An ensemble model for the machine reading comprehension dataset SQuAD

Summer report

María Fernanda Mora

September 12th, 2016

Carnegie Mellon University
Language Technologies Institute

Overview

- Problem definition
- Exploratory analysis
- Pipeline description
- Sentence ranking
- Answer extraction
- Implementation

Problem definition

Problem definition

- Implement a system capable of performing reading comprehension over SQuAD's data set that outperforms the current state of the art.
- *SQuAD's challenge:*
  - No candidate answers provided
  - A correct answer to a question can be any sequence of tokens from the given text
  - Q&A in SQuAD were created by humans, hence more realistic

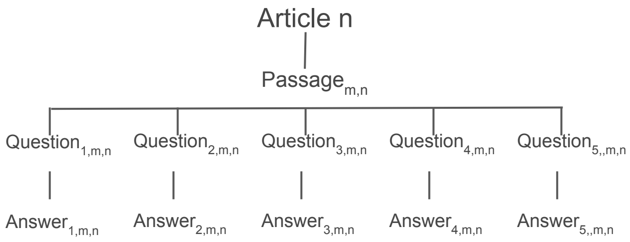
Exploratory analysis

Exploratory analysis

- General statistics
- Lexical analysis
- Syntactic analysis

Complete dataset

- 536 Wikipedia articles
- 108K QA pairs
- Training, dev and test
- Hierarchical view:



Model evaluation

- Output: sequence of tokens
- Measures: Exact match, F1

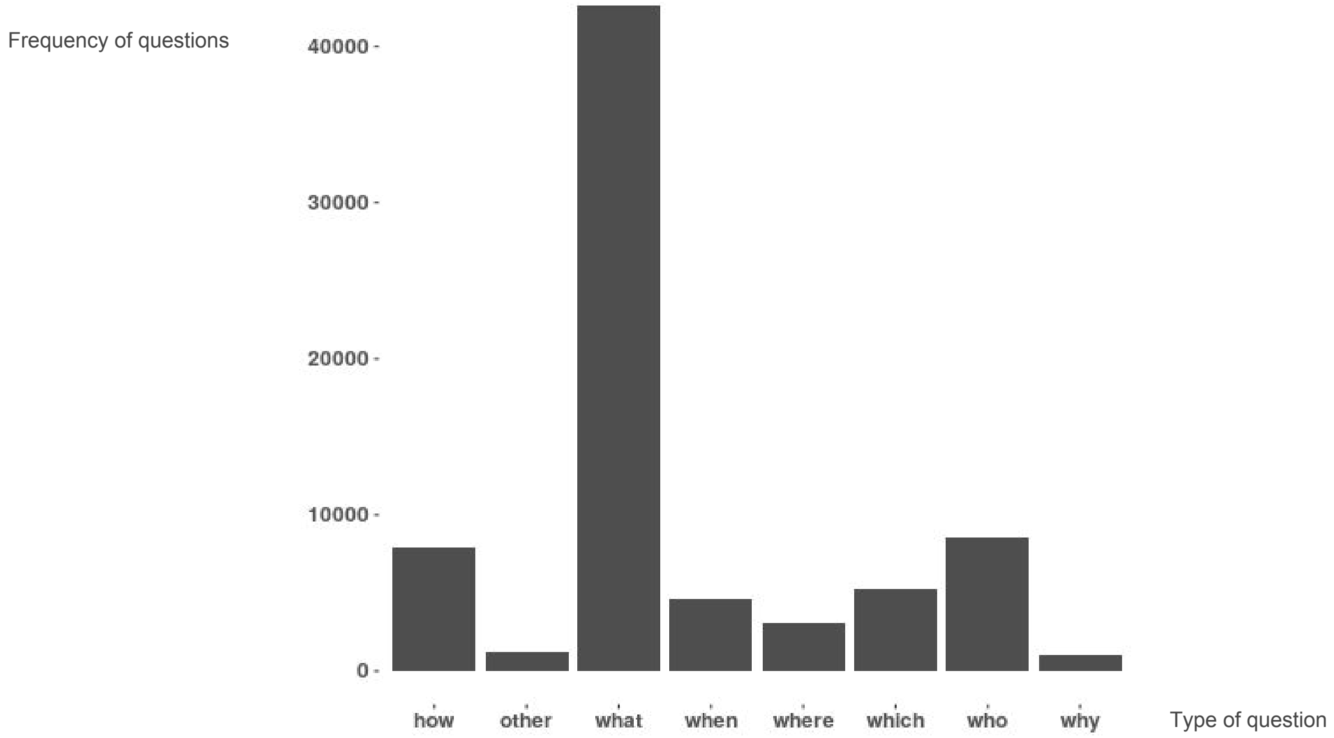
Training dataset

- 378 Wikipedia articles
- ~ 42 passages per article
- 5 questions per passage
- 1 answer per question
- ~ 80K QA pairs

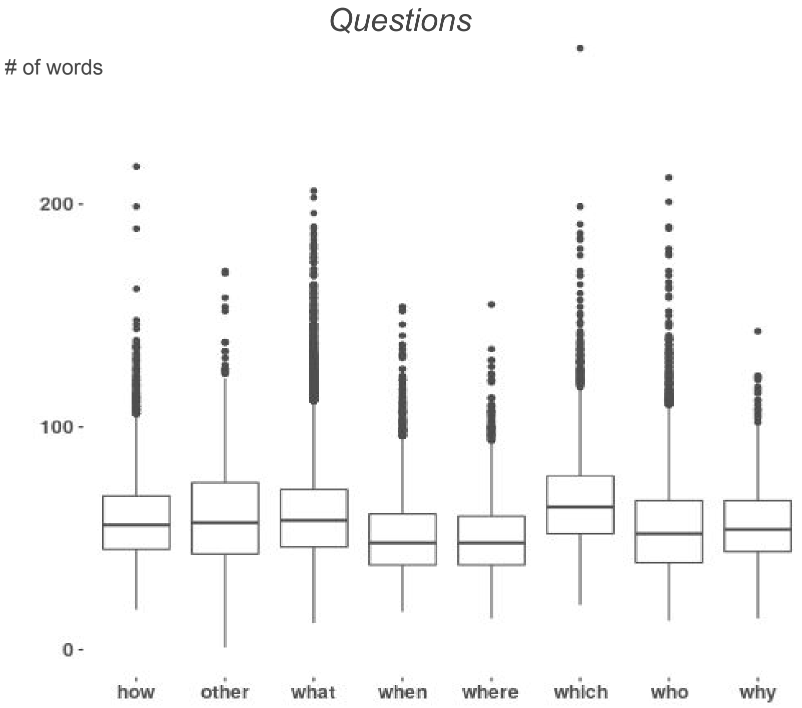
Vocabulary Size

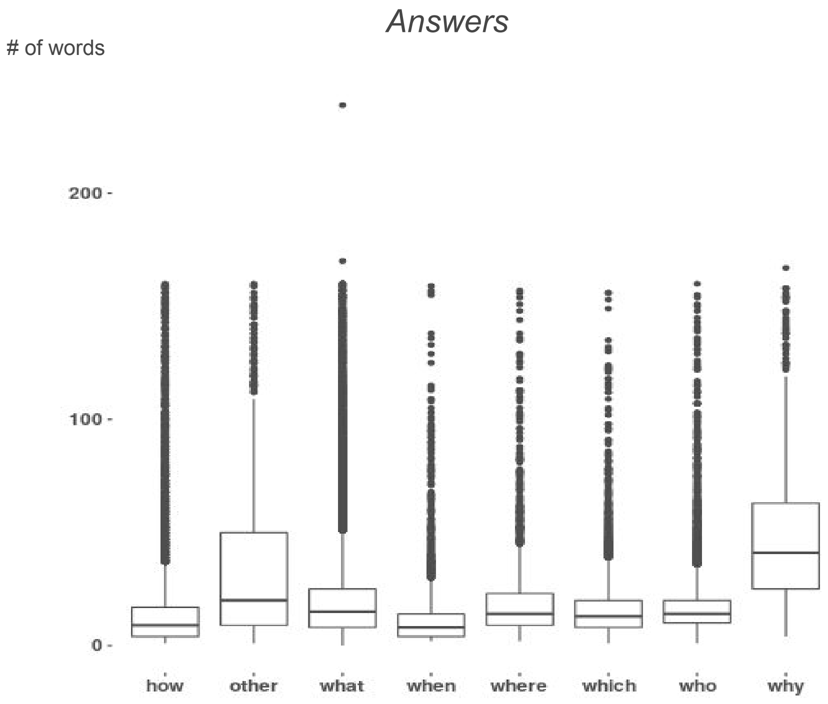
	# words
Passages	~88K (98% without stop words)
Questions	~1K (93% w/o stop words)
Answers	~0.5K (93% w/o stop words)

>99% of the questions are factoid; >50% are *what* questions



Questions length is similar; answers to *why* and *other* questions show length variation



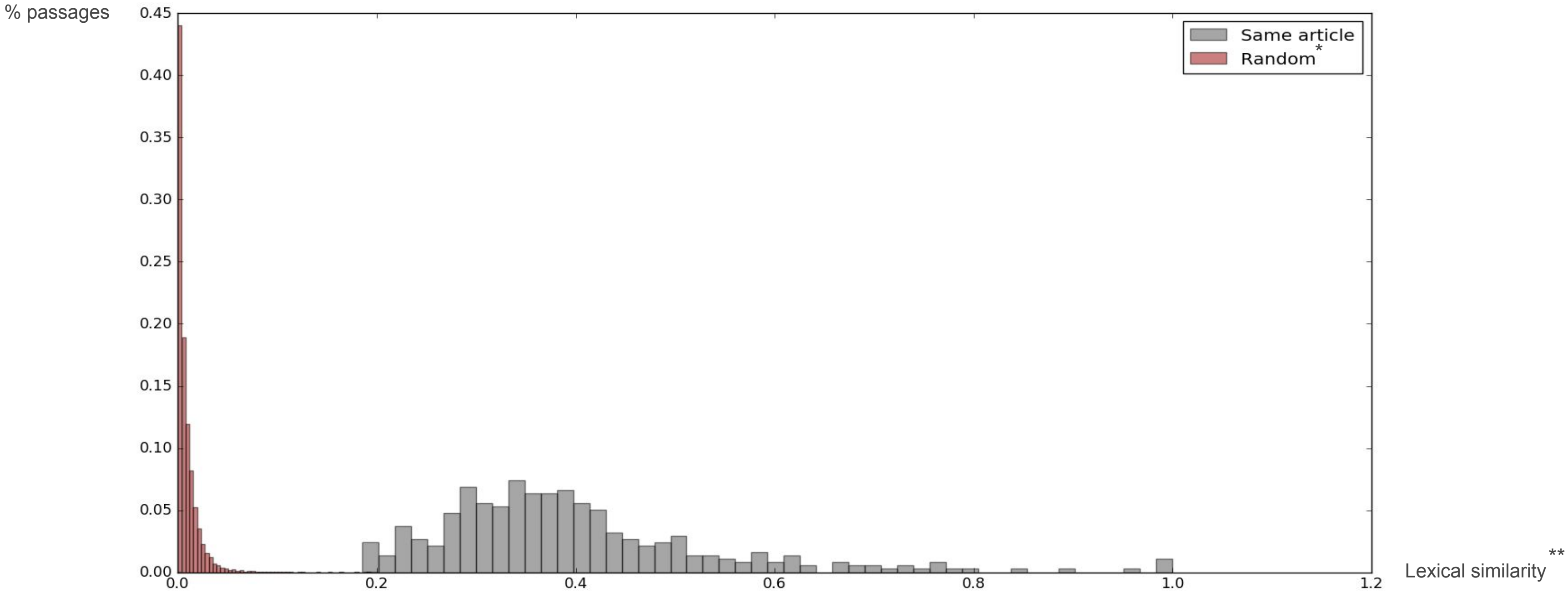


Questions are larger than answers; *why* questions have the largest answers but represent <5%

Exploratory analysis

- General statistics
- Lexical analysis
- Syntactic analysis

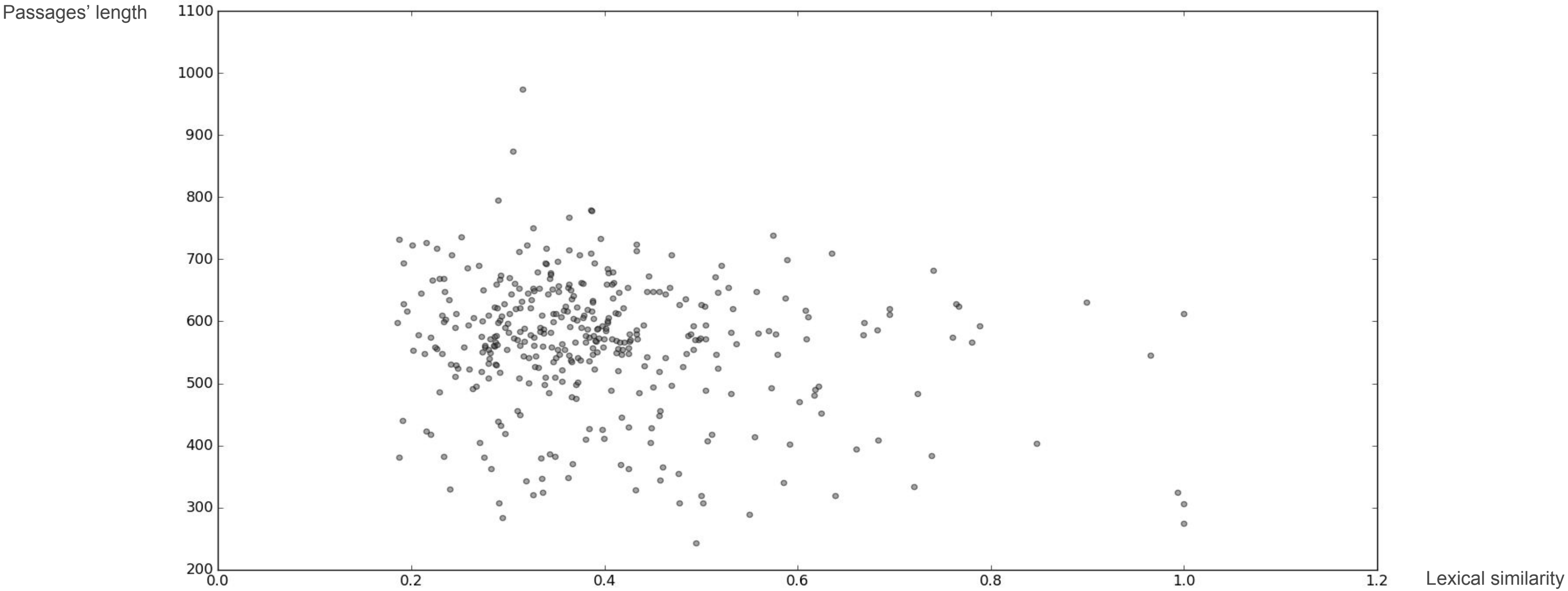
There exists a lexical similarity 0.3-0.4 between passages of the same article



