Neural Probabilistic Language Models (NPLM)

Alvaro Soto

Computer Science Department (DCC), PUC

Language Model (LM)

- A model to quantify the co-occurrence of sequence of words in text.
- Ex. assign a score (usually a probability) to the word sequence: "This is a class presentation".

- By assigning scores or probabilities to sequences of words (sentences), a LM provides a method to distinguish between probable and improbable sentences.
- Many applications: semantic role labeling, sentiment analysis, named entity recognition, language translation, text parsing, and so on.

Probabilistic Language Model (PLM) (aka Statistical Language Model)

- A LM that uses a probability distribution to quantify word co-occurrence.
- Ex. n-grams models.

Neural Probabilistic Language Model (NPLM)

- A PLM that uses a neural network to obtain the probability of each word sequence.
- Ex. multilayer perceptron (feedforward network).

n-grams

- n-gram: a sequence of n consecutive words. Depending on the value of n, we have: bigrams, trigrams, 4-grams, 5-grams, etc.
- Goal is to access: $P(w_t, w_{t-1}, w_{t-2}, ..., w_{t-n+1})$.
- Problem: longer n-grams require more training data, why?
- Markov assumption limits the relevant history. Ex. for a horizont of n previous words, $P(w_t|w_{t-1:1}) = P(w_t|w_{t-1:t-n+1})$.
- Using Markov assumption and chain rule, we can obtain the prob. of sentences of any length, how? or any marginal like $P(w_t|w_{t-1}, w_{t-2}, ..., w_{t-n+1})$, how?
- Usually, n-gram models can be trained by using frequency counts on large text corpus (multinomial models).

n-grams

Pros

- Simple.
- Reasonable performance.

Cons

- Data sparsity is a big problem to estimate probs. World of n-grams is sparse: Zipfian.
- Also, number of parameters grows exponentially with context size (n-grams length).
- The use of a discrete representation can't generalize properly to unseen instances, i.e., rare or less frequent n-grams.

Discrete representations

- An n-gram model estimates a world model using a discrete distribution (multinomial).
- A main problem of discrete distributions is that all discrete values are equally similar (or dissimilar), there is not a notion of neighborhood.
- As a consequence, there is no simple way to do smoothing based on input similarity. Words like "Restaurant" and "Coffe-Shop" are just "concepts", but not "closely related concepts".
- There are previous works on n-gram smoothing, but so far they do not offer a sound solution.

Continuous representations

- Continuous representations provide a notion of neighborhood and similarity.
- In particular, NPLM models use a continuous vector representation for each word in the vocabulary.
- A NPLM can use training instances such as:

  "I will go to play soccer"

  to obtain information about semantically similar, but less frequent training instances, such as:

  "I will go to play handball"

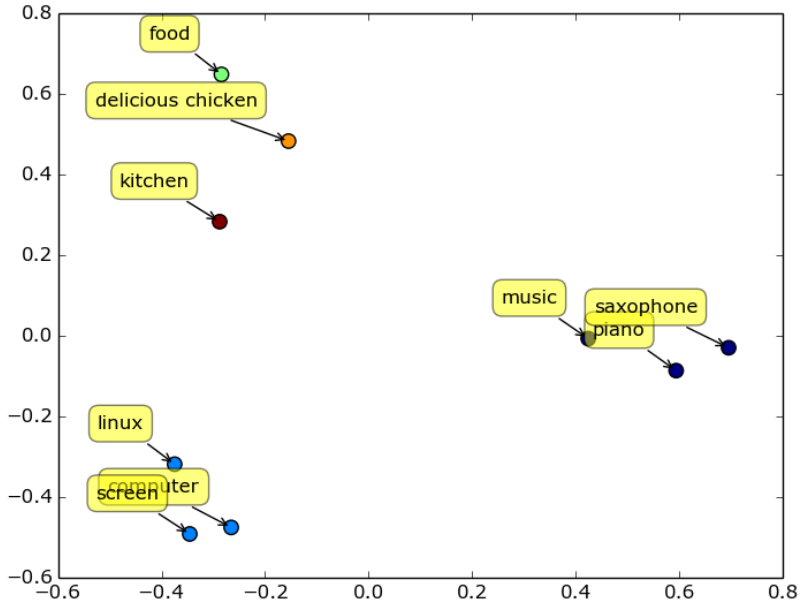
Guiding idea is the "(Semantic) Distributional Hypothesis" (Harris, 1954):

Words in similar contexts have similar meanings.

Distributed Representations

Distributed Representations

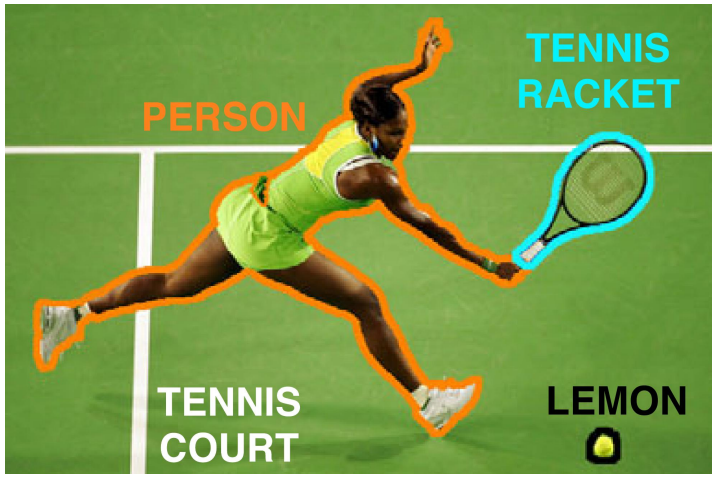
As a consequence, words in similar contexts will have similar representation, i.e., a similar (close) vector embedding.



