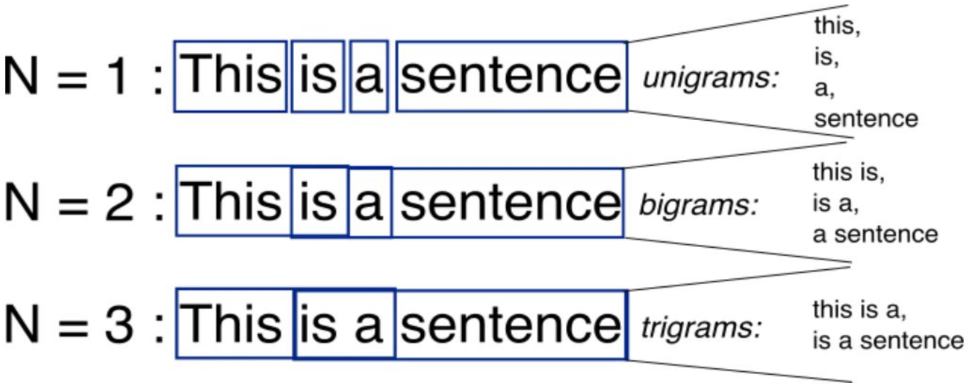
Efficient Estimation of Word Representations in Vector Space

The paper behind word2vec

Andris Oueslati, Marius Boucaut, Yuxian Zuo

N-gram model

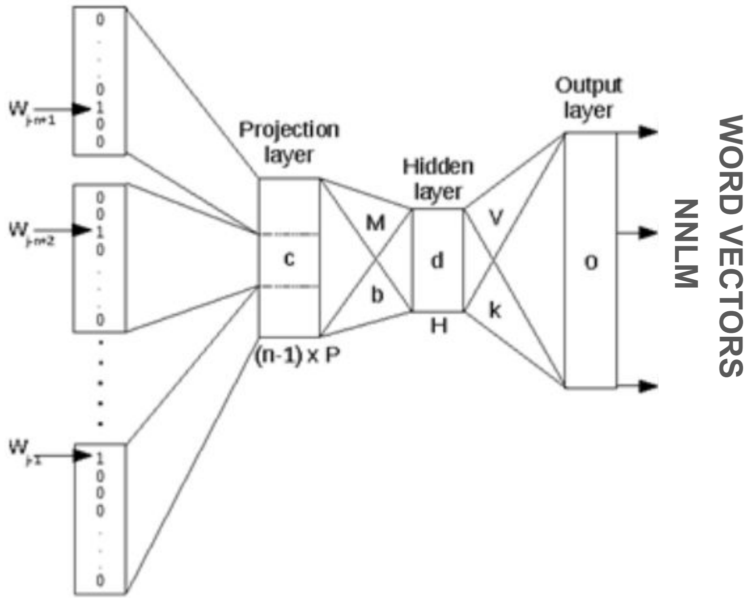
text → Sequence of N items

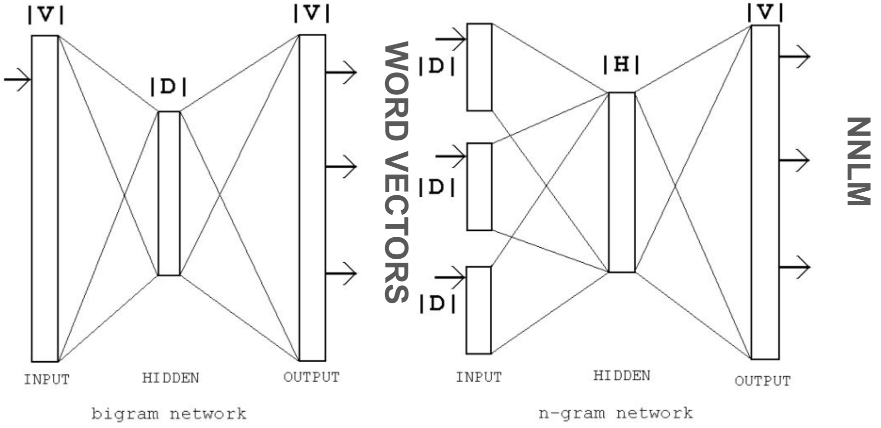


- Lacks context information for smaller N, computationally expensive for large N
- Zero-frequency problem
- Words are discrete units : no relationships or interactions within a sentence, no polysemy
- Relies on large datasets (lot’s of words) to achieve decent performance.