* Random passages were extracted from all the articles

** Measured as cosine similarity

This similarity is independent of the length of the passage



LDA analysis varying number of words and topics showed the following persistent topics

- history
- government
- nation-state
- sports
- art

Exploratory analysis

- General statistics
- Lexical analysis
- Syntactic analysis
    - Embeddings
        - Word
        - Sentence
        - Paragraph

Word embeddings pipeline

Word embeddings	Dimensionality reduction	Clustering	Visualization
Linear Syntactic Topical	t-SNE	k-means	scatterplot

Models

- Glove
- Skip-gram

Parameters

- Window size
- Vector size

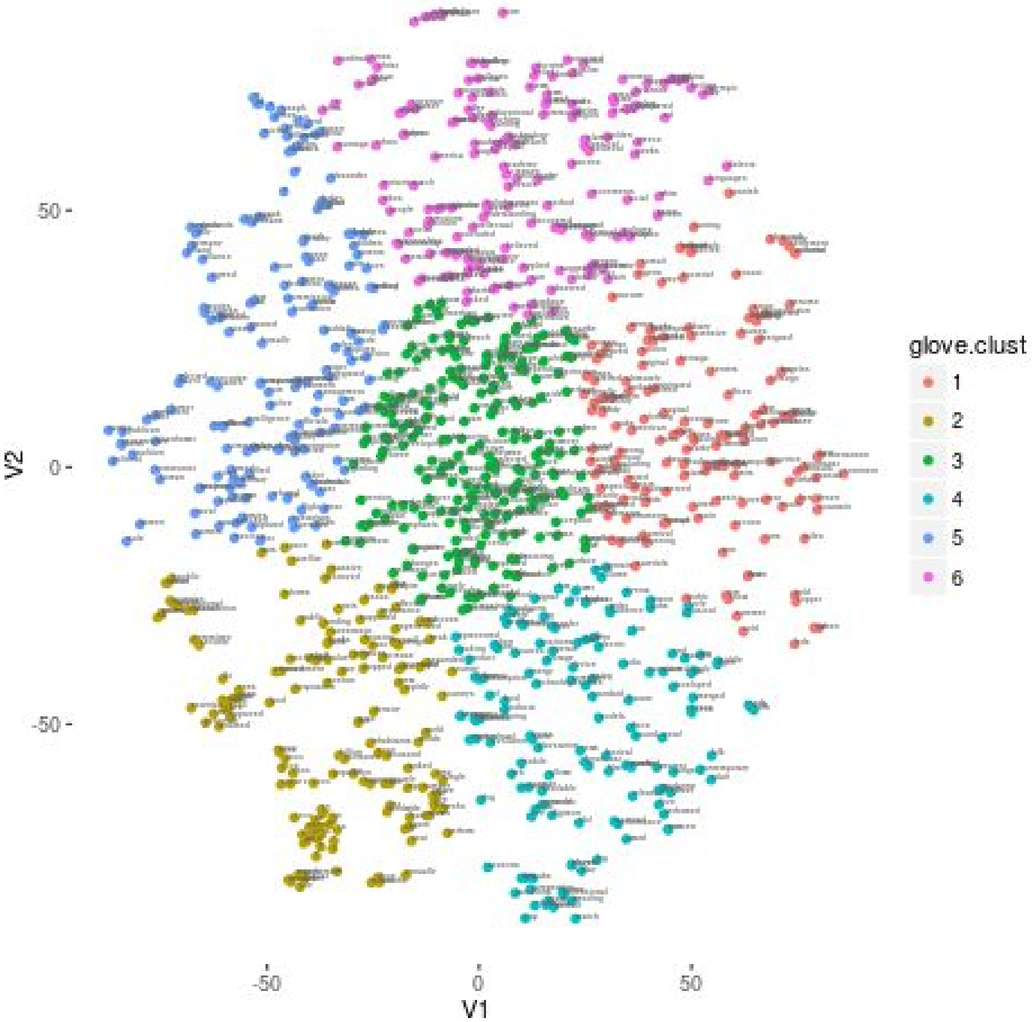
GLOVE Linear Embedding

- Min words in voc = 100
- Size of vectors = 100, 300, 500
- Size of window = 5, 15, 20

