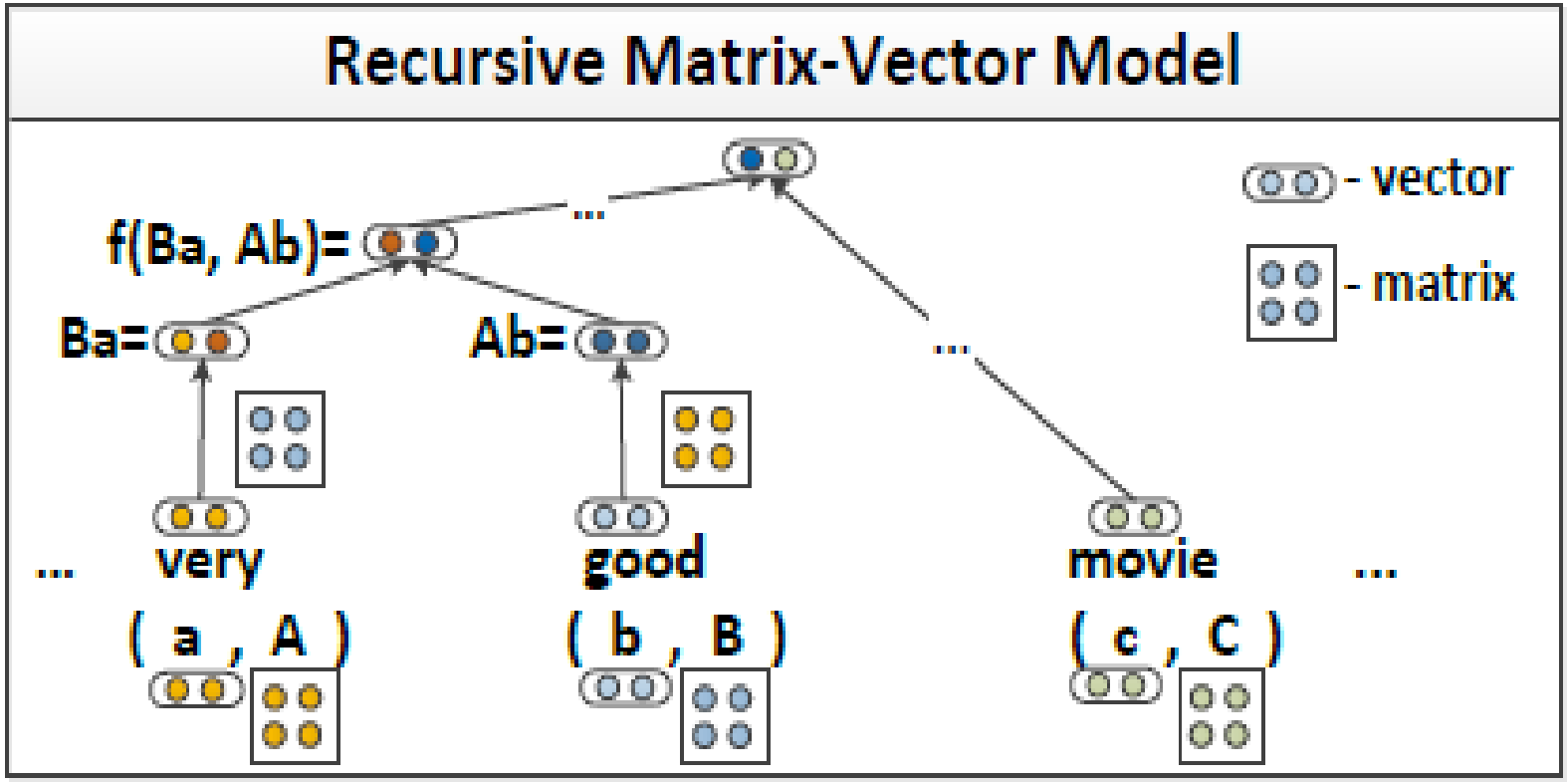
Parsing using Matrix Vector RNN, Socher et al, 2011