Outperformed by neural network based models

Neural network language model



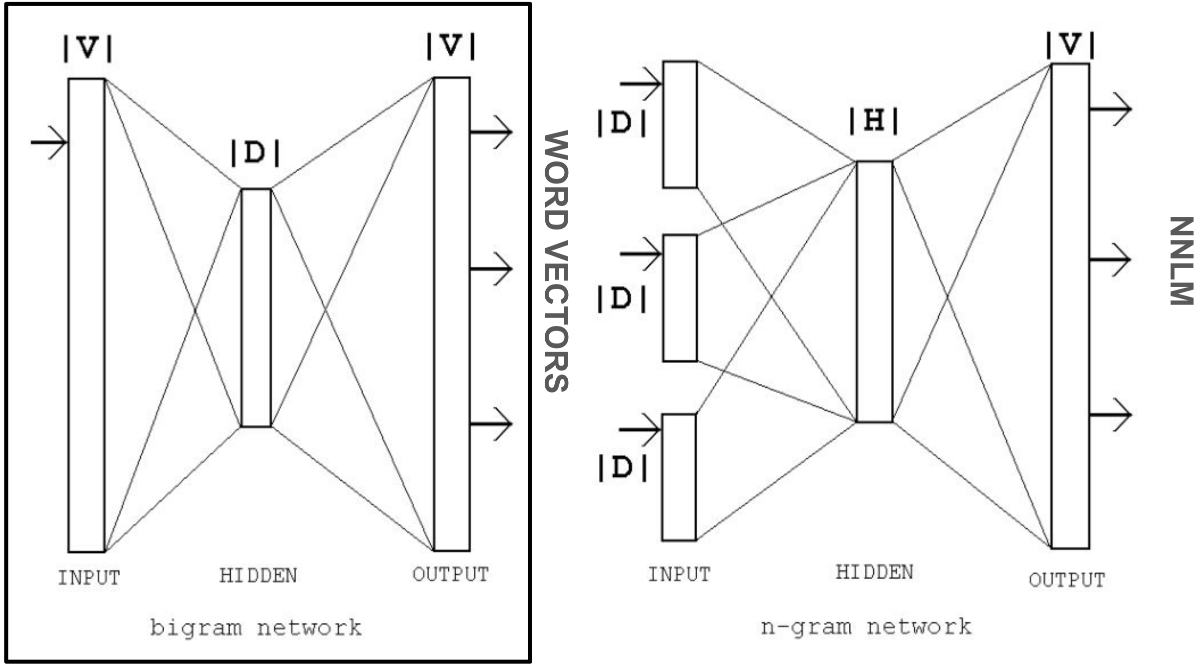


The word vector representation and a statistical language model is learnt at the same time. The vectors are optimized specifically for the task.

The word vectors are first learned using neural network with a single hidden layer. The word vectors are then used to train the NNLM.

Neural network language model

How the word vectors are learned can
impact the performance of the NNLM



How do we compare models?

We search for :