GLOVE Linear Embedding

Window Size = 15

Vector Size = 100

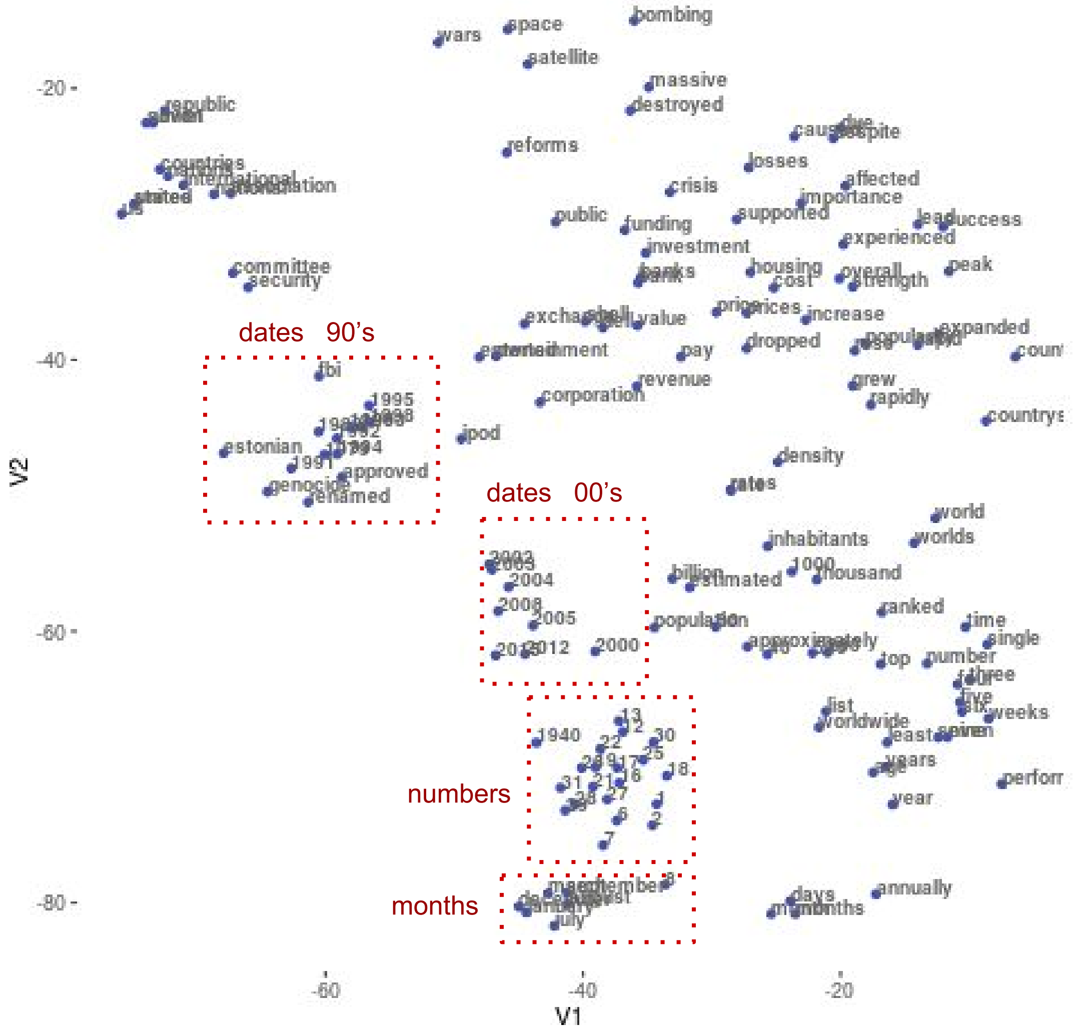


GLOVE Linear Embedding

Window Size = 15

Vector Size = 100

Cluster = 2

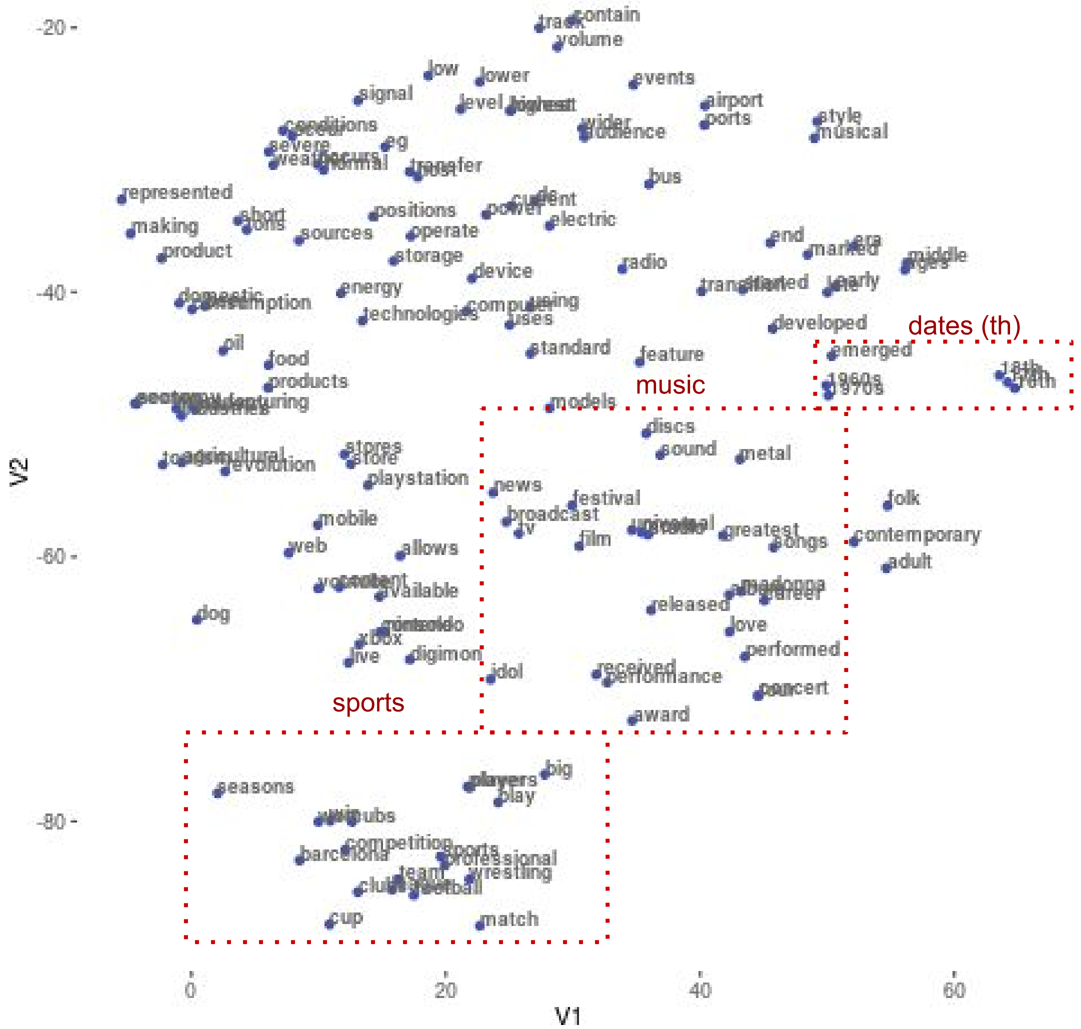


GLOVE Linear Embedding

Window Size = 15

Vector Size = 100

Cluster = 4

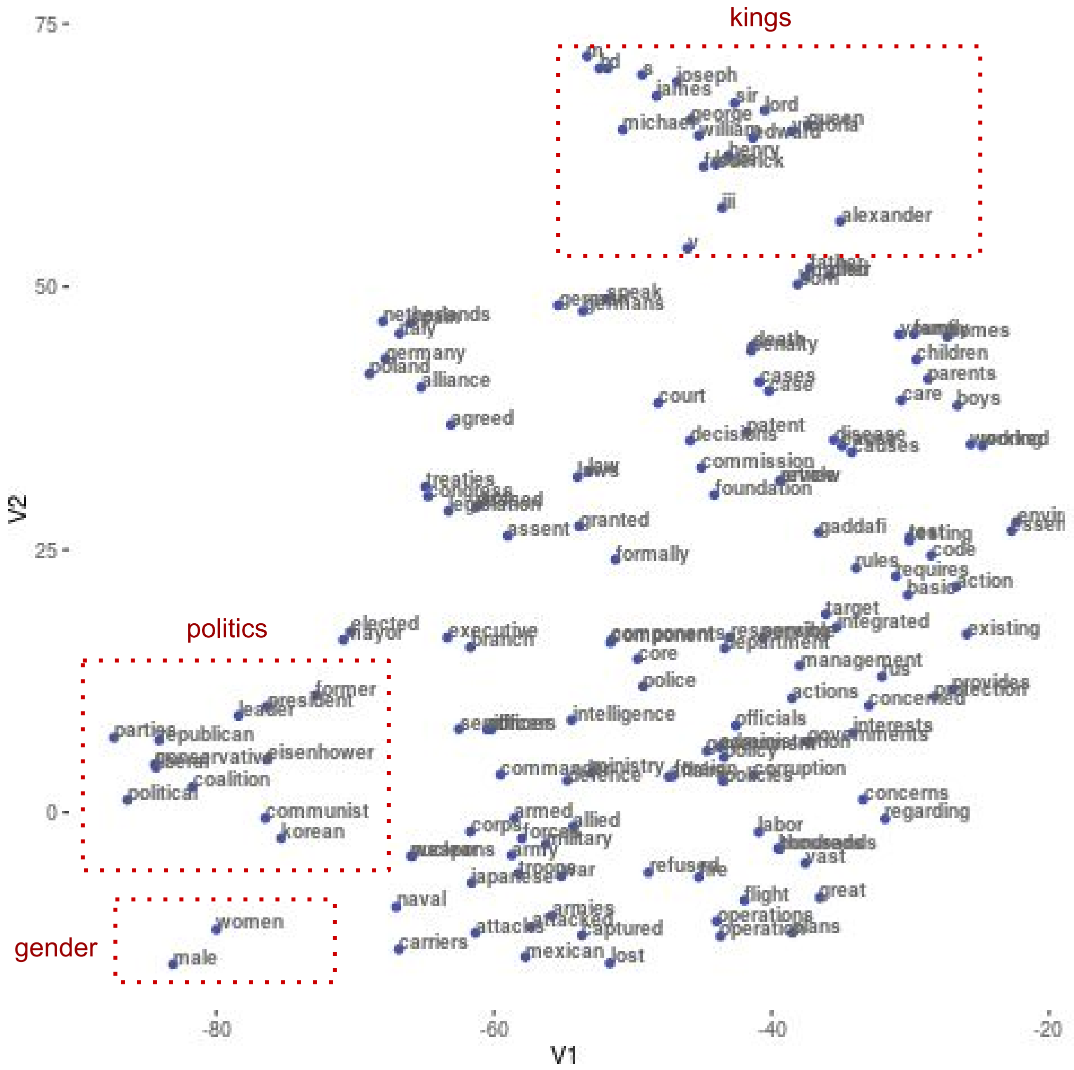


GLOVE Linear Embedding

Window Size = 15

Vector Size = 100

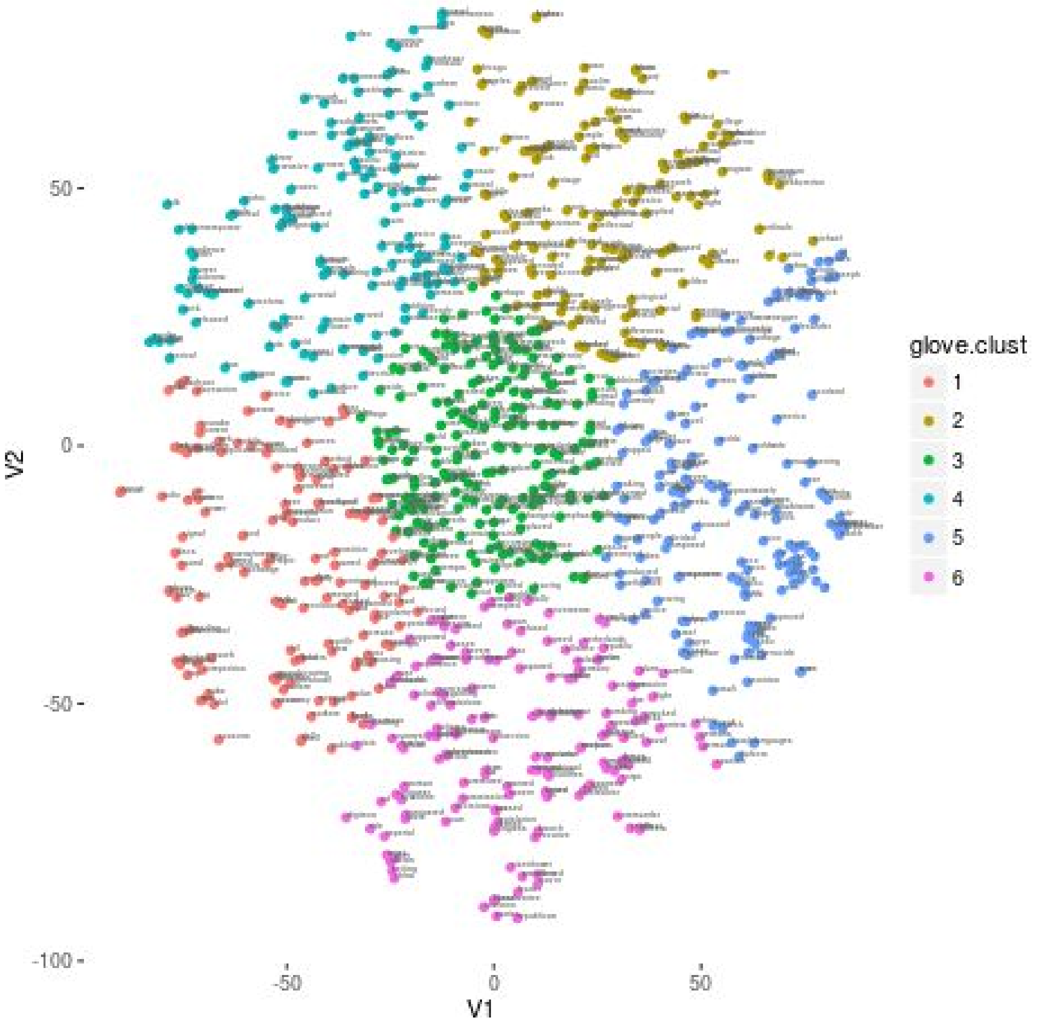
Cluster = 5



GLOVE Linear Embedding

Window Size = 20

Vector Size = 100