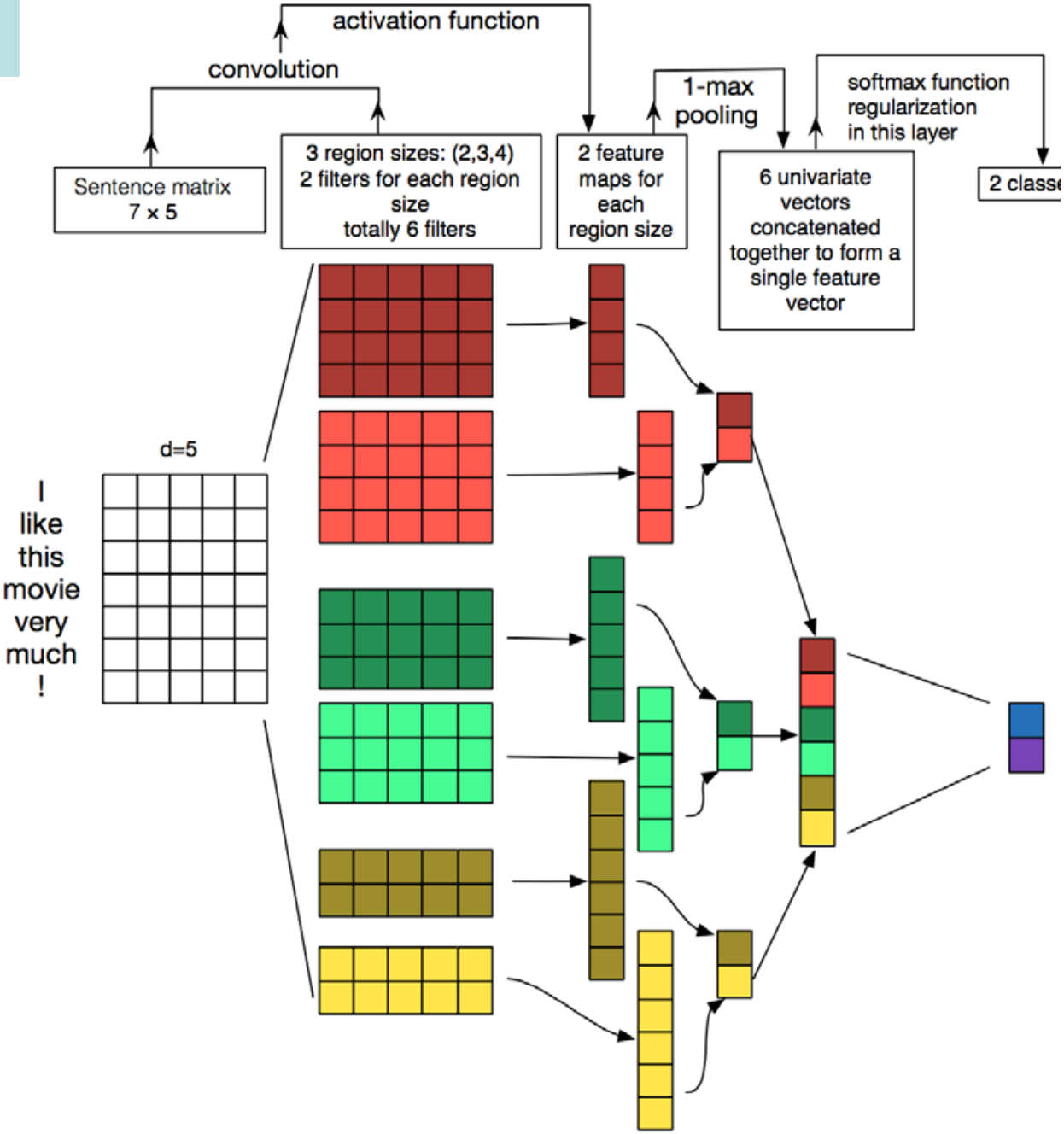
Applications

Parsing using Matrix Vector RNN,
Socher et al, 2011

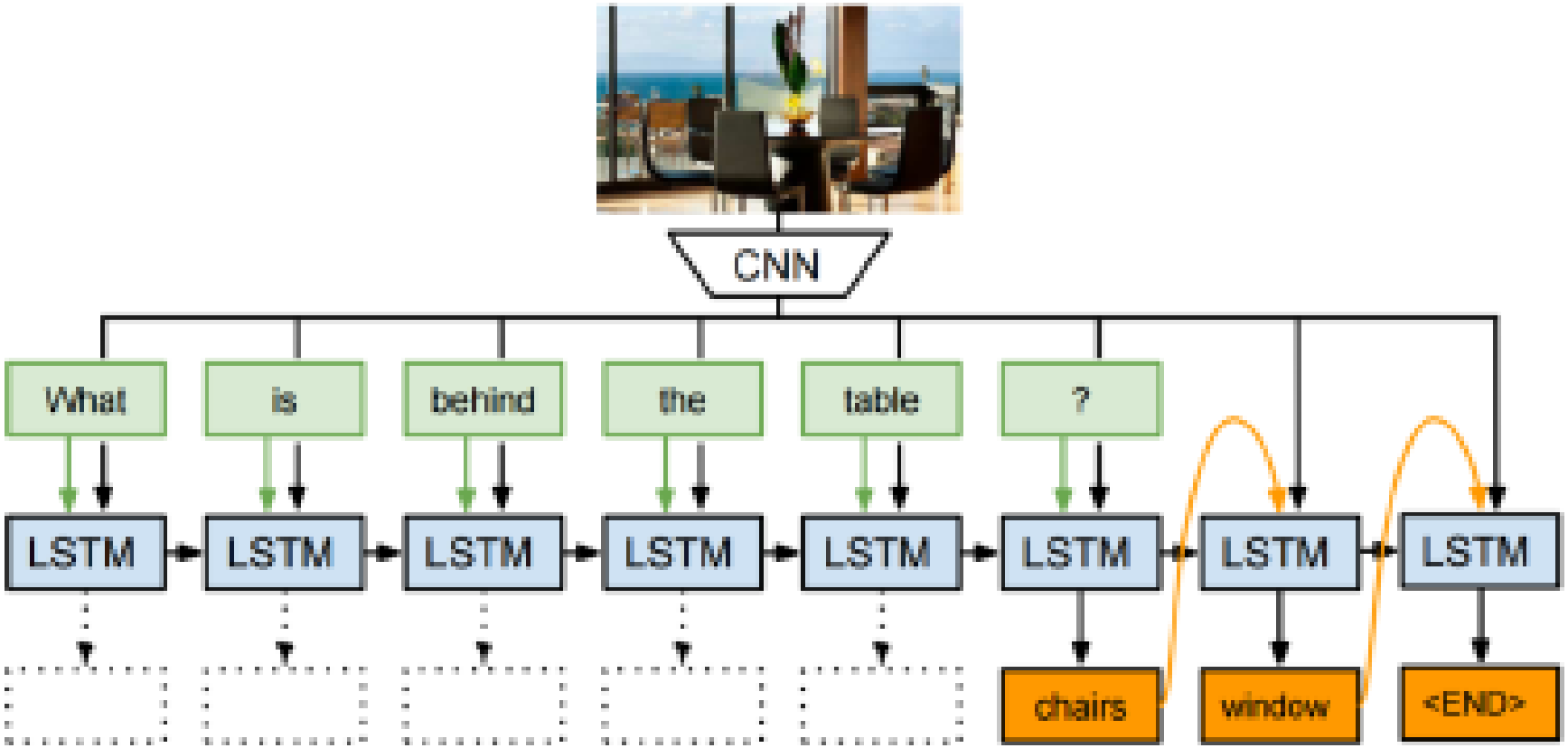