- The lowest the computational complexity

  = number of parameters used in the fully trained model

- The lowest training complexity

  = iterations E x number of words T x computational complexity per training Q

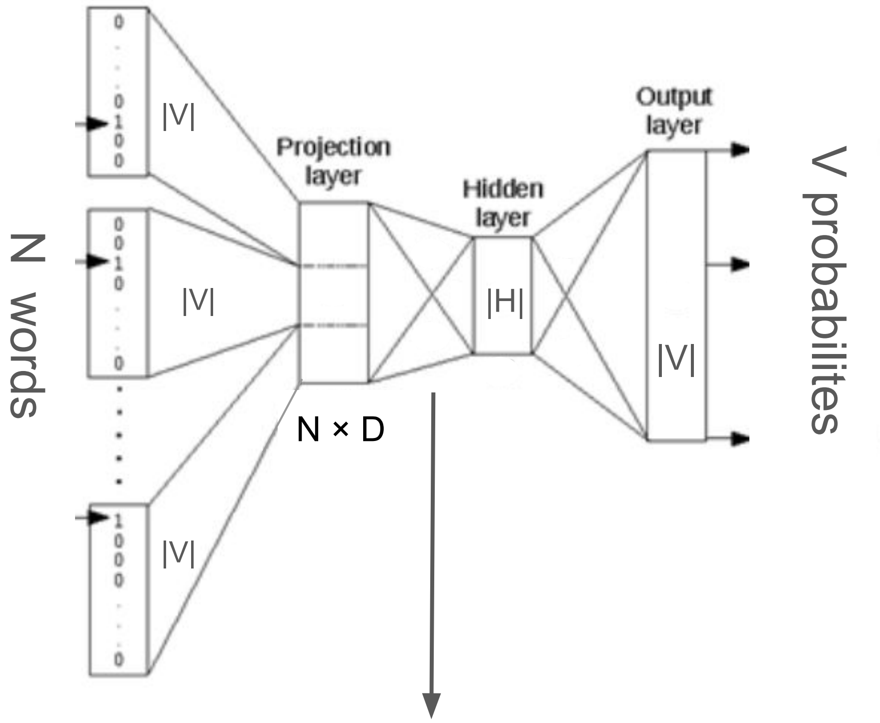
- The highest accuracy

Feedforward Neural Net Language Model

|V| = Vocabulary size

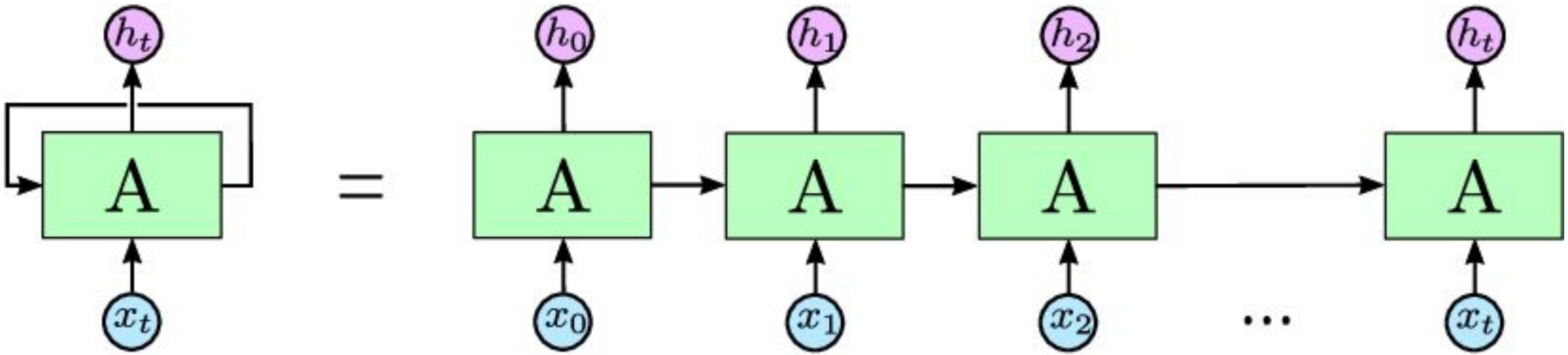
Vocabulary = binary tree

N = 10
P ∈ [500, 2000]
H ∈ [500, 1000]