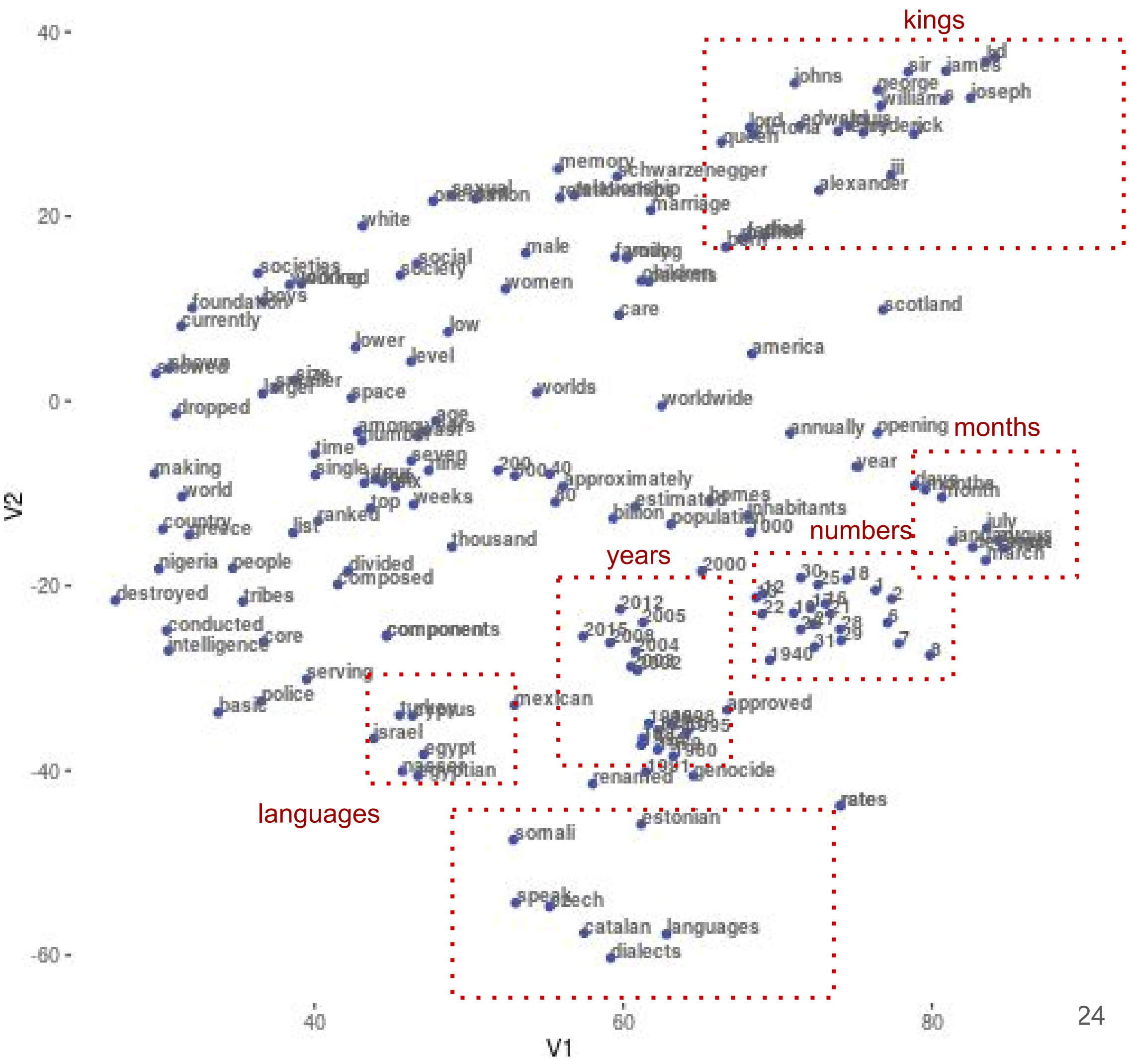


GLOVE Linear Embedding

Window Size = 20

Vector Size = 100

Cluster = 5

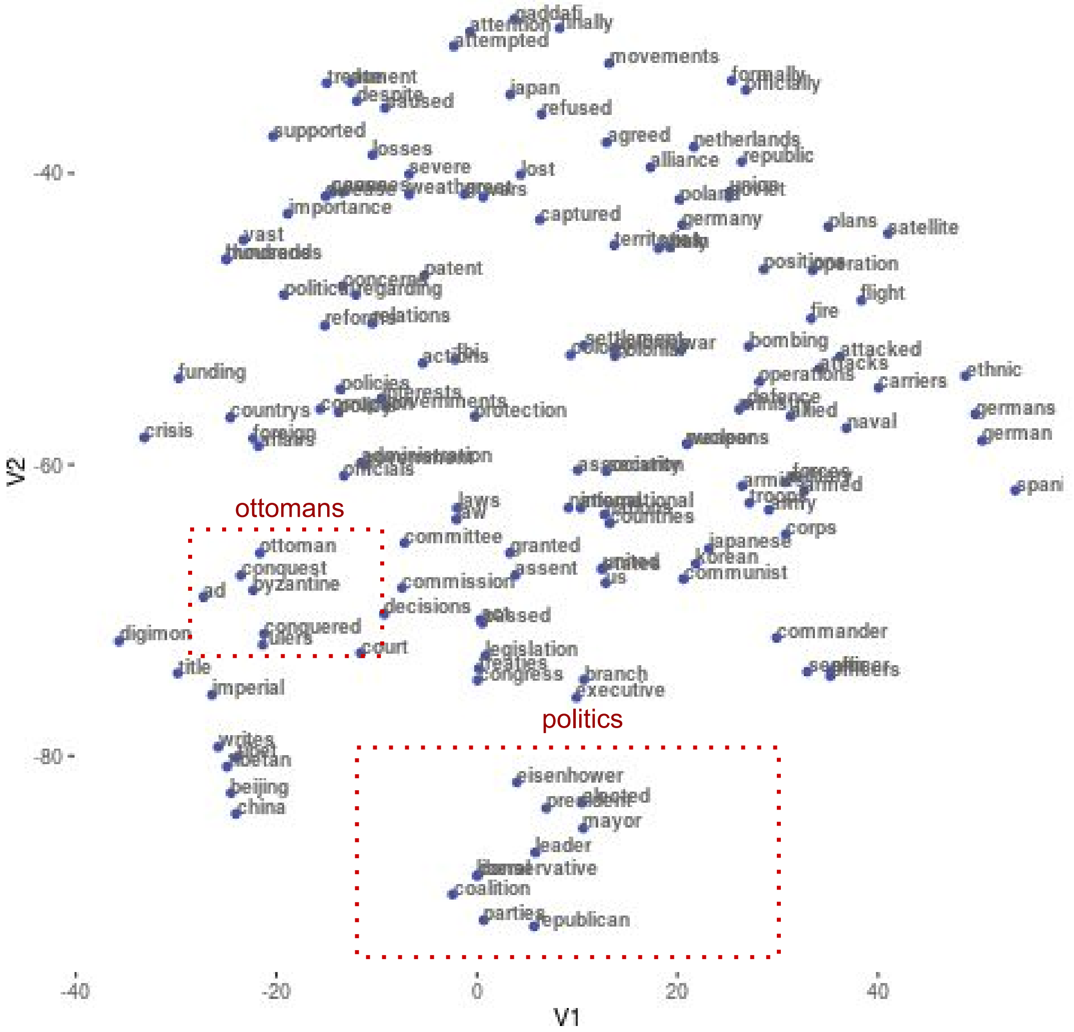


GLOVE Linear Embedding

Window Size = 20

Vector Size = 100

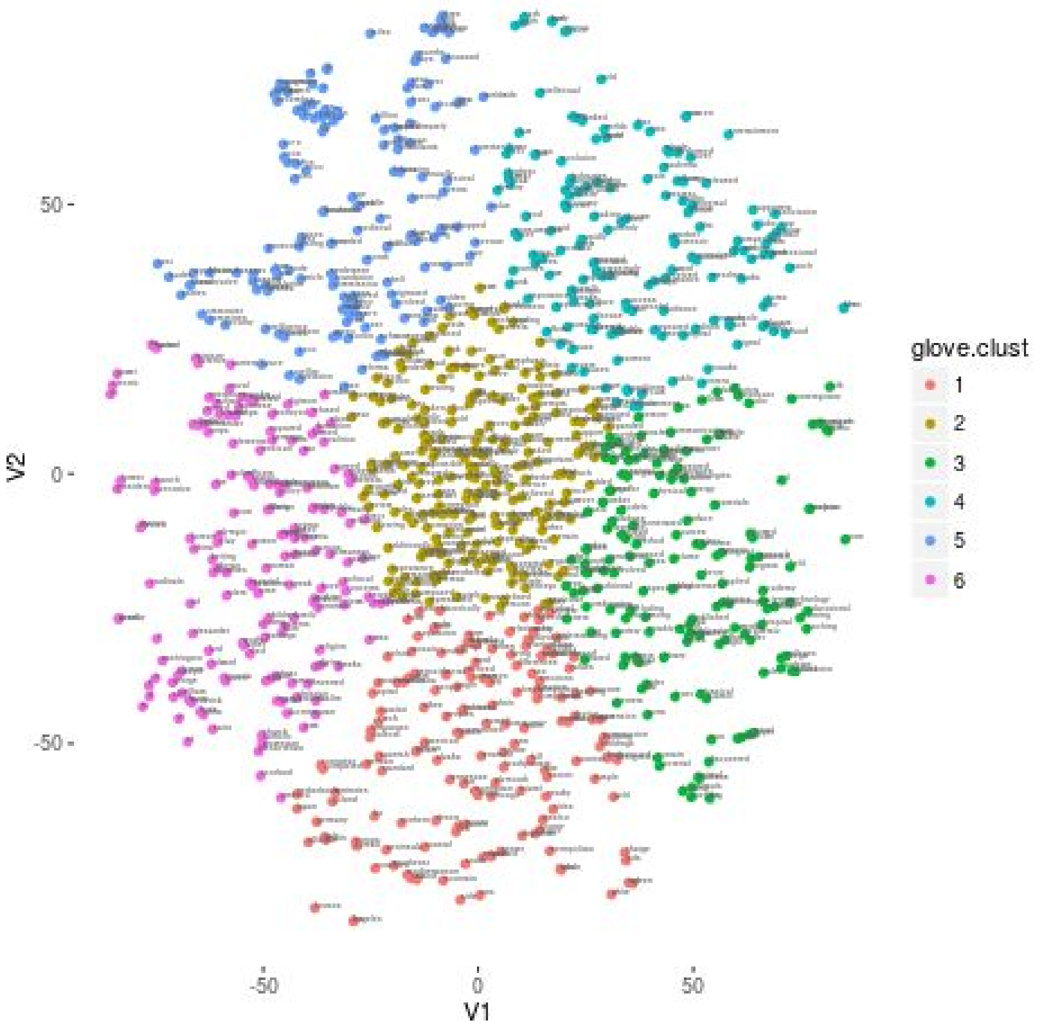
Cluster = 6



GLOVE Linear Embedding

Window Size = 5

Vector Size = 100

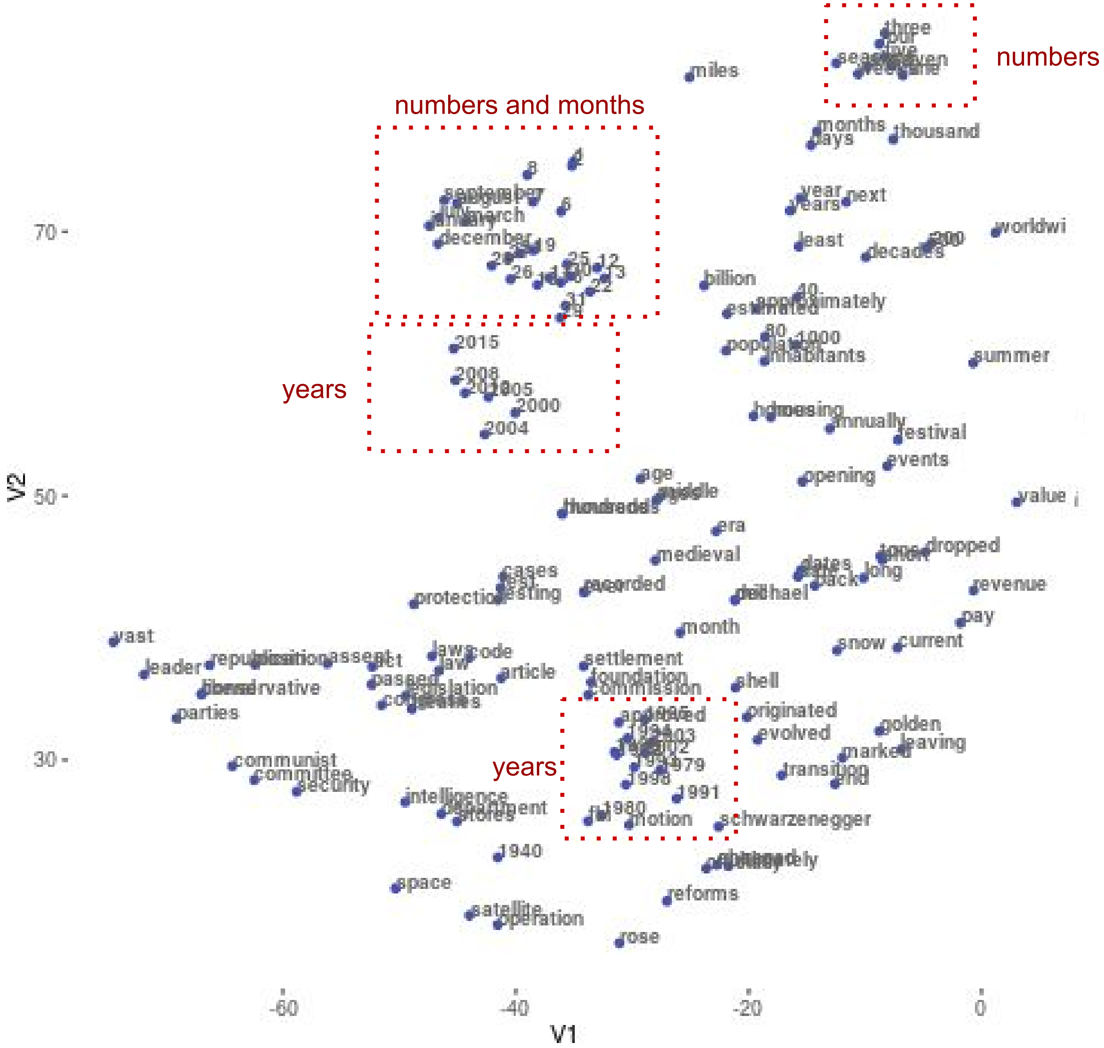


GLOVE Linear Embedding

Window Size = 5

Vector Size = 100

Cluster = 5

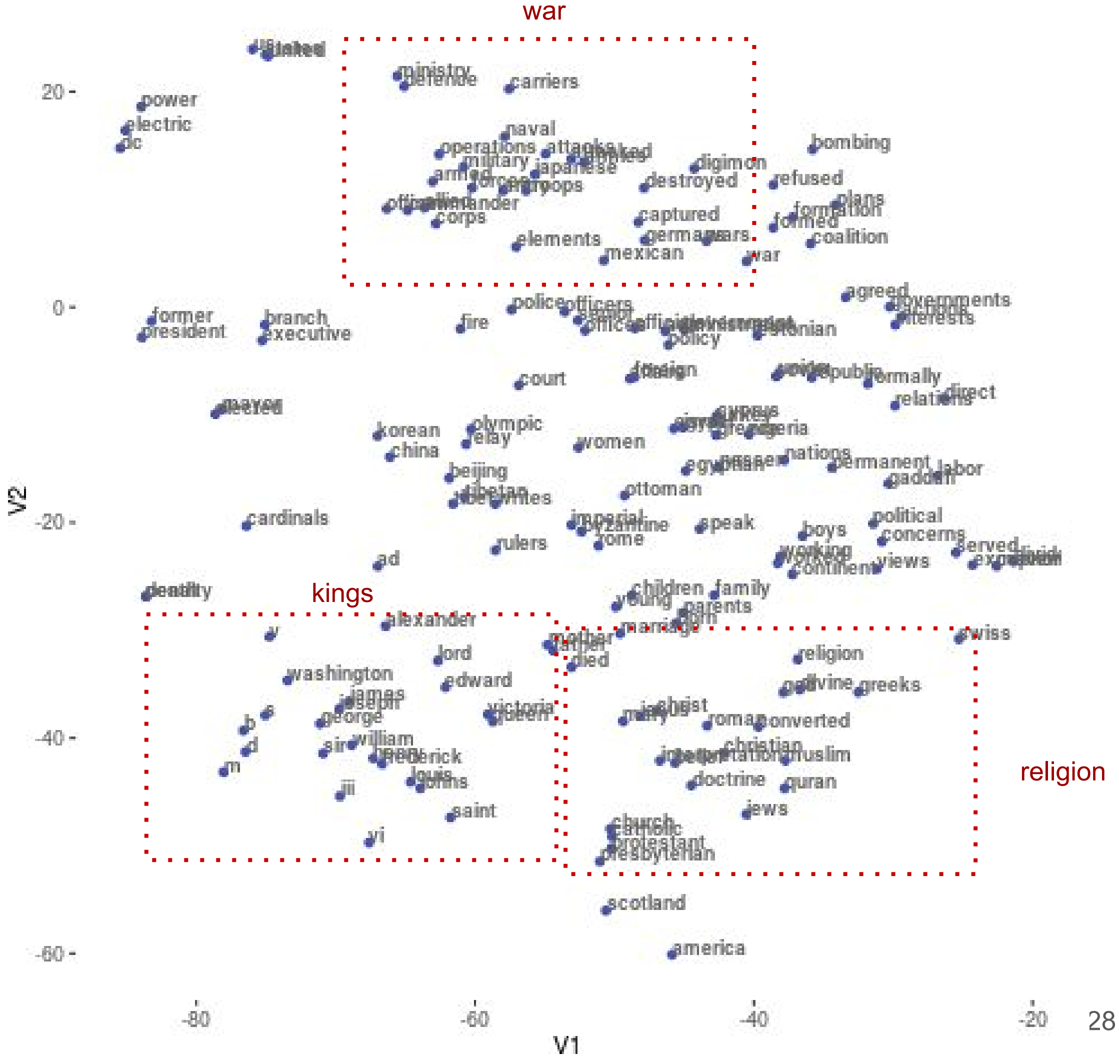


GLOVE Linear Embedding

Window Size = 5