Guiding idea is the "(Semantic) Distributional Hypothesis" (Harris, 1954):

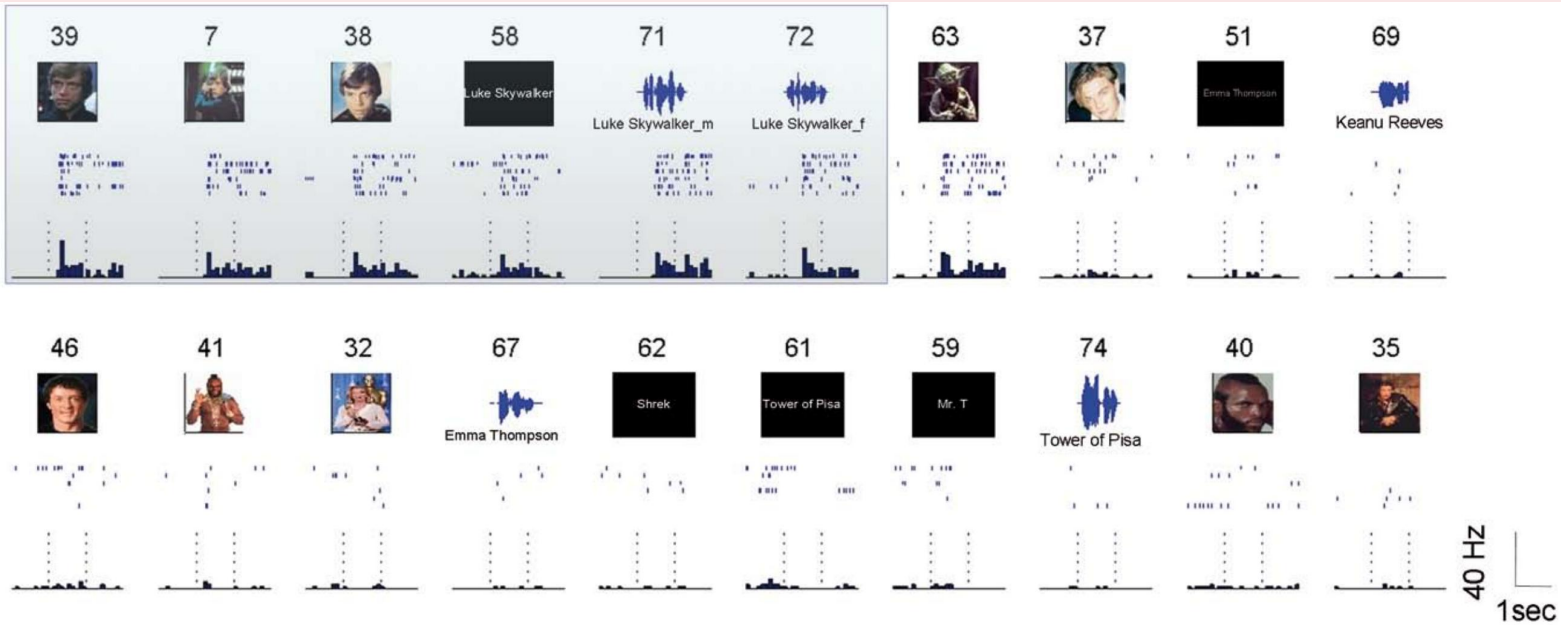
Words in similar contexts have similar meanings.

Same principle can be applied to several situations:



Key idea: contextual information determine meaning.

Welcome to one of the secret of most (not all) human intelligence:



Neural Probabilistic Language Model (NPLM)

- Y. Bengio et al. (NIPS, 2000 and JMLR, 2003) propose one of the first NPLM.
- They use a feedforward NN to model the word predicted probability: $P(w_t|w_{t-1}, w_{t-2}, ..., w_{t-n+1})$, i.e., the probability of the last word given the (n-1) previous ones, where n is a parameter.

Key Idea 1: by using word-to-context relations, this approach is able to use supervised machine learning techniques to solve an unsupervised problem.

Key Advantage 1: learning can take advantage of huge amounts of public documents, which can produce impressive results.

- Y. Bengio et al. (NIPS, 2000 and JMLR, 2003) propose one of the first NPLM.
- They use a feedforward NN to model the word predicted probability: $P(w_t|w_{t-1}, w_{t-2}, ..., w_{t-n+1})$, i.e., the probability of the last word given the (n-1) previous ones, where n is a parameter.

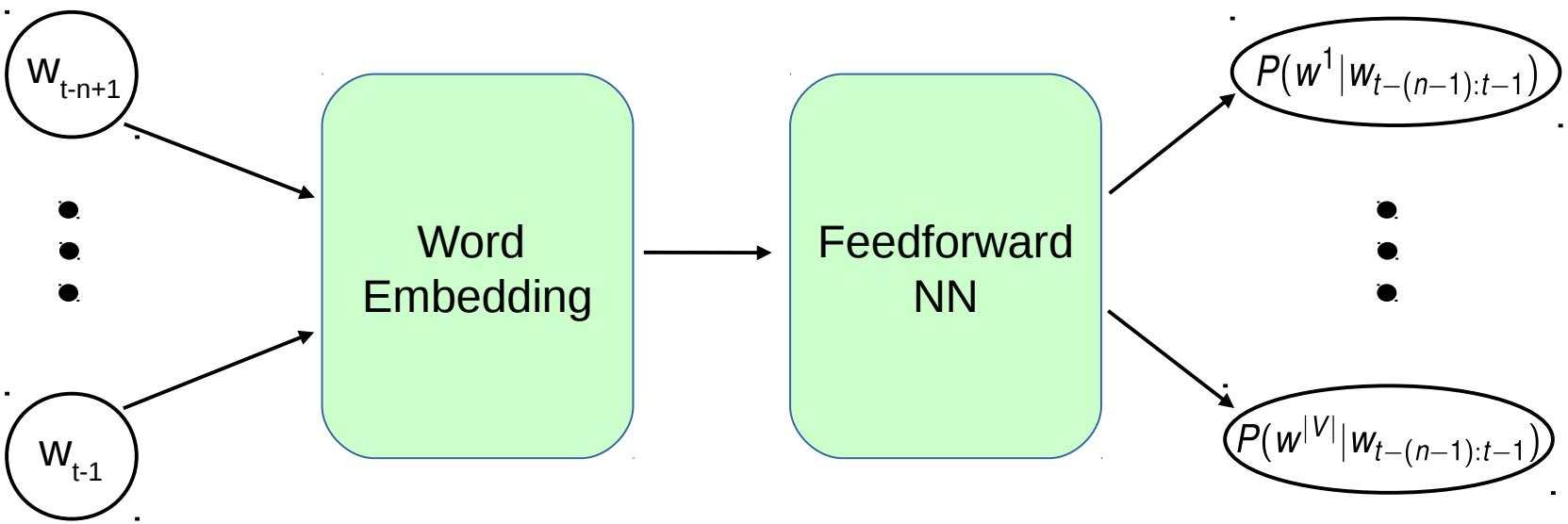
Key Idea 2: by using a continuous representation, the model can learn local similarity relations.