An example of using CNN for sentence classification

Zhang and Wallace, 2015



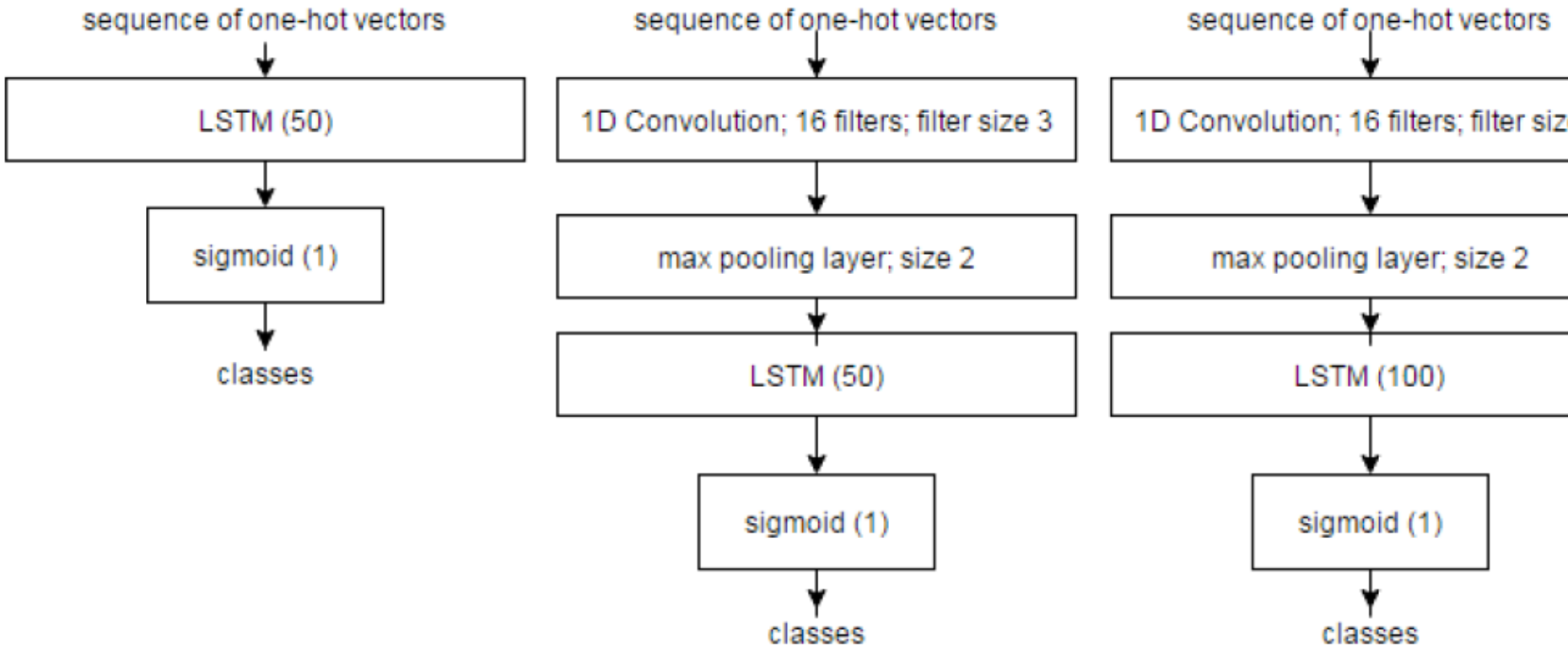
Neural-image QA (Malinowski et al. 2015)