Vector Size = 100

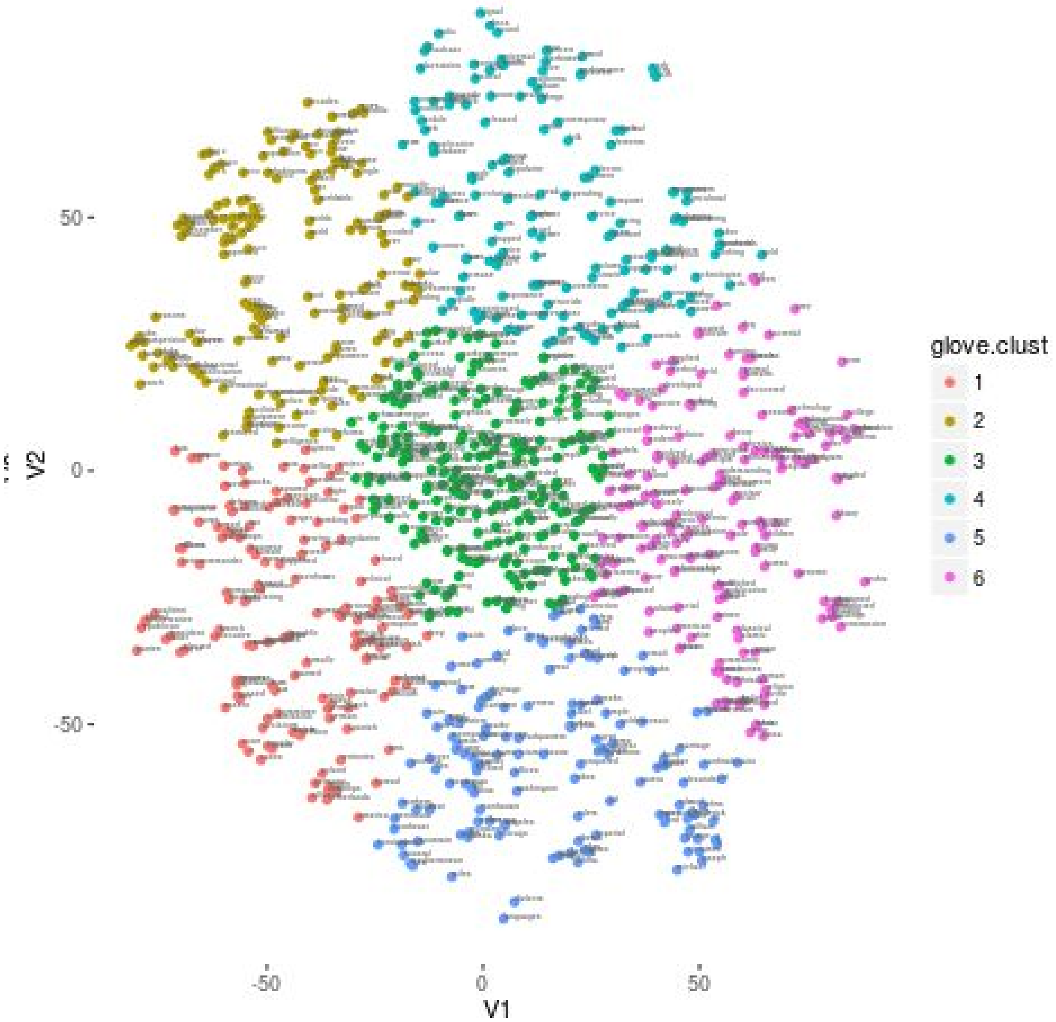
Cluster = 6



GLOVE Linear Embedding

Window Size = 15

Vector Size = 500

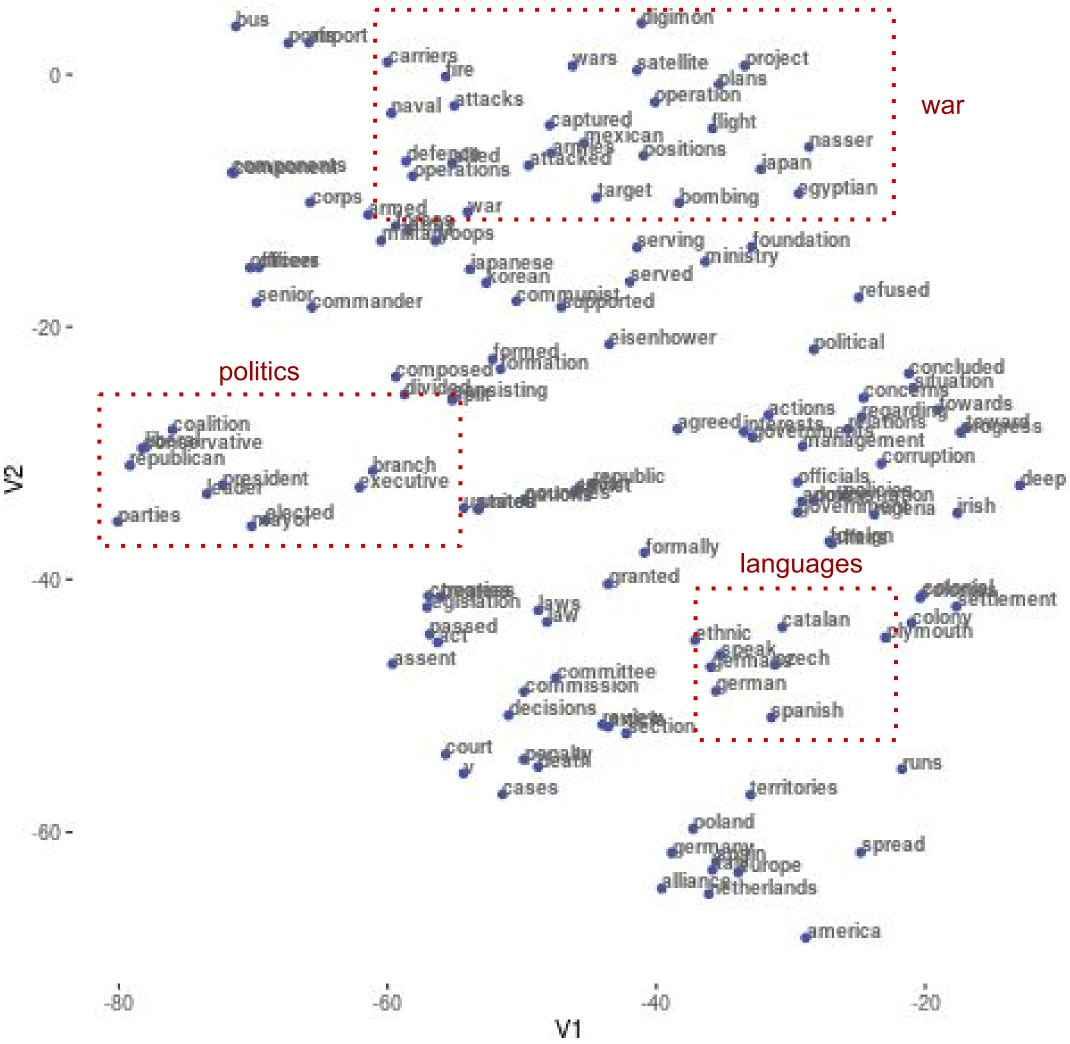


GLOVE Linear Embedding

Window Size = 15

Vector Size = 500

Cluster = 1

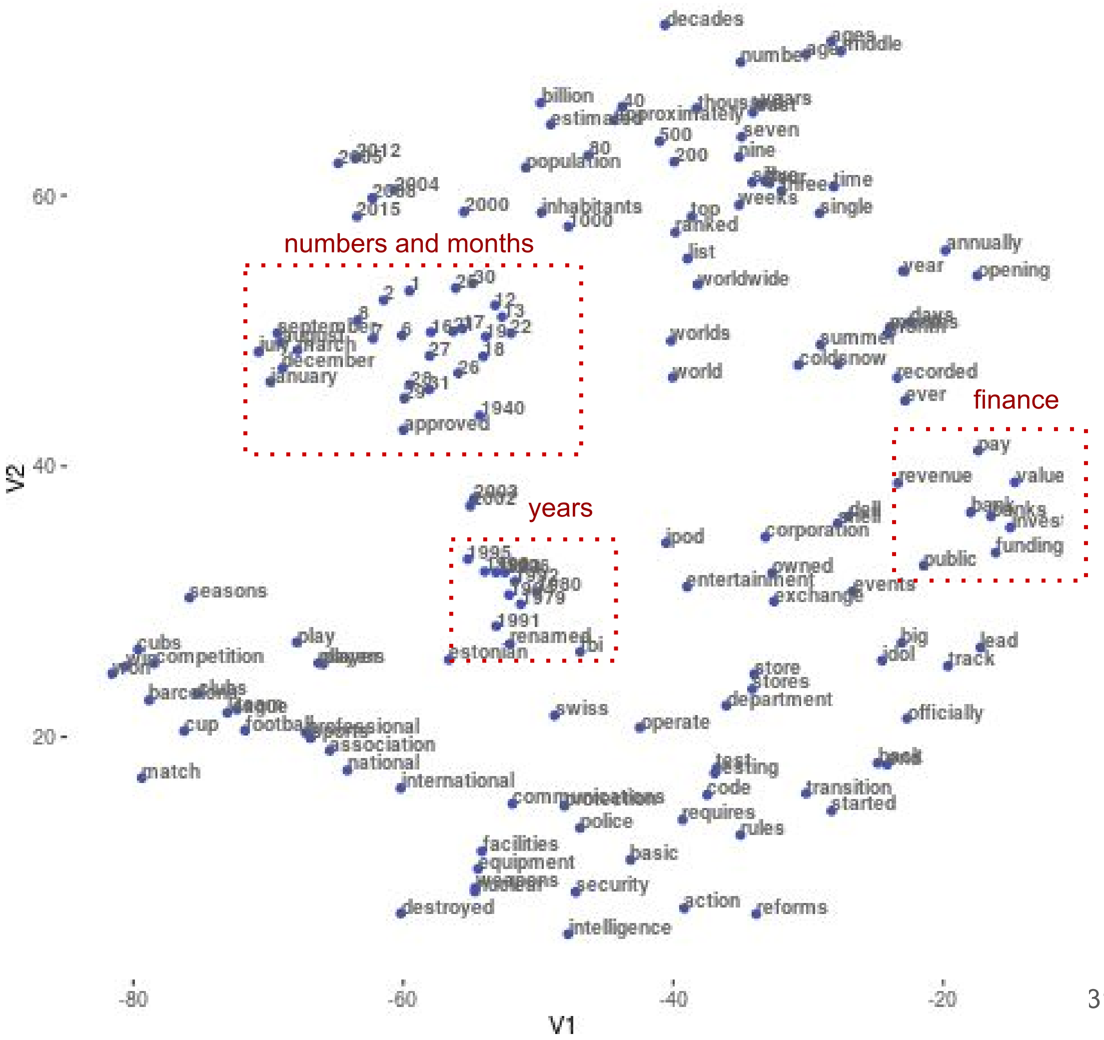


GLOVE Linear Embedding

Window Size = 15

Vector Size = 500

Cluster = 2



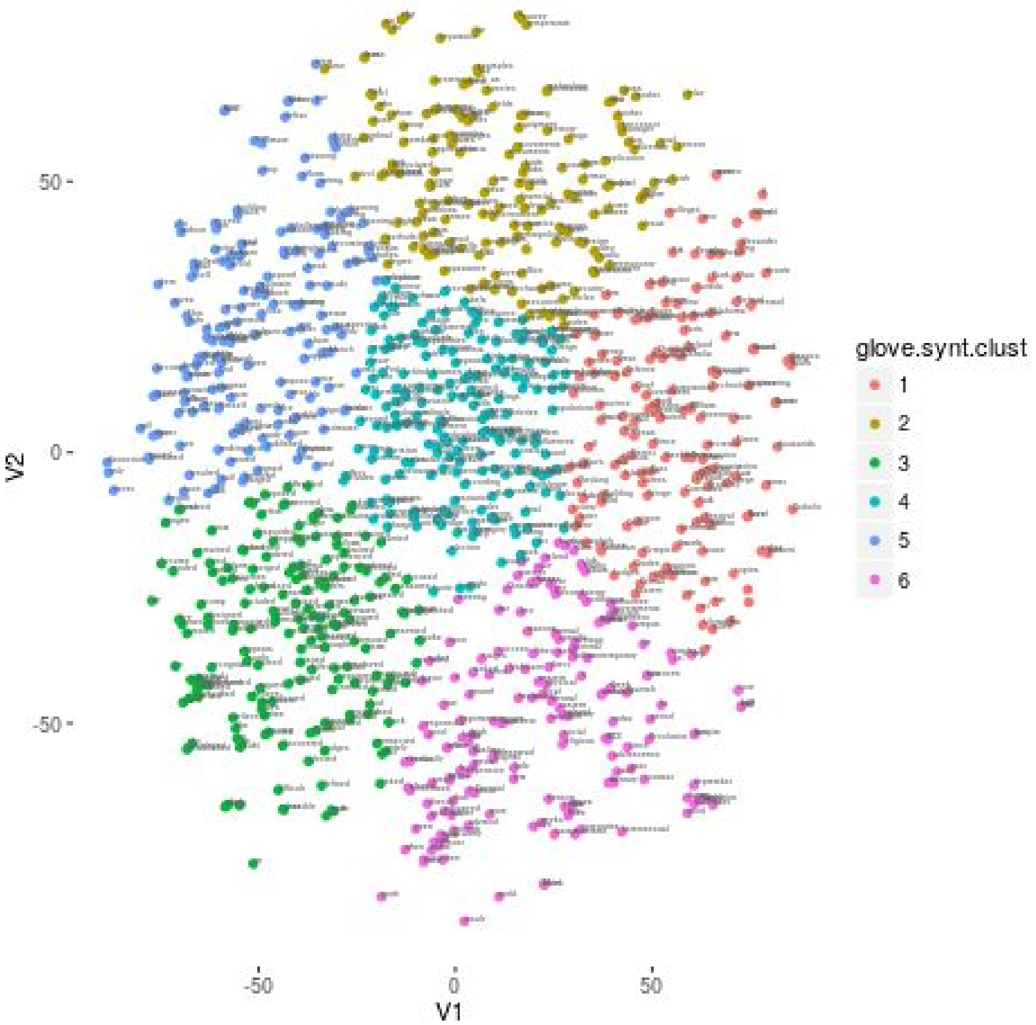
GLOVE Syntactic Embedding

- Min words in voc = 100
- Size of vectors = 100, 300, 500
- Size of window = 5, 15, 20