Key Advantage 2: parts of the target space that are highly represented in the training data (frequent patterns) can transfer knowledge to poorly represented areas (long tail).

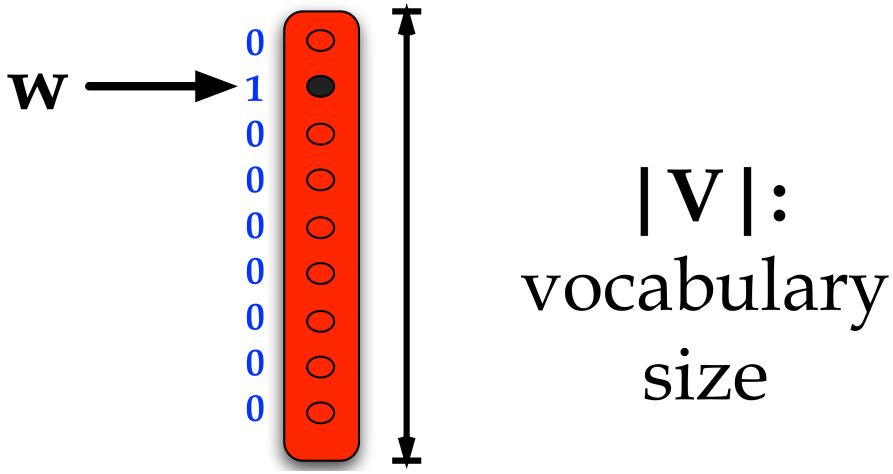
NPLM: Two main goals

- Goal1: Learn a continuous representation for each word w in the vocabulary, i.e., learn a word embedding.
- Goal 2: Learn a model to predict a word w given its context h(w), i.e., learn P(w|h(w)).

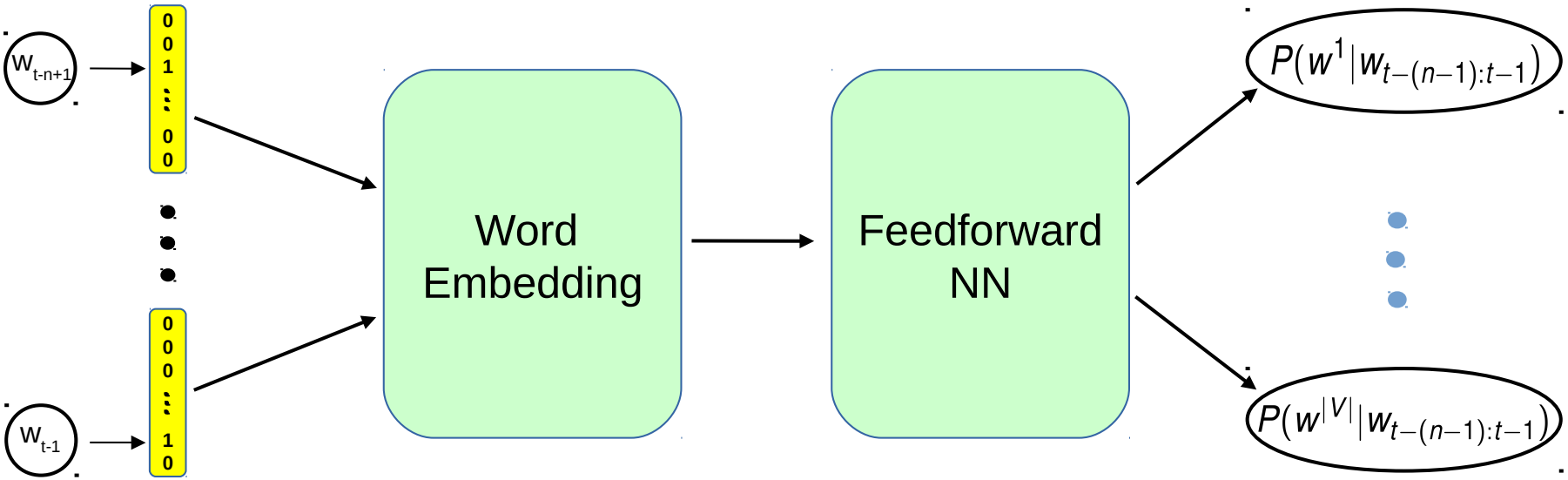
A NPLM learns both goals simultaneously.