Applications

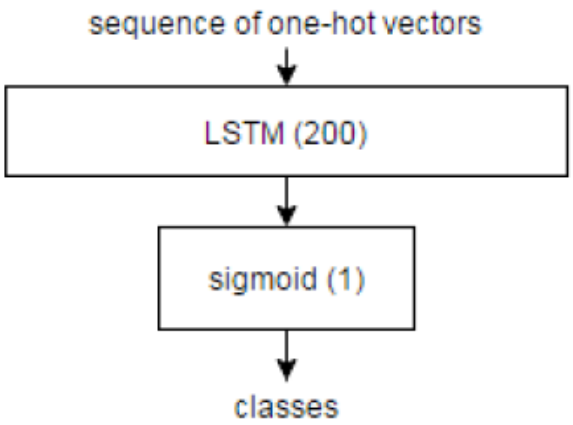
more examples of sentence classification

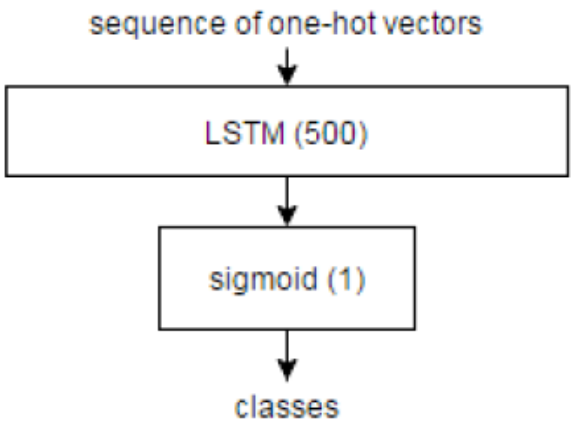
Aleksandr Kimashev, 2017,

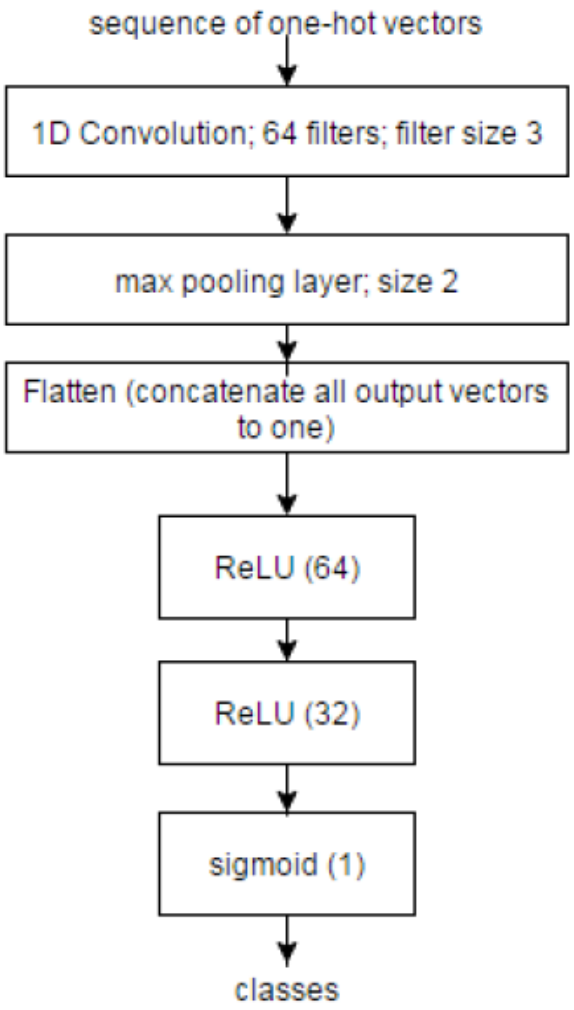


Applications

Aleksandr Kimashev, 2017,

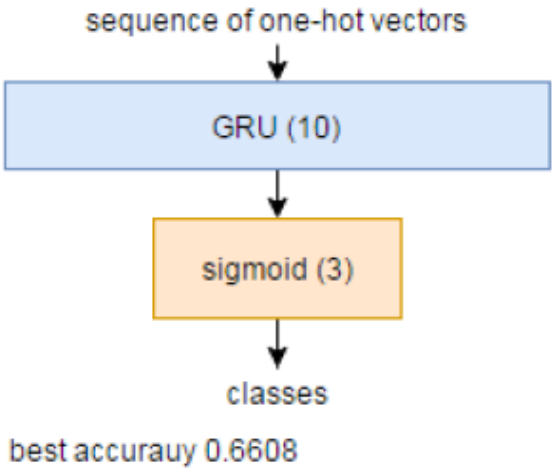


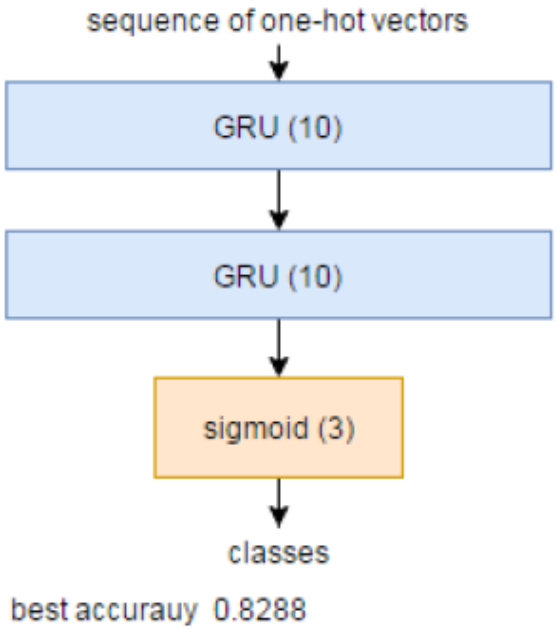


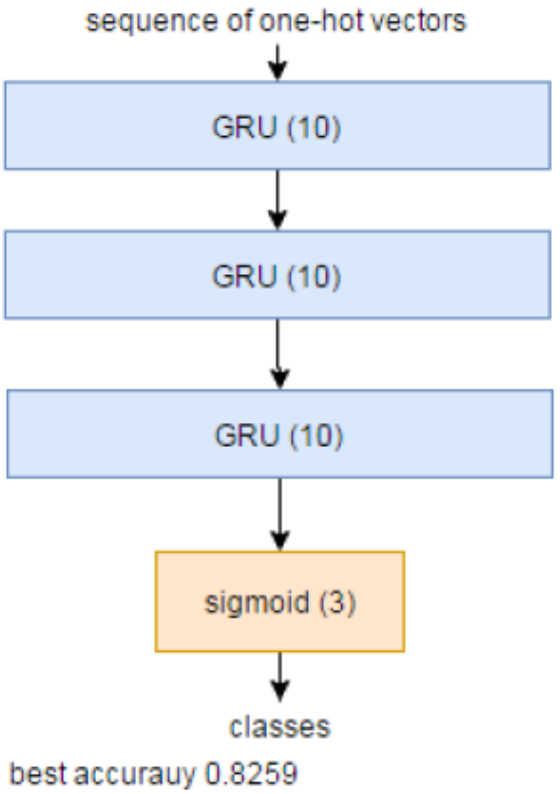


Applications

Aleksandr Kimashev, 2017,

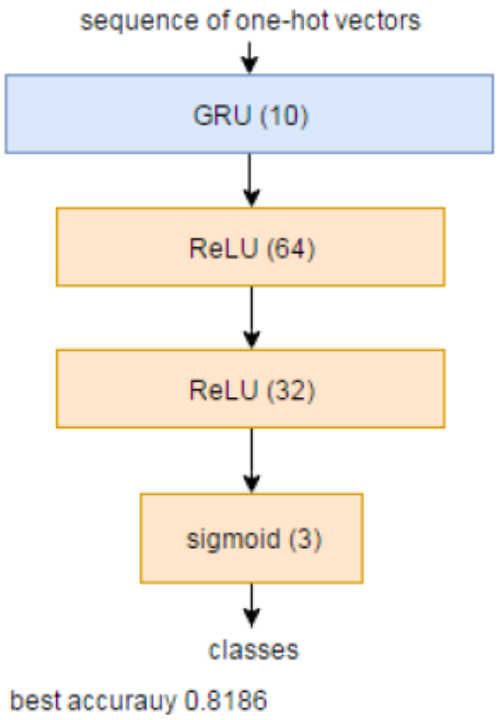


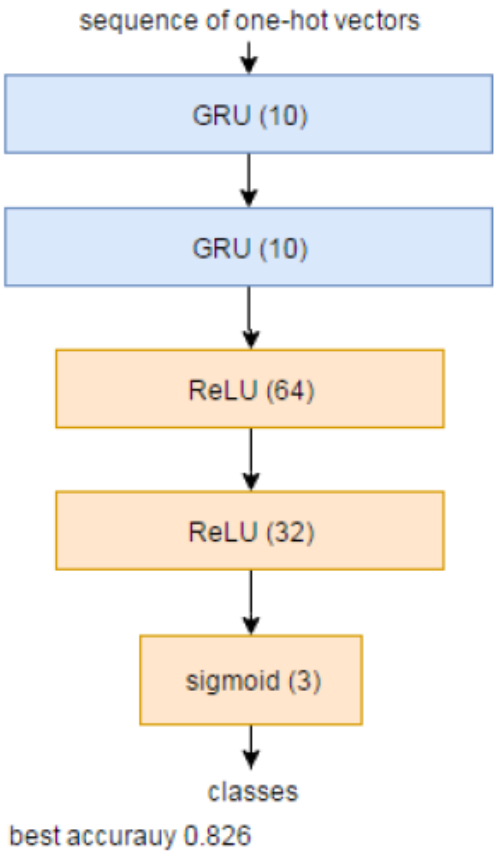