GLOVE Syntactic Embedding

Window Size = 15

Vector Size = 300



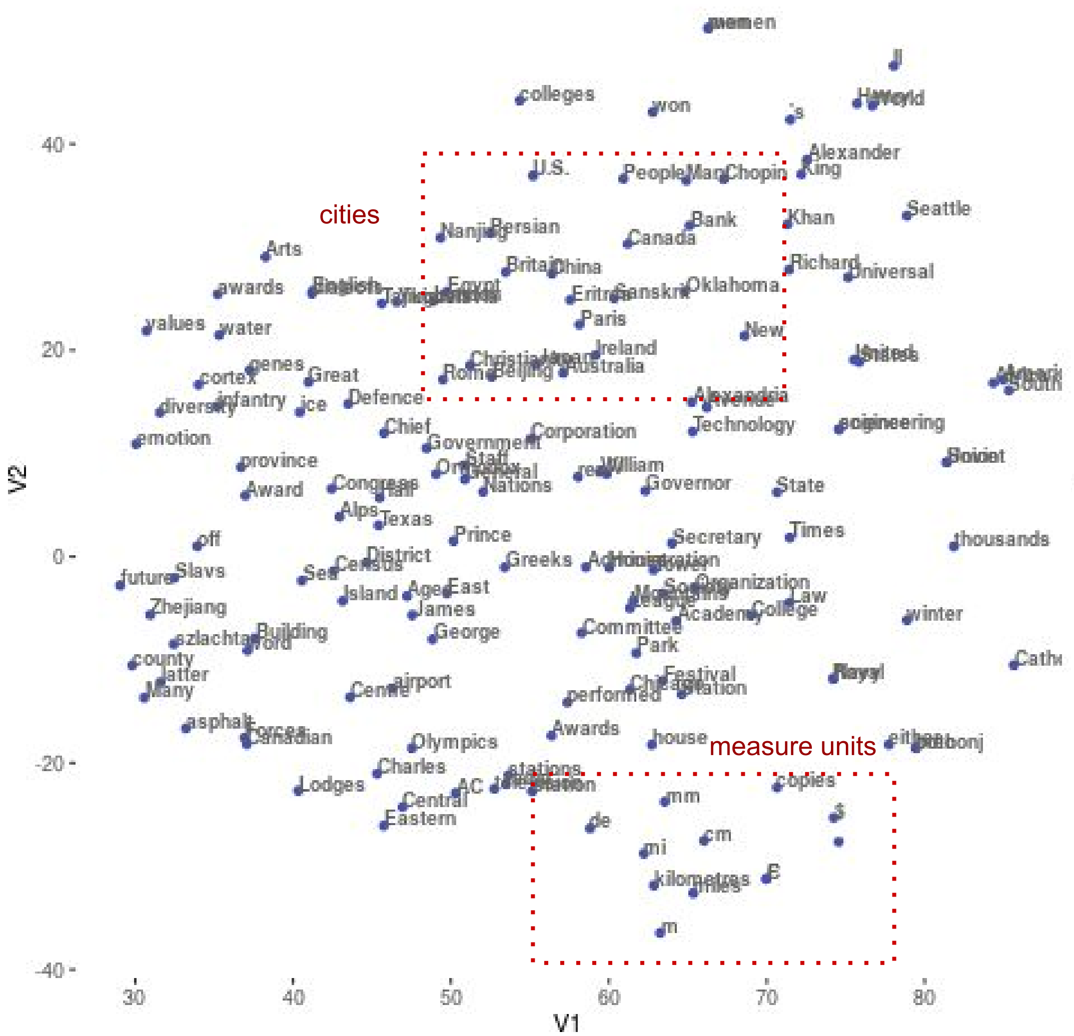
GLOVE Syntactic Embedding

Window Size = 15

Vector Size = 300

Cluster = 1

Captures different relations



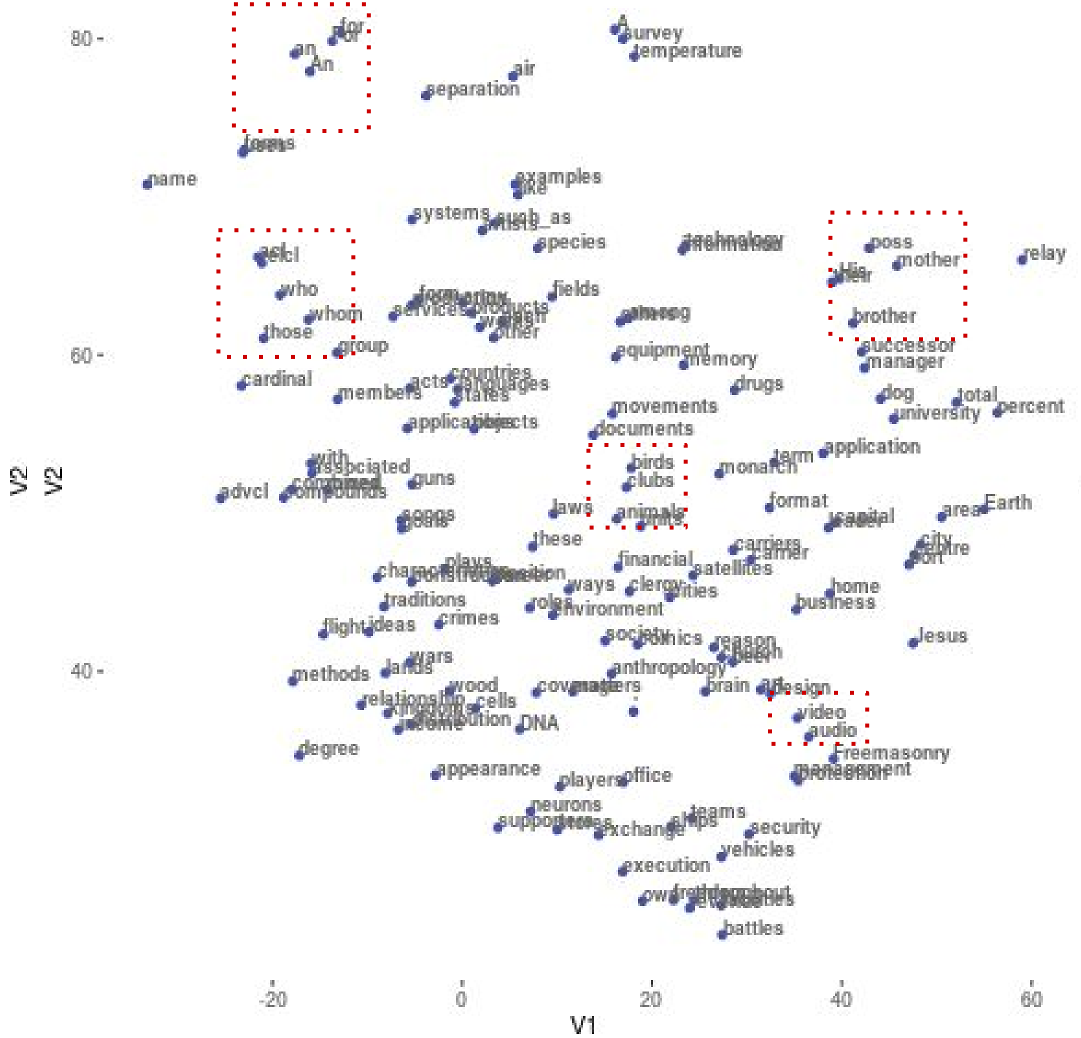
GLOVE Syntactic Embedding

Window Size = 15

Vector Size = 300

Cluster = 2

Captures different relations



GLOVE Topic Embedding

Topic 1th:

- jewish 0.022814
- jews 0.021276
- communities 0.009680
- see 0.005708
- judaism 0.005644
- orthodox 0.005516
- community 0.005324
- hebrew 0.005068
- israel 0.003658
- palestine 0.001864
- synagogue 0.001544
- persecution 0.001416
- jerusalem 0.001352
- group 0.001224
- holocaust 0.001224
- judah 0.001160

Topic 5th:

- pope 0.014170
- paul 0.008777
- john 0.006652
- cardinal 0.006597
- cardinals 0.005726
- bishops 0.005508
- athanasius 0.005344
- vi 0.005072
- rome 0.004963
- bishop 0.004309
- pius 0.003819
- see 0.003547
- vatican 0.003492
- papal 0.003056
- order 0.003002
- saint 0.002675

Topic 9th:

- economic 0.013044
- financial 0.009602
- economy 0.008634
- government 0.008365
- development 0.008311
- industry 0.007559
- public 0.007317
- world 0.005945
- trade 0.005918
- also 0.005649
- international 0.005596
- countries 0.005569
- production 0.005112
- sector 0.004762
- crisis 0.004762
- organization 0.004708