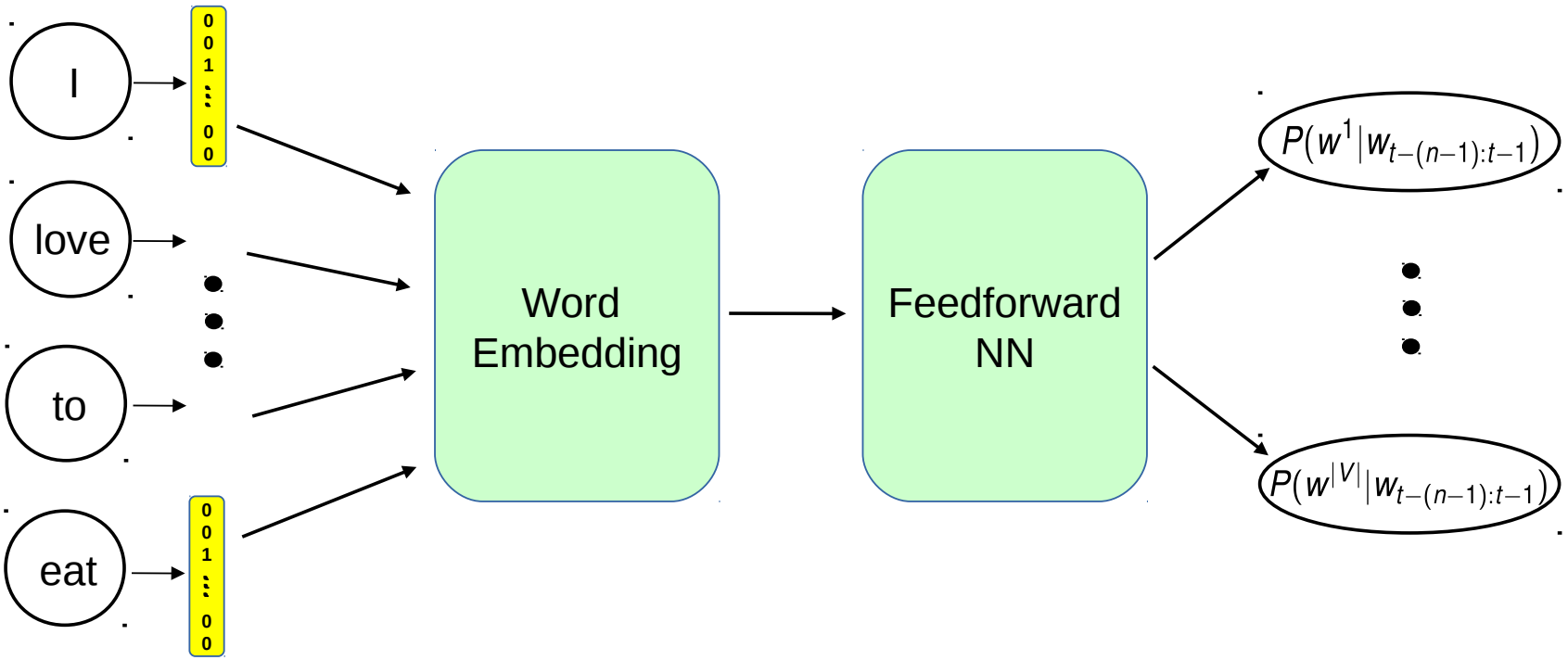
- Typically, input words are encoded by a 1-of-$|V|$ representation (aka one-hot representation).



NPLM in a Nutshell

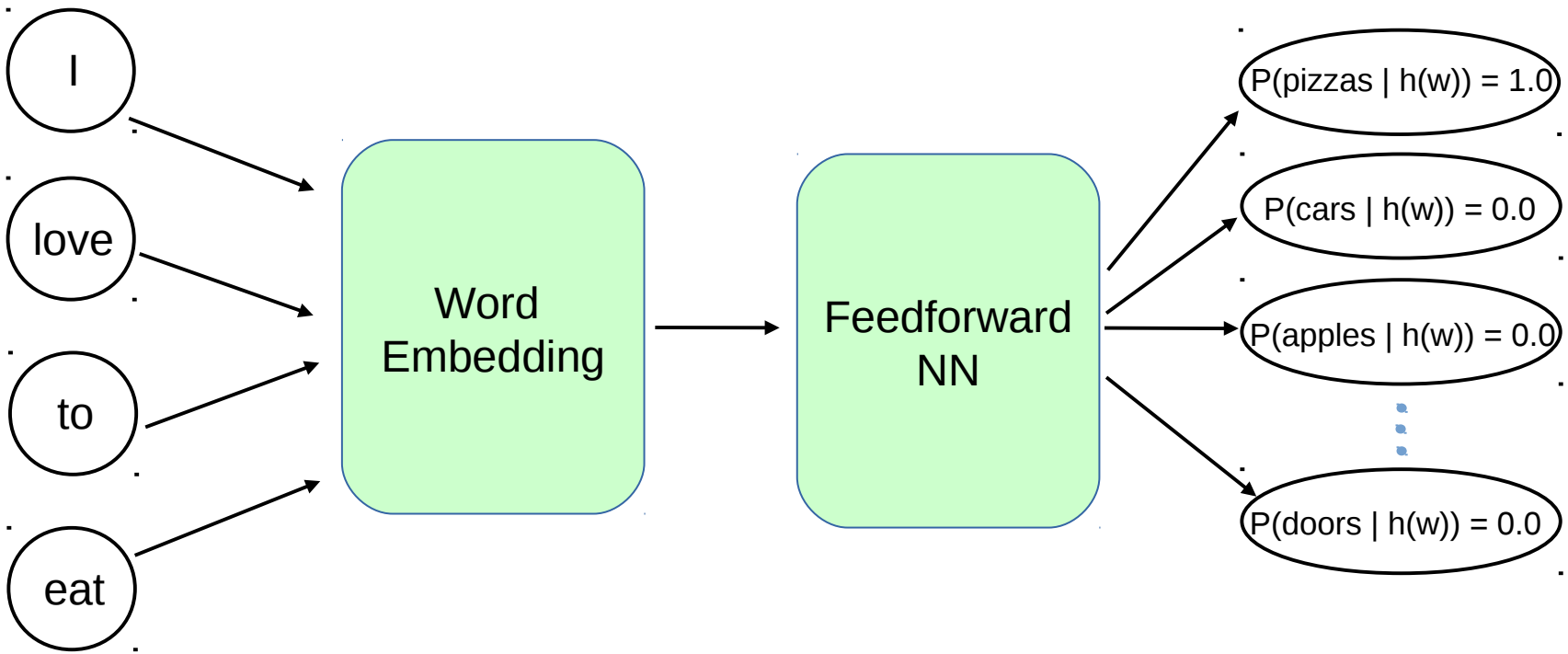


NPLM in a Nutshell



Labeled training instance (x,y):