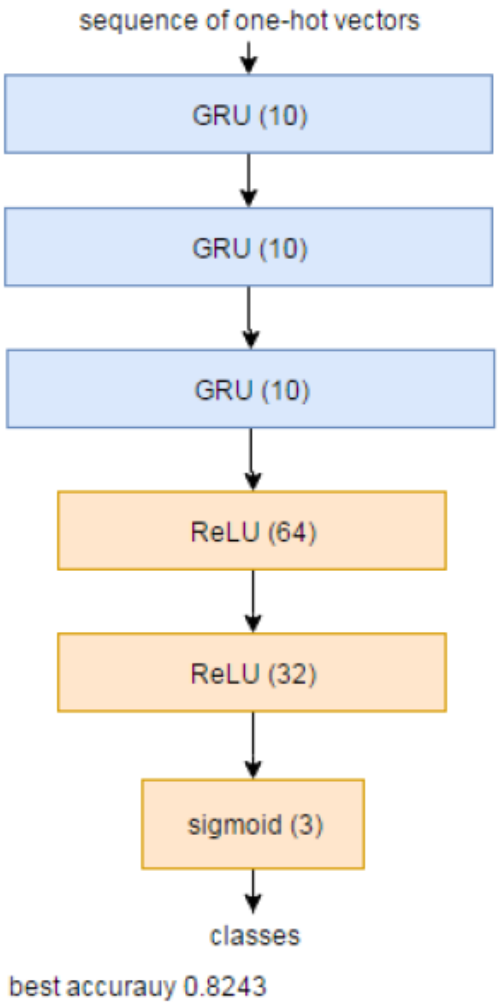


Applications

Aleksandr Kimashev, 2017,

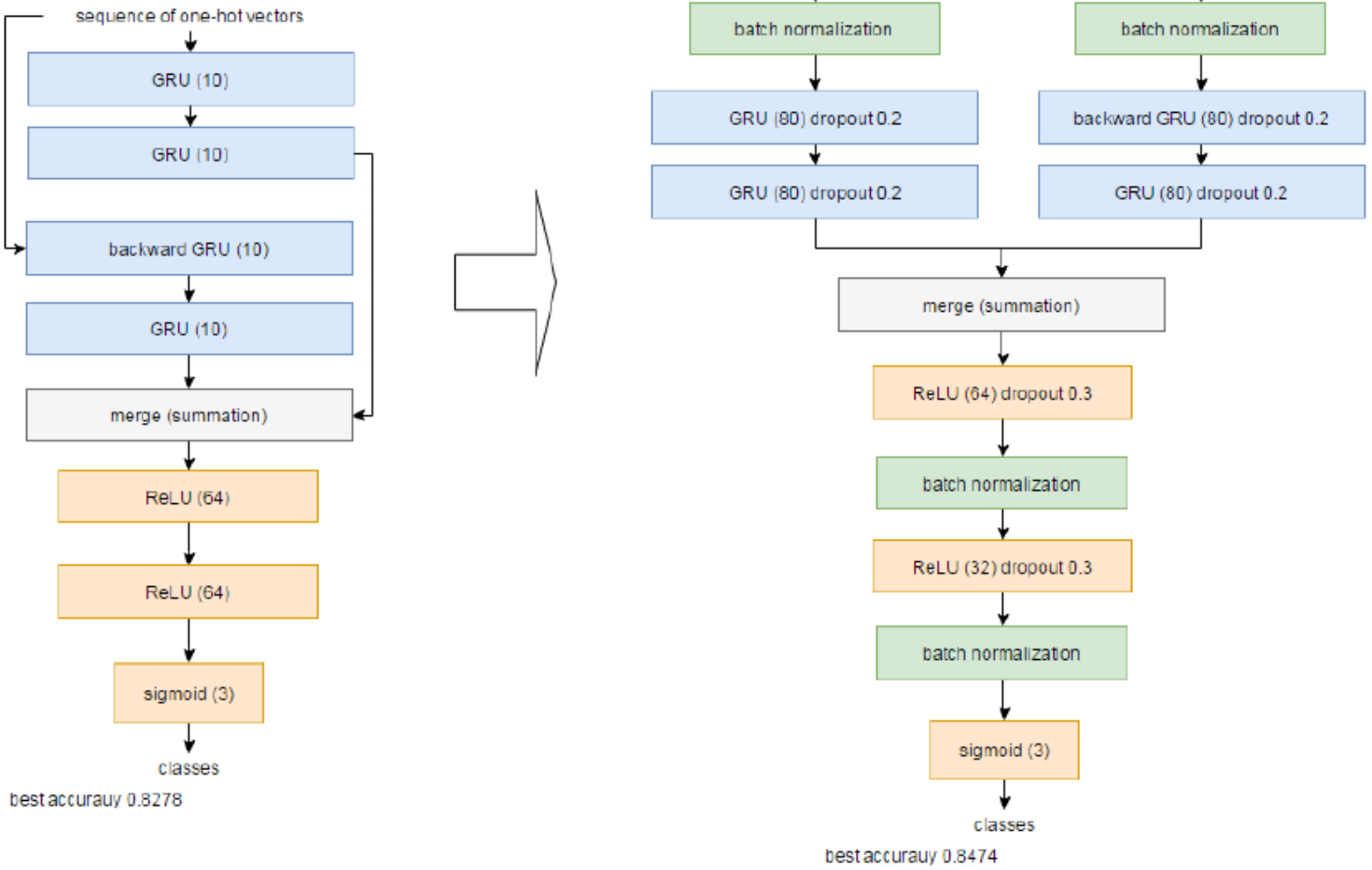






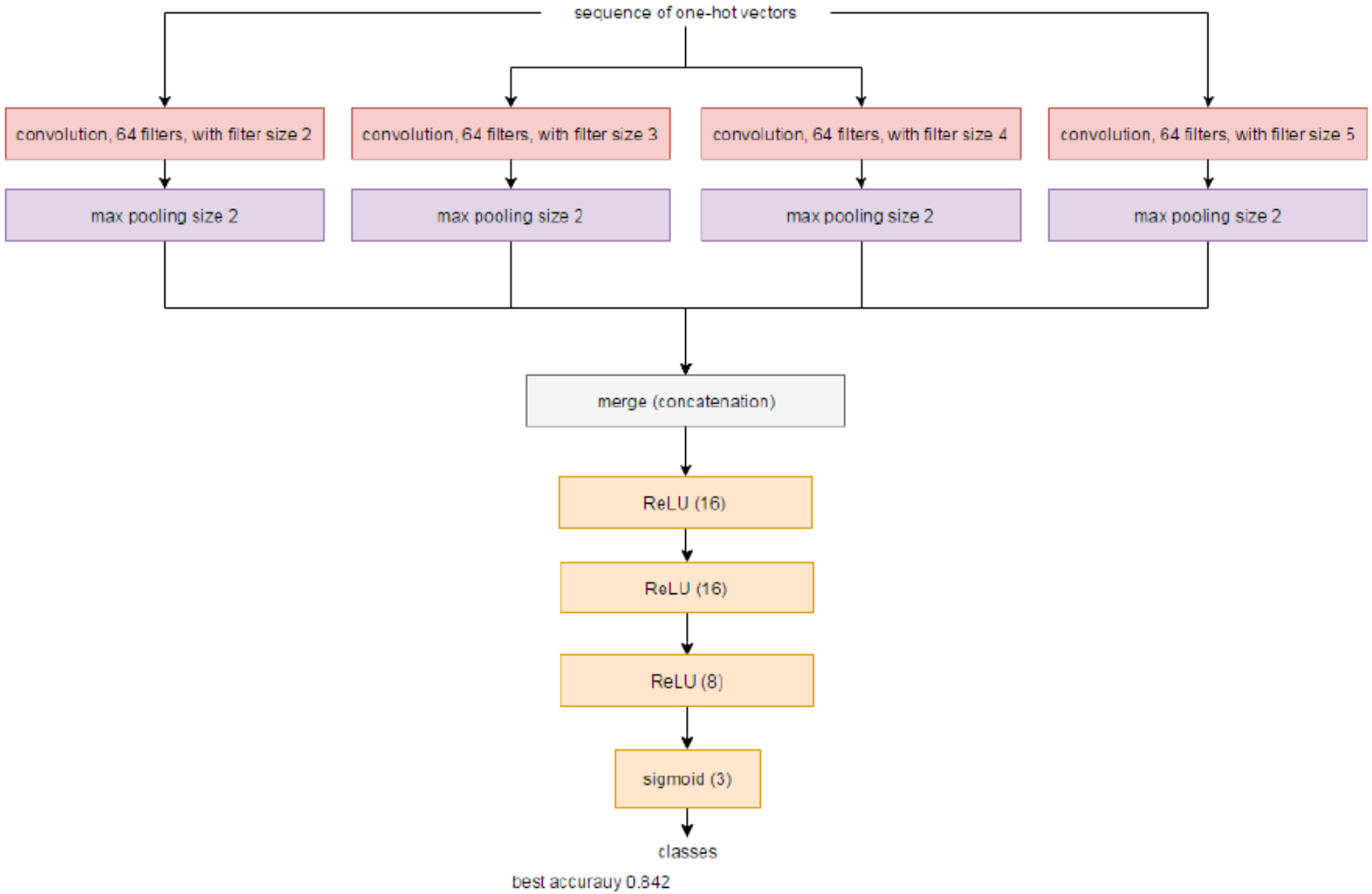
Applications

Aleksandr Kimashev, 2017,



Applications

Aleksandr Kimashev, 2017,



Outline

- Introduction
- Short review of Distributional Semantics, Semantic spaces, VSM, …
- Embeddings
  - Embedding of words
  - Embedding of more complex units
- Simple Linear Models
- Neural Networks models for NLP
- Applications