Q ~ N x D x H

Recurrent Neural Network Language Model

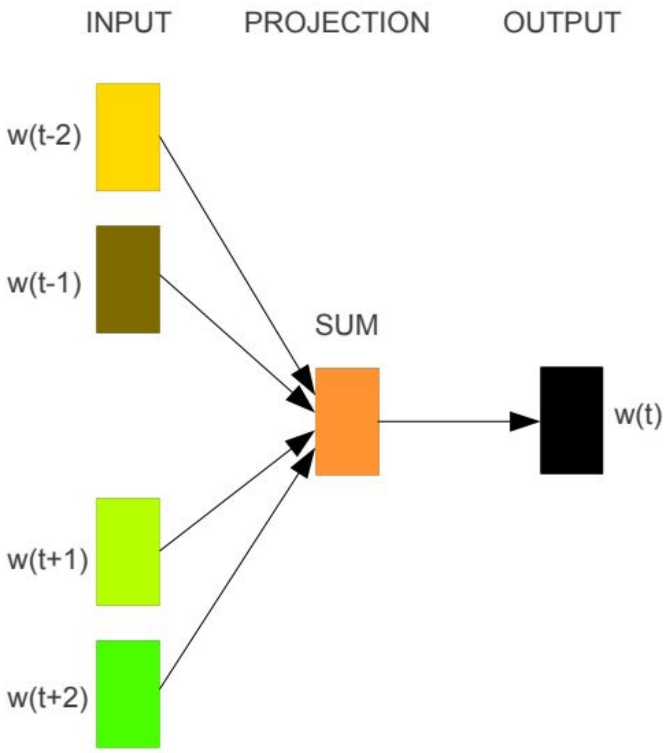


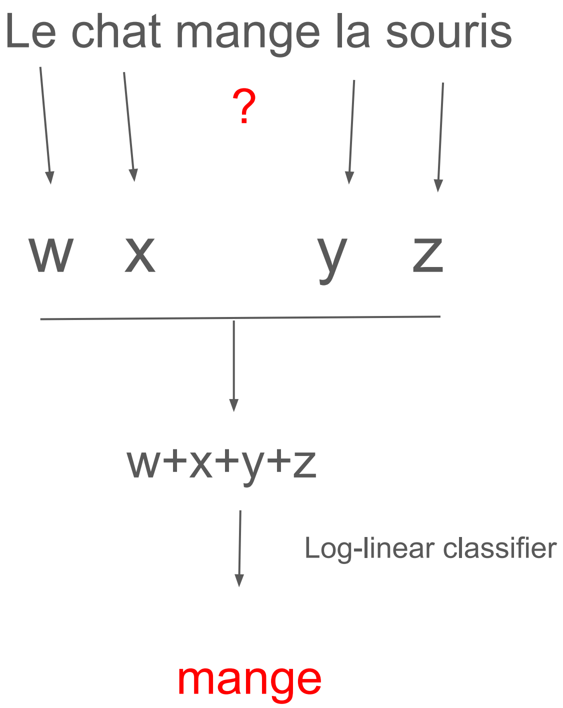
Reduce complexity to train more

Feedforward Neural Net Language Model	Recurrent Neural Net Language Model
$Q = N \times D \times H$	$Q = H \times H$

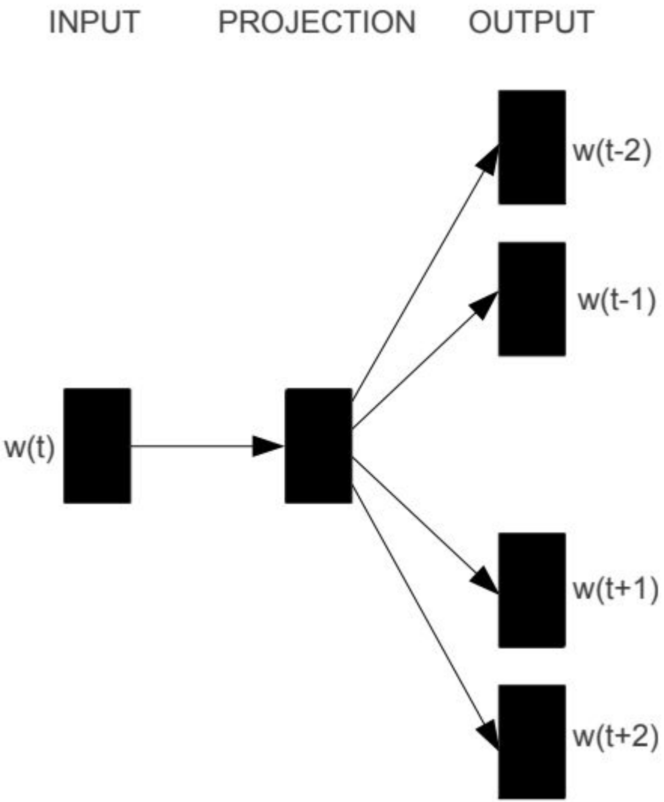
Hidden layer ~~Not so much~~ Complexity → Heavy training

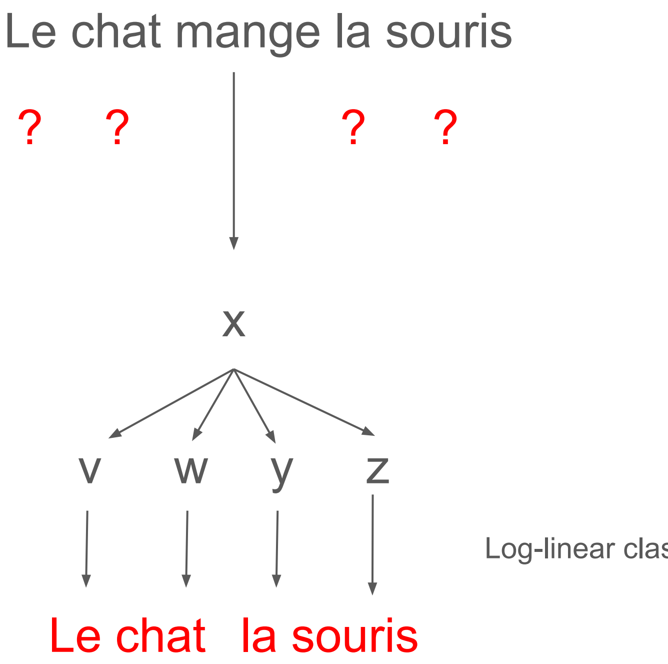
Continuous Bag-of-Words Model - **CBOW**