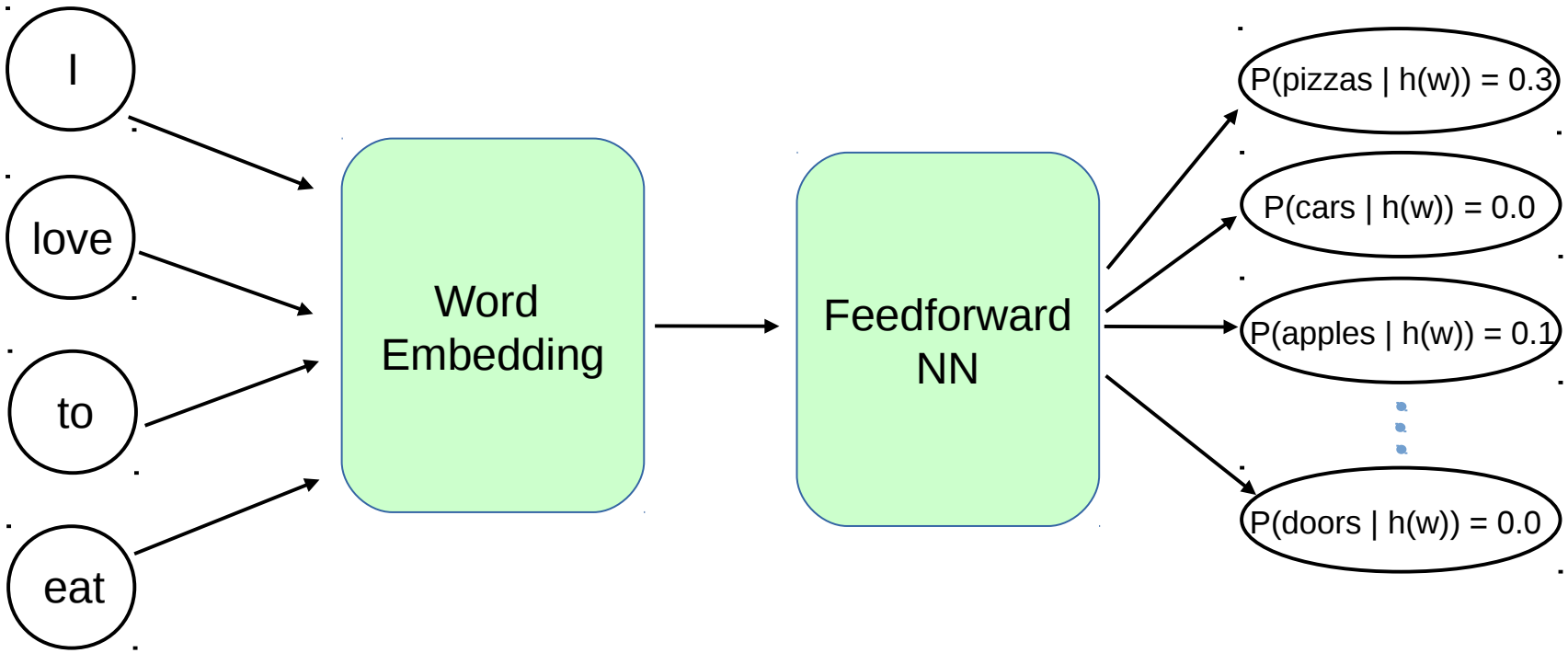
x: I love to eat
y: Pizzas



At training time, model maximizes log-likelihood of the word sequences, $w_{t:t-n+1}$, in the training data: $ML(\theta) = \arg\max_{\theta} \sum_{w_{t:t-n+1}} \log P(w_t | w_{t-1:t-n+1}; \theta)$

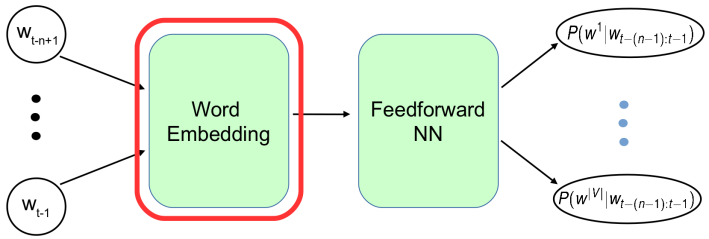
Test instance (x,y):

x: I love to eat
y: ?



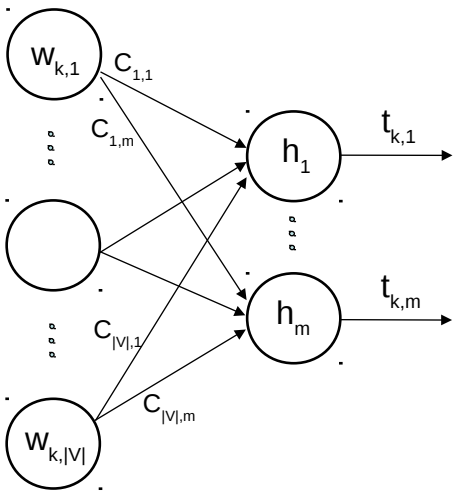
At test time, the model predicts the probability of the next word in the sequence, i.e., $P(w_t|w_{t-1:t-n+1};\theta)$.