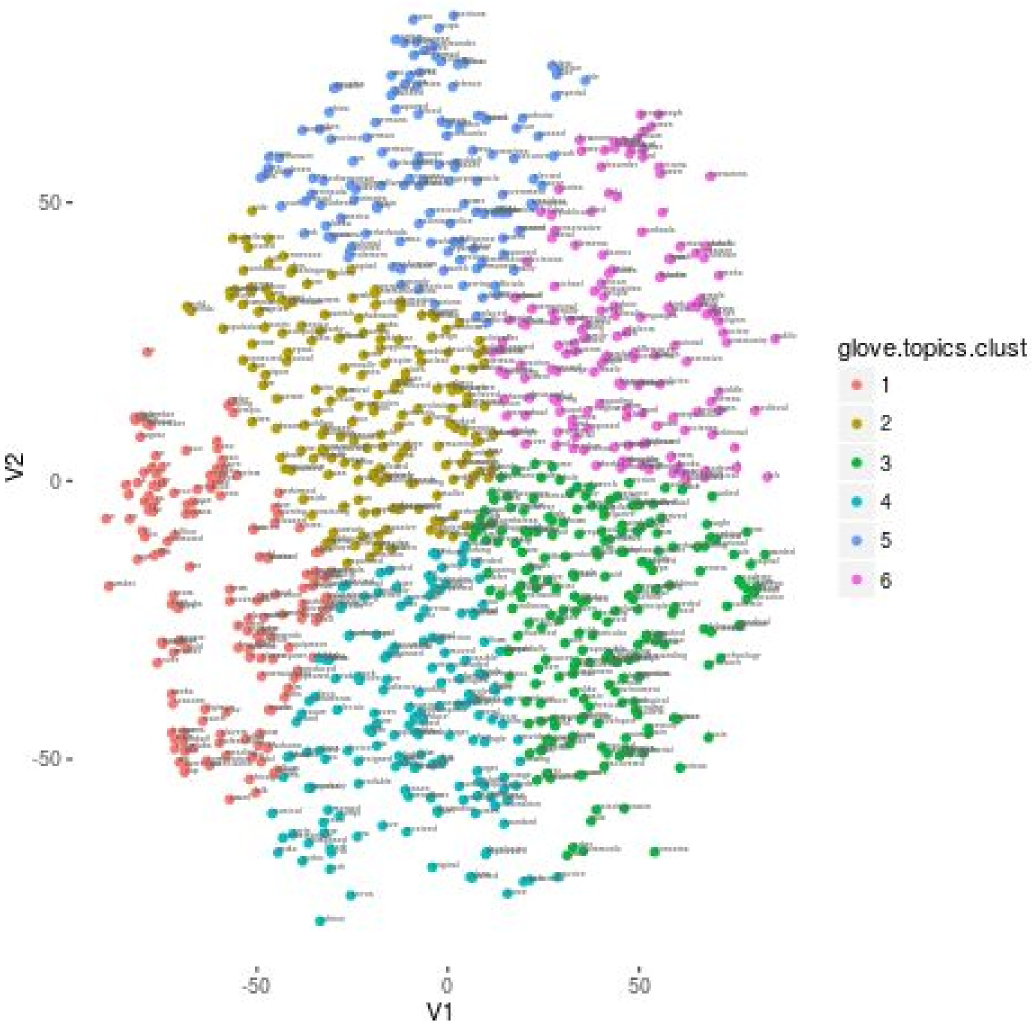
GLOVE Topic Embedding

- Min words in voc = 100
- Size of vectors = 300
- Size of window = 15

GLOVE Topic Embedding

Window Size = 15

Vector Size = 300

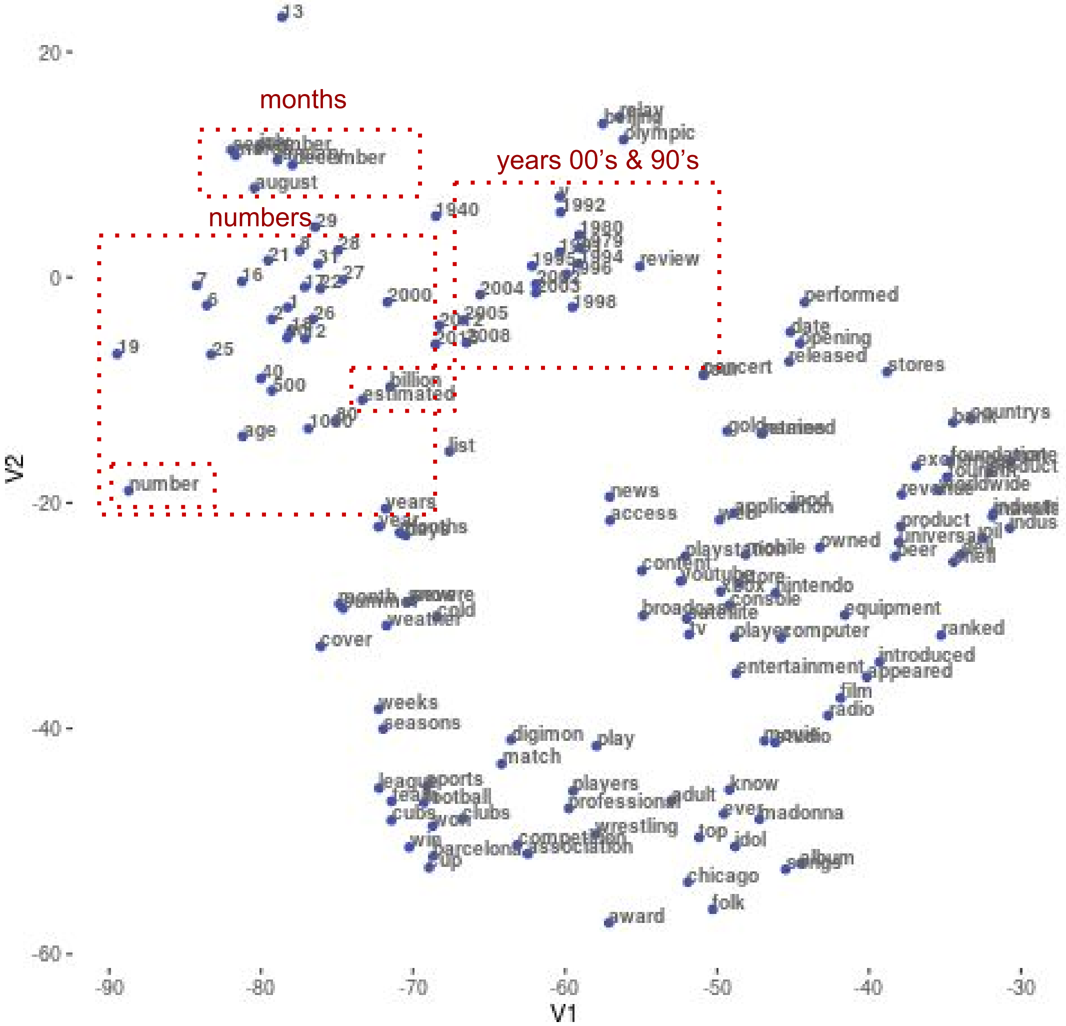


GLOVE Topic Embedding

Window Size = **15**

Vector Size = **300**

Cluster = **1**



GLOVE with topic capture

Broad topics

GLOVE Topic Embedding

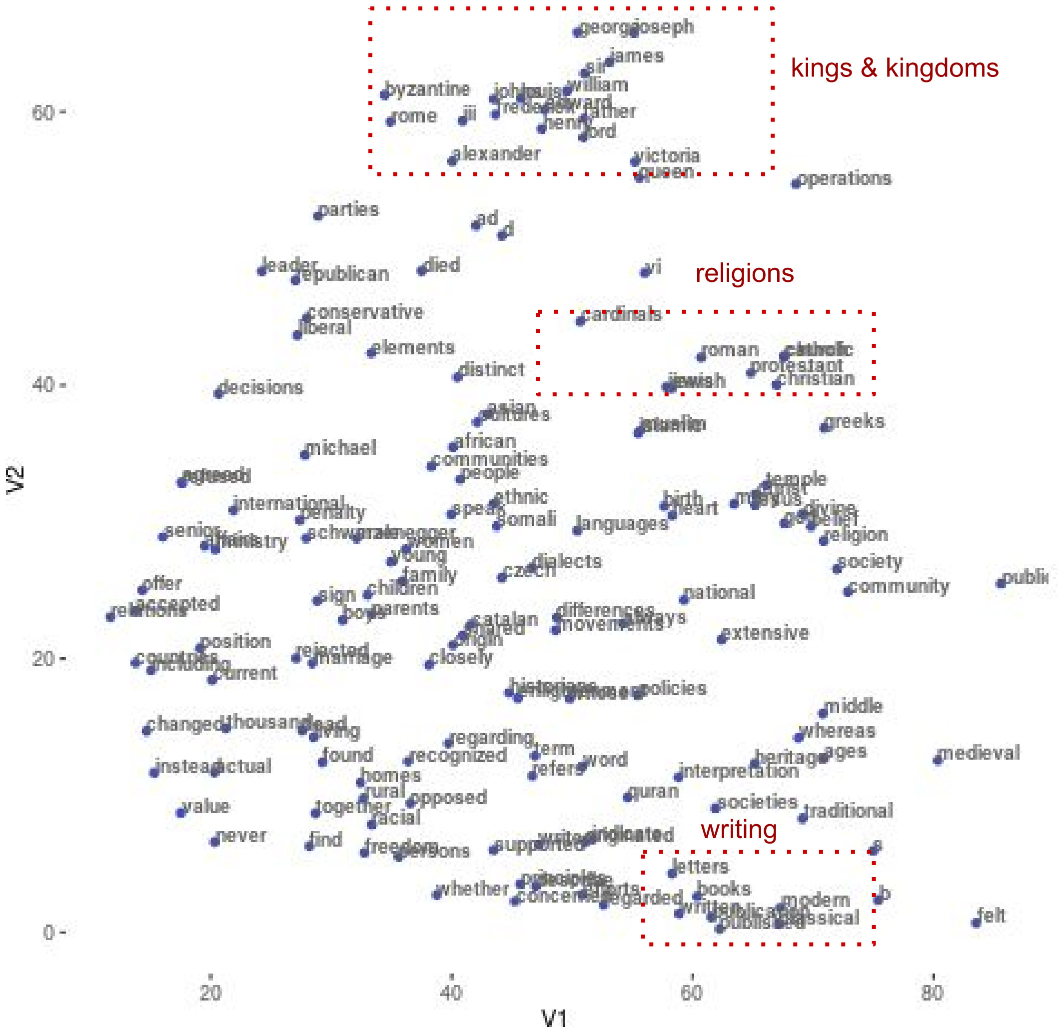
Window Size = 15

Vector Size = 300

Cluster = 6

GLOVE with topic capture

Broad topics



Exploratory analysis

- General statistics
- Lexical analysis
- Syntactic analysis
  - Embeddings
    - Word
    - Sentence
    - Paragraph

Similar words to 6W+how questions

Python’s Doc2Vec on questions

Min_count=10

Window Size = 10

Vector Size = 100

what (80% of questions)

1. which: 0.67
2. where
3. represent
4. resemble
5. supports
6. origins
7. institution
8. protect
9. formal
10. mainly

who

1. succeeded: 0.78
2. successor
3. supports
4. prevented
5. group
6. party
7. freemasons
8. criticized
9. rebel
10. toward

Similar words to 6W+how questions

how

1. there: 0.73
2. about
3. people
4. lines
5. live
6. days
7. million
8. many
9. millions
10. killed

which

1. named: 0.67
2. dominated
3. consisted
4. formed
5. mayor
6. divides
7. Somali
8. dominant
9. formerly
10. reform

Similar words to 6W+how questions

when

1. why: 0.71
2. son
3. John
4. succeeded
5. leave
6. revolution
7. richard
8. constantinople
9. ask
10. before

where

1. v

Similar words to 6W+how questions

why

1. stepper: 0.82
2. absorb
3. doing
4. mark
5. without
6. efficacy
7. genes
8. can
9. insects
10. maintain

Exploratory analysis

- General statistics
- Lexical analysis
- Syntactic analysis
  - Embeddings
    - Word
    - Sentence
    - Paragraph

Paragraph embeddings detect similarities between words

Python’s Doc2Vec on paragraphs

Min_count=10

Window Size = 10

Vector Size = 100

Synonym identification:

- sim(['college', 'professor'], ['university', 'teacher']) = 0.92
- sim(['marriage', 'husband', 'baby'], ['wife', 'wedding', 'children']) = 0.85
- sim(['house','residence','bed','accommodation','address'],['shelter','mansion ','home', 'place']) = 0.77

This analysis also detects non-related terms and analogies

Python’s Doc2Vec on paragraphs

Min_count=10

Window Size = 10

Vector Size = 100

Non-related terms identification:

- similarity('husband', 'floor') = 0.30
- similarity('night', 'chicken') = 0.29
- similarity('computer', 'city') = 0.22

Analogies

- woman is to king as man is to…? prince
- Most similar to “queen”: Madonna, widow, performed
- Most similar to “man”: said, wrote, god

The topics found with LDA can be refined using paragraph embeddings

LDA:

church + roman + first + emperor ~ history

state + govern + force + war ~ government

Most similar words to LDA keywords:

1. rome: 0.86
2. byzantine
3. centuries
4. patriarch
5. 14th
6. survived
7. 12th
8. successors
9. constantine
10. succession

Roman Empire?

1. government: 0.85
2. administration
3. sovereign
4. military
5. suppress
6. forces
7. initiated
8. supported
9. organized
10. urged

war?

The topics found with LDA can be refined using paragraph embeddings

LDA:

city + new + state + area + unit ~ nation-state

game + team + play ~ sports

Most similar words to LDA keywords:

1. located: 0.86
2. metropolitan
3. headquarters
4. county
5. designated
6. operated
7. downtown
8. currently
9. main
10. serves

metropolitan areas?

1. championship: 0.89
2. games
3. players
4. fans
5. exhibition
6. afl
7. matches
8. teams
9. nfl
10. super

championship?

The topics found with LDA can be refined using paragraph embeddings

LDA:

music + film + record ~ art

Most similar words to LDA keywords:

1. films: 0.9
2. featured
3. movie
4. studio
5. singers
6. guitar
7. songs
8. artist
9. albums
10. hip-hop

music and film recording?

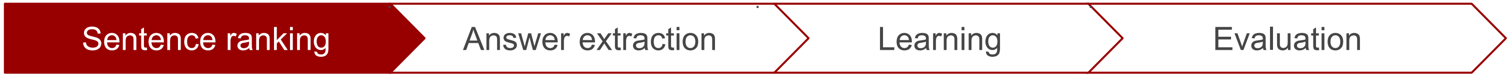
Pipeline description

High level baseline pipeline

Sentence ranking → Answer extraction → Learning → Evaluation

Sentence ranking

High level baseline pipeline



Sentence Ranking

The whole idea of sentence ranking is to exploit lexical and syntactical similarities between the question and the answer passage to obtain the sentence with the highest likelihood of being the answer.

Sentence Ranking Convolutional Neural Networks

Convolutional neural network model for reranking pairs of short texts:

- Learn optimal vector representation of Q-D
- Learn a similarity function between Q-D vectors

Sentence Ranking Convolutional Neural Networks

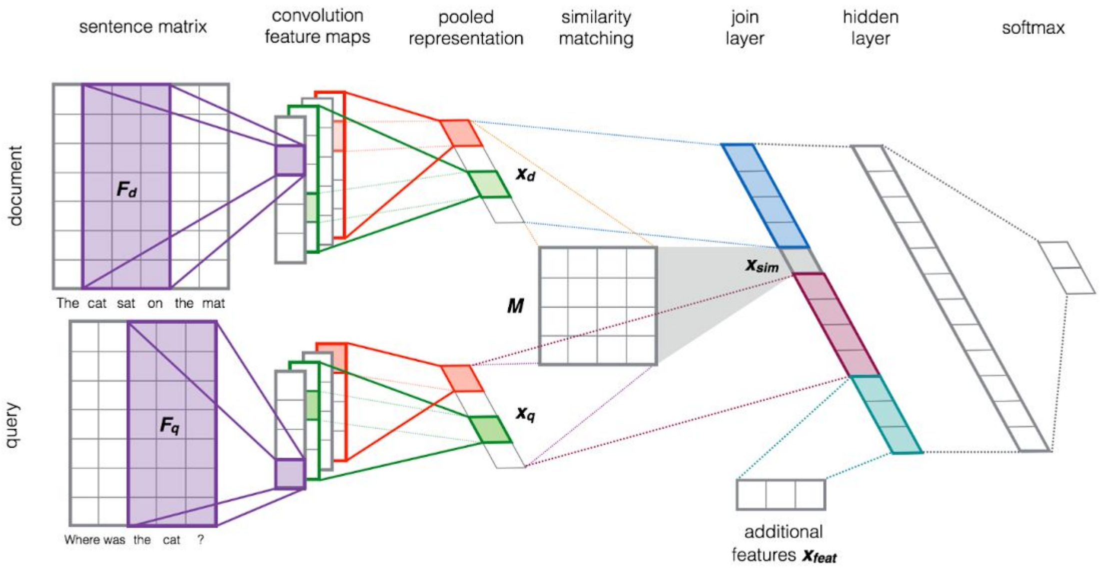


Figure 2: Our deep learning architecture for reranking short text pairs.

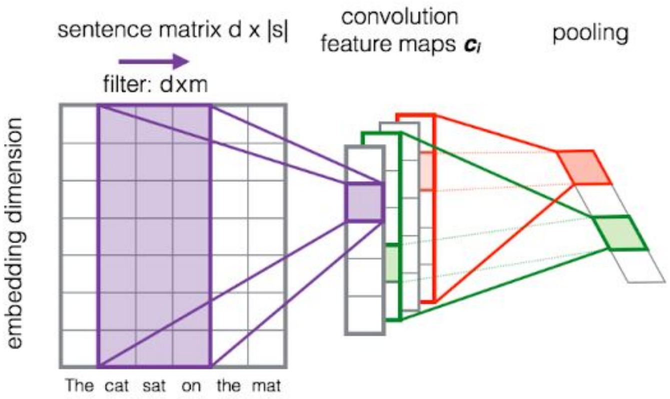
Sentence Ranking Convolutional Neural Networks

Sentences are represented as sequences of words, where each word is an |s| dimensional continuous representation.

$$\mathbf{S} = \begin{bmatrix} | & & | \\ \mathbf{w}_1 & \dots & \mathbf{w}_{|s|} \\ | & & | \end{bmatrix}$$

A filter f is applied to the sequence in order to capture interactions among words.