Skip-gram


INPUT
PROJECTION
OUTPUT
w(t)
w(t-2)
w(t-1)
w(t+1)
w(t+2)


Le chat mange la souris
? ? ? ?
x
v w y z
Le chat la souris
Log-linear clas

Results - Word Similarity

“What is the word that is similar to small in the same sense as biggest is similar to big?”

$$X = vector(''biggest'') - vector(''big'') + vector(''small'')$$

- 5 types of semantic questions, and 9 types of syntactic questions
- Overall, there are **8869** semantic and **10675** syntactic questions

- Scored about **60%** (assuming exact match, i.e., synonyms are counted as mistakes)

Results - Word Similarity

Table 1: *Examples of five types of semantic and nine types of syntactic questions in the Semantic-Syntactic Word Relationship test set.*

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Results - Maximization of Accuracy

Table 2: *Accuracy on subset of the Semantic-Syntactic Word Relationship test set, using word vectors from the CBOW architecture with limited vocabulary. Only questions containing words from the most frequent 30k words are used.*

Dimensionality / Training words	24M	49M	98M	196M	391M	783M
50	13.4	15.7	18.6	19.1	22.5	23.2
100	19.4	23.1	27.8	28.7	33.4	32.2
300	23.2	29.2	35.3	38.6	43.7	45.9
600	24.0	30.1	36.5	40.8	46.6	50.4

- After some point, adding more dimensions or adding more training data provides diminishing improvements.

Results - Comparison of Model Architectures

Table 3: *Comparison of architectures using models trained on the same data, with 640-dimensional word vectors. The accuracies are reported on our Semantic-Syntactic Word Relationship test set, and on the syntactic relationship test set of [20]*

Model Architecture	Semantic-Syntactic Word Relationship test set		MSR Word Relatedness Test Set [20]
	Semantic Accuracy [%]	Syntactic Accuracy [%]	
RNNLM	9	36	35
NNLM	23	53	47
CBOW	24	64	61
Skip-gram	55	59	56

- **NNLM:** Outperforms RNN in both syntactic and semantic tasks.
- **CBOW:** Excels in syntactic tasks, performing better than NNLM.
- **Skip-gram:** Slightly weaker in syntactic tasks compared to CBOW but significantly better in semantic tasks than all other models.

Results - Large Scale Parallel Training of Models

- Trained using one CPU only, the CBOW model required about **one day**, while the Skip-gram model took approximately **three days** to train on the same dataset.
- However, when models were implemented using the distributed framework **DistBelief**, CPU usage for CBOW and Skip-gram models became similar..

Table 6: *Comparison of models trained using the DistBelief distributed framework. Note that training of NNLM with 1000-dimensional vectors would take too long to complete.*

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days x CPU cores]
			Semantic	Syntactic	Total	
NNLM	100	6B	34.2	64.5	50.8	14 x 180
CBOW	1000	6B	57.3	68.9	63.7	2 x 140
Skip-gram	1000	6B	66.1	65.1	65.6	2.5 x 125

Results - Microsoft Sentence Completion Challenge

The Microsoft Sentence Completion Challenge evaluates language modeling by ***completing 1040 sentences with one missing word from five choices.***

Table 7: *Comparison and combination of models on the Microsoft Sentence Completion Challenge.*

Architecture	Accuracy [%]
4-gram [32]	39
Average LSA similarity [32]	49
Log-bilinear model [24]	54.8
RNNLMs [19]	55.4
Skip-gram	48.0
Skip-gram + RNNLMs	**58.9**

- **Combined Skip-gram and RNNLM** scores achieved a new state-of-the-art accuracy of 58.9% (59.2% on the development set and 58.7% on the test set).

Conclusion

- Simple model architectures can efficiently train high-quality word vectors with lower computational complexity compared to neural networks.
- CBOW and Skip-gram models can process massive datasets (e.g., corpora with one trillion words) using the *DistBelief distributed framework.*
- Neural network-based word vectors have been used in tasks like sentiment analysis and paraphrase detection. It can be expected that these applications can benefit from the architectures described in this paper.
- In the future, it would be also interesting to compare these techniques to Latent Relational Analysis and others.
- High-quality word vectors are expected to be fundamental for advancing NLP applications.