Goal 1: Learn word embedding



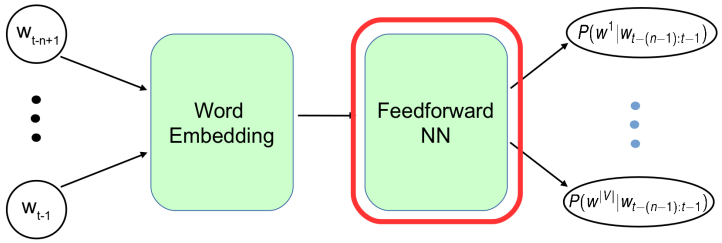
Learn Projection Matrix

$\mathbf{C} \in R^{|V|xm}$

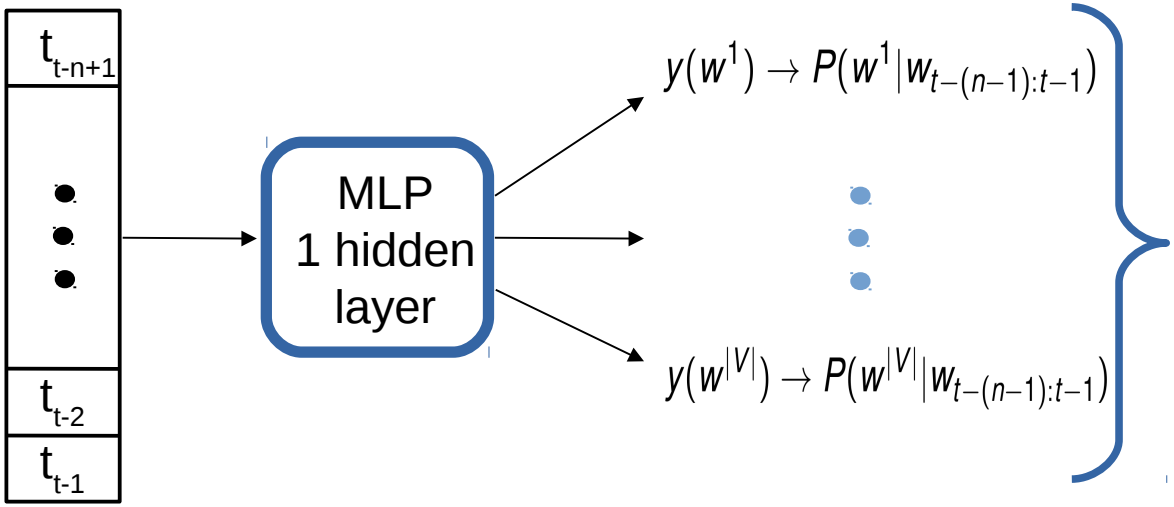


Projection matrix C embeds a word W_k to a vector $t_k \in \mathbb{R}^m$

Goal 2: Learn to Predict Words



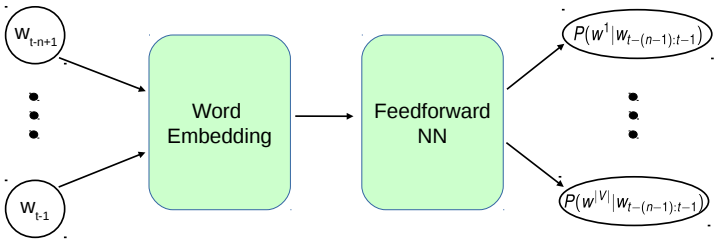
Learn weights for Feedforward NN

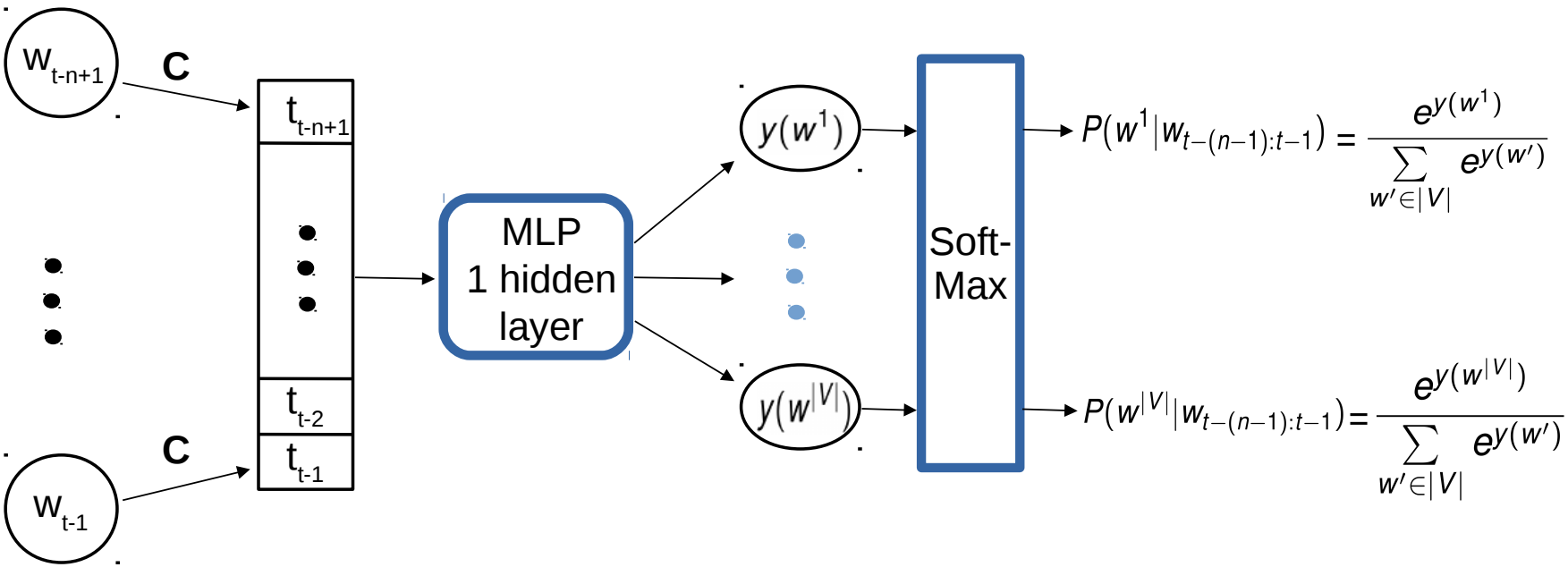


Soft-Max outputs:

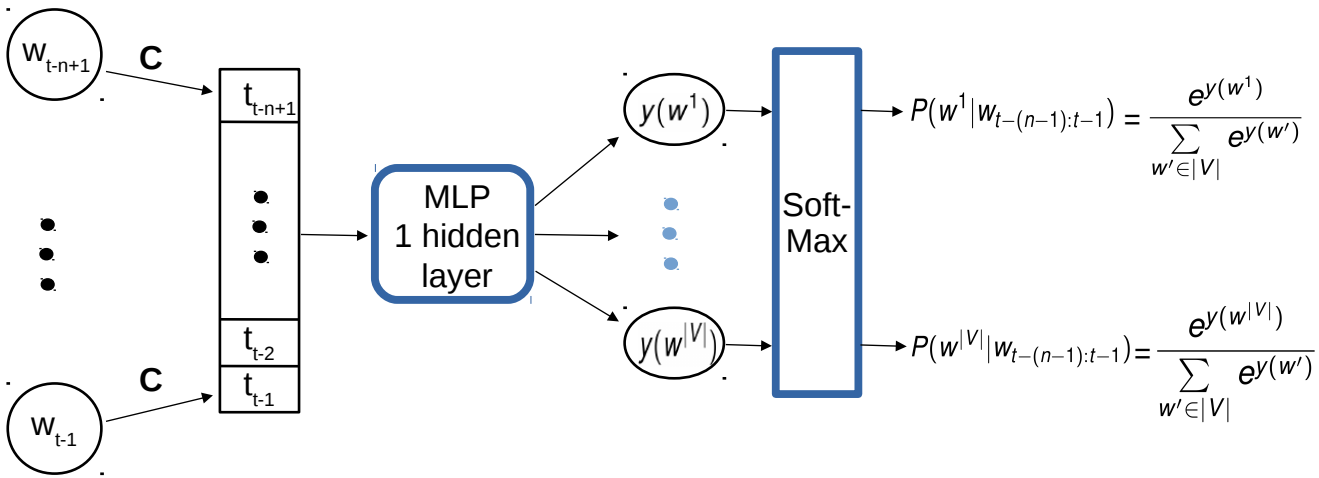
$$P(w^k|w_{t-(n-1):t-1}) = \frac{e^{y(w^k)}}{\sum_{w' \in |V|} e^{y(w')}}$$