- Conclusions

Conclusions

- Embeddings
  - Good for words, LM, MT, Sum
    - Billion of words for learning models
    - Unsupervised learning from domain specific corpora
    - Probably better than LSI; LDA, …
    - Combining unsupervised learning with task-dependent supervised layers
  - Not so good for composition of words into more complex units
    - Convolution and pooling seem to be rather naïve approaches for dealing with word order and relevance.
    - Socher's approaches seem to go in the good direction
      - Including additional information beyond words: pos, parse, synsets, …
  - Nice to embed KB
    - Freebase, dbpedia, BioPortal, …
    - Other rdf (why not owl) modeled KB

Conclusions

- NN models
  - Many new models
  - Many forms of combination
    - Stacking
    - Bidirectional
    - Attention-based
    - Memory-based
    - Combining task-specific models for NN architectures
  - Combination with other approaches:
    - Reinforcement learning
    - Building NN from complex kernels (sequence, tree, graph)

Conclusions

- Deep Learning
  - Good Results in many NLP tasks
  - Need of big datasets for training
  - Good learning capabilities
    - Big models
    - Efficient use of computer resources, GPU, …
  - Difficult to interpret
    - Magic, miracle ???
    - Can we get conclusions from a successful model ??
  - Greedy learning of layers is ok??
  - How many layers ??
  - How many neurons in each layer ??
  - How about not NN-based models (deep graphical models, …) ??