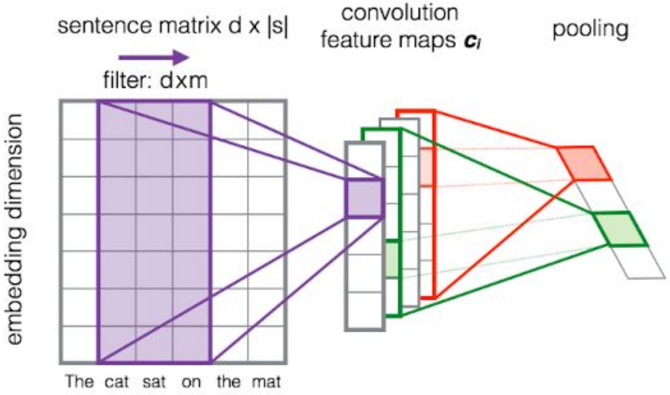
$$\mathbf{c}_i = (\mathbf{s} * \mathbf{f})_i = \mathbf{s}^T_{[i-m+1:i]} \cdot \mathbf{f} = \sum_{k=i}^{i+m-1} s_k f_k$$



Sentence Ranking Convolutional Neural Networks

After this is done, a nonlinear activation function, ReLU in this case, is applied to every c_i and the results are pooled together via max pooling into a single c_{pooled} array representation.

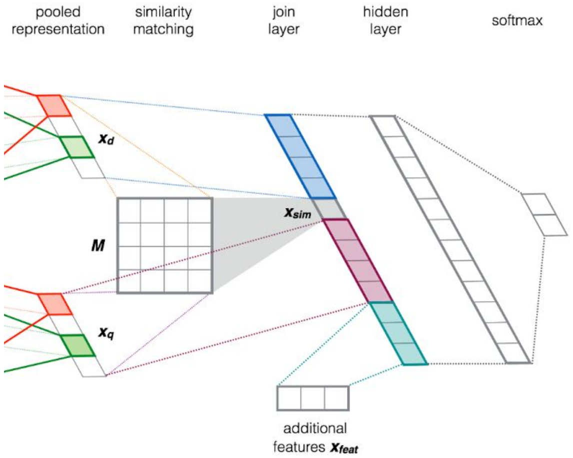
$$\mathbf{c}_{\text{pooled}} = \begin{bmatrix} \text{pool}(\alpha(\mathbf{c}_1 + b_1 * \mathbf{e})) \\ \cdots \\ \text{pool}(\alpha(\mathbf{c}_n + b_n * \mathbf{e})) \end{bmatrix}$$



Sentence Ranking Convolutional Neural Networks

Once these representations are obtained for each sentence x_d and each query x_q, a x_{sim} score is obtained by x_d'Mx_q and an x_{join} is created by simple concatenation.

Each x_{join} is passed through a hidden layer to exploit interactions among its different components, and finally a softmax is used for the ranking.



Sentence Ranking Convolutional Neural Networks

The model is trained to minimize the cross-entropy function:

$$\begin{aligned}\mathcal{C} &= -\log \prod_{i=1}^{N} p(y_i|\mathbf{q}_i, \mathbf{d}_i) + \lambda\|\theta\|_2^2 \\ &= -\sum_{i=1}^{N} [y_i \log \mathbf{a}_i + (1 - y_i)\log(1 - \mathbf{a}_i)] + \lambda\|\theta\|_2^d,\end{aligned}$$

where a is the output of the softmax and θ contains all the parameters of the network:

$$\theta = \{\mathbf{W}; \mathbf{F}_q; \mathbf{b}_q; \mathbf{F}_d; \mathbf{b}_d; \mathbf{M}; \mathbf{w}_h; b_h; \mathbf{w}_s; b_s\}$$

Regularization is used to avoid overfitting and stochastic gradient descent to cope with the non convexity of the problem.

Sentence Ranking Convolutional Neural Networks

In our model, we added an hybrid vector representation that used both, the representation trained over the AQUAINT corpus (to obtain the most general context of each word), and over the SQUAD dataset (to obtain the particular uses of each word). We also used Jaccard similarity as a proxy of relevance judgments, and we added topic information to the x_{joint} representation.

Sentence Ranking BM25 & Jaccard similarity

Another approach that uses only lexical similarity, under the bag of words was applied, namely BM25 and Jaccard similarity:

- BM25

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

$$\text{IDF}(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5},$$

- Jaccard similarity

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Sentence Ranking Results

MODEL	MRR Score
Convolutional Neural Networks	.25
BM25	.71
Jaccard	.76

We believe that the bad results of the ConvNets is due to underfitting of the training data.

High level baseline pipeline

Sentence ranking | Answer extraction | Learning | Evaluation

Answer extraction

Answer extraction Random Forests

Idea: Use *features* that extract lexical, syntactical and semantical structure of sentence, question and answer to train a classifier.

For each word in candidate answer sentence:

- indicator of right neighbor in question
- right neighbor NER
- right neighbor POS
- word Animacy
- word Gender
- word NER
- word Number
- word POS
- word type

- dependency with father
- indicator father in question
- father NER
- father POS
- indicator of word in question
- indicator of left neighbor in question
- left neighbor NER
- left neighbor POS
- question type

Answer extraction Random Forests

Example: It

(False, u'It', u'PRP', u'O', False, 'whom', '', '', '', '', '', u'is', u'VBZ', u'O', False, False, u'replica', u'NN', u'O', u'nsubj', False, False, u'INANIMATE', u'SINGULAR', u'NEUTRAL', u'PRONOMINAL')

Answer extraction Random Forests

Model

- 1000 trees
- 5 variables per cut
- Gini criterion

Results

Training:

F1 score = .49

precision = .62

Test:

F1 score = .47

precision = .6

Answer extraction Random Forests


```
"56d601e41c85041400946ed0": "sacked him seven times and",
"56d601e41c85041400946ed1": "Bowl 50 and",
"56d601e41c85041400946ed2": "tackles 21/2 sacks",
"56d98b33dc89441400fdb53b": "him seven times",
"56d98b33dc89441400fdb53c": "Bowl 50 and",
"56d98b33dc89441400fdb53d": "two",
"56d98b33dc89441400fdb53e": "tackles 21/2 sacks",
"56be5333acb8001400a5030a": "Bowl 50 in",
"56be5333acb8001400a5030b": "of 5 million",
"56be5333acb8001400a5030c": "and Bruno Mars who headlined",
"56be5333acb8001400a5030d": "Bruno Mars who",
"56be5333acb8001400a5030e": "Bruno Mars who",
"56beaf5e3aeaaa14008c91fd": "50",
"56beaf5e3aeaaa14008c91fe": "average of 5 million for",
"56beaf5e3aeaaa14008c91ff": "Bruno Mars who",
"56beaf5e3aeaaa14008c9200": "Mars",
"56beaf5e3aeaaa14008c9201": "thirdmost",
"56bf1ae93aeaaa14008c951b": "Bowl 50 in",
"56bf1ae93aeaaa14008c951c": "of 5 million",
"56bf1ae93aeaaa14008c951e": "Bruno Mars who",
"56bf1ae93aeaaa14008c951f": "Bruno Mars who",
"56d2051ce7d4791d00902608": "of 5 million",
"56d2051ce7d4791d00902609": "of 5 million",
"56d2051ce7d4791d0090260a": "Bruno Mars who",
"56d2051ce7d4791d0090260b": "and Bruno Mars who headlined",
"56d602631c85041400946ed8": "50",
"56d602631c85041400946eda": "Bruno Mars who"
```

Pipeline implementation

Pipeline implementation

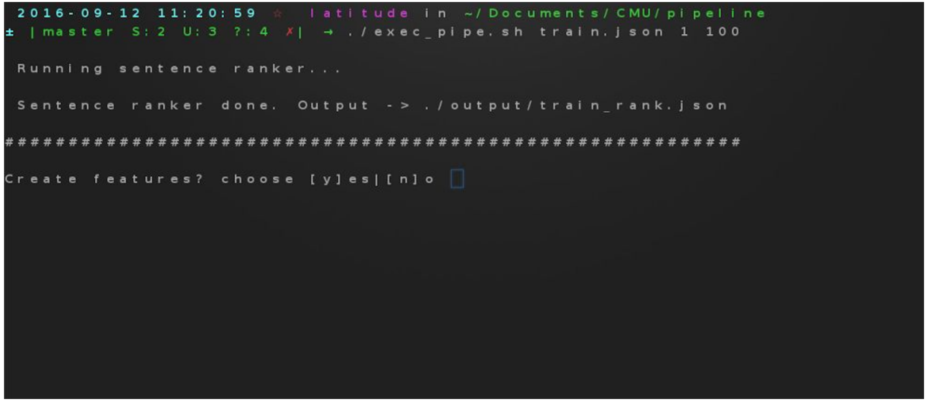
Our pipeline implementation supports:

- An end to end pipelined execution.
- Model training
- Model testing
- Interactive Mode

Pipeline implementation

Model training:

The system allows you to choose the number of sentences to be considered as part of the answer as well as the number of instances used on the training phase.



Pipeline implementation

Model testing:

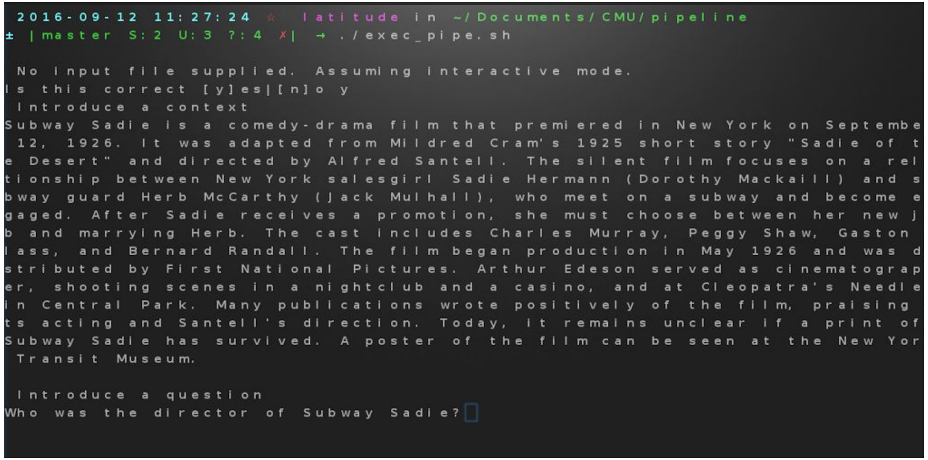
It also gives you the option to train or test the model. And provides a final evaluation with Stanford’s script.

```
##########################################################

Create features? choose [y]es|[n]o n

Do you want to t[r]ain|t[e]st? e
##########################################################

 Running answer extractor...

##########################################################

 Answer extraction test done. Output -> ./output/predicts.json

 Stanford's test format. Output -> ./output/test_stanford.json


 Stanford's evaluation:

{"f1": 0.20952380952380956, "exact_match": 0.08490566037735849}

 Done!!!
```

Pipeline implementation

Model interactive mode:

Finally, to enable testing of new models, the system also supports interactive mode.



Pipeline implementation

Model interactive mode:

Finally, to enable testing of new models, the system also supports interactive mode.

```
##########################################################

 Running feature constructor ...
/usr/lib/python2.7/dist-packages/cffi/model.py:526: UserWarning: 'point_con
version_form_t' has no values explicitly defined; next version will refuse
to guess which integer type it is meant to be (unsigned/signed, int/long)
  % self._get_c_name())
0

 Feature extractor done. Output -> ./output/interact_features.json

##########################################################

 Running answer extractor...

##########################################################

 The Answer is:

[1] "Sadie"

 Done!!!
```

Pipeline implementation

End to end execution results evaluated under Stanford's metric:

{"f1": 0.20368373764600187, "exact_match": 0.07547169811320754}

	Exact Match		F1	
	Dev	Test	Dev	Test
Random Guess	1.1%	1.3%	4.1%	4.3%
Sliding Window	13.2%	12.5%	20.2%	19.7%
Sliding Win. + Dist.	13.3%	13.0%	20.2%	20.0%
Logistic Regression	40.0%	40.4%	51.0%	51.0%
Human	80.3%	77.0%	90.5%	86.8%

References

- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR, 2013.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of NIPS, 2013.
- Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic Regularities in Continuous Space Word Representations. In Proceedings of NAACL HLT, 2013.
- Le, Quoc V., and Tomas Mikolov. "Distributed Representations of Sentences and Documents." ICML. Vol. 14. 2014.
- Tomas Mikolov, Quoc V. Le and Ilya Sutskever.Exploiting Similarities among Languages for Machine Translation.

References

- Levy, Omer, and Yoav Goldberg. "Dependency-Based Word Embeddings." ACL (2). 2014.
- Levy, Omer, Yoav Goldberg, and Ido Dagan. "Improving distributional similarity with lessons learned from word embeddings." Transactions of the Association for Computational Linguistics 3 (2015): 211-225.
- Levy, Omer, Yoav Goldberg, and Israel Ramat-Gan. "Linguistic Regularities in Sparse and Explicit Word Representations." CoNLL. 2014.
- L.J.P. van der Maaten and G.E. Hinton. **Visualizing High-Dimensional Data Using t-SNE**. *Journal of Machine Learning Research* 9(Nov):2579-2605, 2008.
- Rajpurkar, Pranav, et al. "SQuAD: 100,000+ Questions for Machine Comprehension of Text." *arXiv preprint arXiv:1606.05250* (2016).

References

- Jolliffe, Ian. *Principal component analysis*. John Wiley & Sons, Ltd, 2002.
- Blei, David M., and John D. Lafferty. "Topic models." *Text mining: classification, clustering, and applications* 10.71 (2009): 34.
- Severyn, Aliaksei, and Alessandro Moschitti. "Learning to rank short text pairs with convolutional deep neural networks." *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2015.