Both steps together





Resulting Network Structure



- Matrix $C \in \mathbb{R}^{m \times |V|}$ corresponds to a bilinear transformation.
- This transformation is shared by all words.
- Given the 1-out-of-$|V|$ input representation, embedding of dictionary word w_k is equivalent to column k of C (why?). So it reduces to a linear transformation with complexity $O(m)$.
- As a consequence, embedding C is usually implemented using a lookup table.

Results

Brown corpus

- Stream of 1.181.041 words from a large variety of English texts and books.
- Training set 800.000 words. Validation set 200.000 words (model selection, weight decay, early stopping). Test set 181.041 words.
- $|V| = 47.578$. Rare words with frequency ≤ 3 are merged into a single symbol, reducing the vocabulary size to $|V| = 16.383$.

Associated Press (AP) News

- Training set 13.994.528 words. Validation set 963.138 words. Test set 963.071 words.
- $|V| = 148.721$, reduced to $|V| = 17964$ by keeping only the most frequent words (and keeping punctuation), mapping upper case to lower case, mapping numeric forms to special symbols, mapping rare words to a special symbol, and mapping proper nouns to another special symbol.

Results

Main result in Bengio et al., 2003

- Proposed NPLM significantly outperforms n-grams models.
- Test perplexity difference of about 24% on Brown and about 8% on AP News, when taking the MLP versus the n-gram that worked best on the validation set (a 5-grams approach).
- Training time: 5 epochs for AP News (14M words) take over 3 weeks using 40 CPUs (2002).

Perplexity (PP)

- Measure average level of uncertainty about next word.
- Lower perplexity is better.

NPLMs and Word Embeddings

NPLMs and word feature learning

Word feature learning

- As we discussed, the two previous NPLM approaches achieve 2 goals: i) Word vector embedding and ii) Word-context probabilistic modeling (language modeling).

NPLMs and word feature learning

Word feature learning

- As we discussed, the two previous NPLM approaches achieve 2 goals: i) Word vector embedding and ii) Word-context probabilistic modeling (language modeling).
- Actually, the original goal of the previous models was word-context modeling, but as a collateral effect the methods provide also a word vector representation.

NPLMs and word feature learning

Word feature learning

- As we discussed, the two previous NPLM approaches achieve 2 goals: i) Word vector embedding and ii) Word-context probabilistic modeling (language modeling).
- Actually, the original goal of the previous models was word-context modeling, but as a collateral effect the methods provide also a word vector representation.
- Learning a suitable vector representation for words is indeed a highly relevant goal. This task can be cast as feature learning, where the resulting features can be used (transfer) to feed several relevant NLP applications.

NPLMs and word feature learning

Word feature learning

- As we discussed, the two previous NPLM approaches achieve 2 goals: i) Word vector embedding and ii) Word-context probabilistic modeling (language modeling).
- Actually, the original goal of the previous models was word-context modeling, but as a collateral effect the methods provide also a word vector representation.
- Learning a suitable vector representation for words is indeed a highly relevant goal. This task can be cast as feature learning, where the resulting features can be used (transfer) to feed several relevant NLP applications.

NPLMs and word feature learning

Word feature learning

- As we discussed, the two previous NPLM approaches achieve 2 goals: i) Word vector embedding and ii) Word-context probabilistic modeling (language modeling).
- Actually, the original goal of the previous models was word-context modeling, but as a collateral effect the methods provide also a word vector representation.
- Learning a suitable vector representation for words is indeed a highly relevant goal. This task can be cast as feature learning, where the resulting features can be used (transfer) to feed several relevant NLP applications.

Word2vec

- As described in Mikolov et al., 2013, NPLM methods such as Bengio 2003 are highly inefficient to obtain vector representations. In short, they have too many parameters and are costly to train.

NPLMs and word feature learning

Word feature learning

- As we discussed, the two previous NPLM approaches achieve 2 goals: i) Word vector embedding and ii) Word-context probabilistic modeling (language modeling).
- Actually, the original goal of the previous models was word-context modeling, but as a collateral effect the methods provide also a word vector representation.
- Learning a suitable vector representation for words is indeed a highly relevant goal. This task can be cast as feature learning, where the resulting features can be used (transfer) to feed several relevant NLP applications.

Word2vec

- As described in Mikolov et al., 2013, NPLM methods such as Bengio 2003 are highly inefficient to obtain vector representations. In short, they have too many parameters and are costly to train.
- As a consequence, these methods can hardly scale to large datasets with billions of words and millions of words in the vocabulary.

NPLMs and word feature learning

Word feature learning

- As we discussed, the two previous NPLM approaches achieve 2 goals: i) Word vector embedding and ii) Word-context probabilistic modeling (language modeling).
- Actually, the original goal of the previous models was word-context modeling, but as a collateral effect the methods provide also a word vector representation.
- Learning a suitable vector representation for words is indeed a highly relevant goal. This task can be cast as feature learning, where the resulting features can be used (transfer) to feed several relevant NLP applications.

Word2vec

- As described in Mikolov et al., 2013, NPLM methods such as Bengio 2003 are highly inefficient to obtain vector representations. In short, they have too many parameters and are costly to train.
- As a consequence, these methods can hardly scale to large datasets with billions of words and millions of words in the vocabulary.
- If we do not care about language modelling, we can use simpler models to learn word vector representations: word2vec.

NPLMs and word feature learning

Word feature learning

- As we discussed, the two previous NPLM approaches achieve 2 goals: i) Word vector embedding and ii) Word-context probabilistic modeling (language modeling).
- Actually, the original goal of the previous models was word-context modeling, but as a collateral effect the methods provide also a word vector representation.
- Learning a suitable vector representation for words is indeed a highly relevant goal. This task can be cast as feature learning, where the resulting features can be used (transfer) to feed several relevant NLP applications.

Word2vec

- As described in Mikolov et al., 2013, NPLM methods such as Bengio 2003 are highly inefficient to obtain vector representations. In short, they have too many parameters and are costly to train.
- As a consequence, these methods can hardly scale to large datasets with billions of words and millions of words in the vocabulary.
- If we do not care about language modelling, we can use simpler models to learn word vector representations: word2vec.
- Idea: simpler model can scale to bigger datasets. Algorithmic design should trade model's complexity for efficiency.

Continuous BoW and Skip-gram models, Mikolov et al., 2013

- Mikolov et al., ICLR-2013 and NIPS-2013, present two NPLM models that primarily focus on word embedding.

Continuous BoW and Skip-gram models, Mikolov et al., 2013

- Mikolov et al., ICLR-2013 and NIPS-2013, present two NPLM models that primarily focus on word embedding.
- These models explore simple network architectures that can be efficiently trained on large datasets.

Continuous BoW and Skip-gram models, Mikolov et al., 2013

- Mikolov et al., ICLR-2013 and NIPS-2013, present two NPLM models that primarily focus on word embedding.
- These models explore simple network architectures that can be efficiently trained on large datasets.

Continuous BoW and Skip-gram models, Mikolov et al., 2013

- Mikolov et al., ICLR-2013 and NIPS-2013, present two NPLM models that primarily focus on word embedding.
- These models explore simple network architectures that can be efficiently trained on large datasets.

With respect to Bengio et al. 2003, they propose two main simplifications:

1. Remove the hidden layer and directly connect the low-dim word vector representation to the soft-max output units. To achieve this, they propose two scheme:

Continuous BoW and Skip-gram models, Mikolov et al., 2013

- Mikolov et al., ICLR-2013 and NIPS-2013, present two NPLM models that primarily focus on word embedding.
- These models explore simple network architectures that can be efficiently trained on large datasets.

With respect to Bengio et al. 2003, they propose two main simplifications:

1. Remove the hidden layer and directly connect the low-dim word vector representation to the soft-max output units. To achieve this, they propose two scheme:
   - Continuous Bag-of-Words Model (CBoW).

Continuous BoW and Skip-gram models, Mikolov et al., 2013

- Mikolov et al., ICLR-2013 and NIPS-2013, present two NPLM models that primarily focus on word embedding.
- These models explore simple network architectures that can be efficiently trained on large datasets.

With respect to Bengio et al. 2003, they propose two main simplifications:

1. Remove the hidden layer and directly connect the low-dim word vector representation to the soft-max output units. To achieve this, they propose two scheme:
   - Continuous Bag-of-Words Model (CBoW).
   - Continuous Skip-gram Model (Skip-gram).

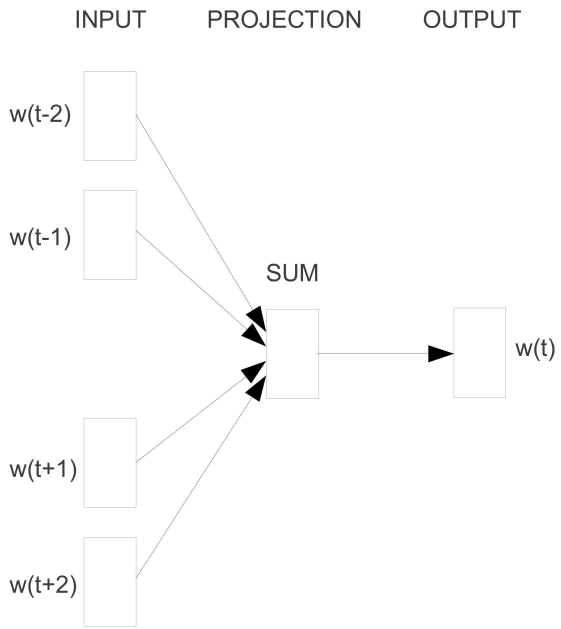
Continuous BoW and Skip-gram models, Mikolov et al., 2013

- Mikolov et al., ICLR-2013 and NIPS-2013, present two NPLM models that primarily focus on word embedding.
- These models explore simple network architectures that can be efficiently trained on large datasets.

With respect to Bengio et al. 2003, they propose two main simplifications:

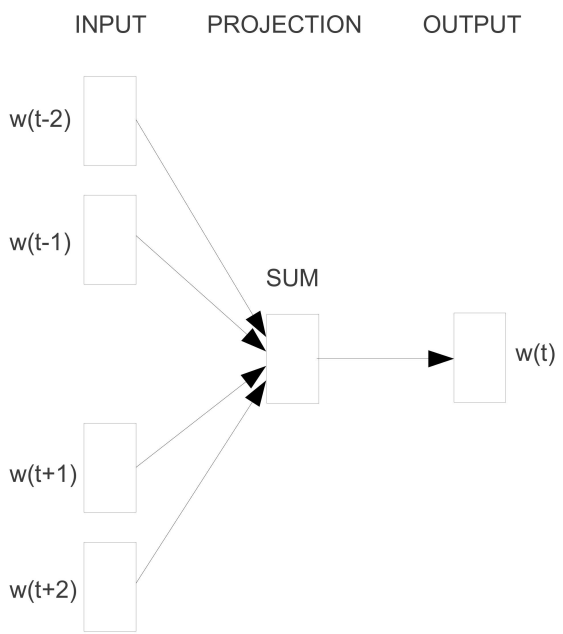
1. Remove the hidden layer and directly connect the low-dim word vector representation to the soft-max output units. To achieve this, they propose two scheme:
   - Continuous Bag-of-Words Model (CBoW).
   - Continuous Skip-gram Model (Skip-gram).
2. Replace the costly soft-max step by simpler schemes, such as, hierarchical soft-max (HSMax) or NCE.

Continuous Bag-of-Words Model (CBoW)



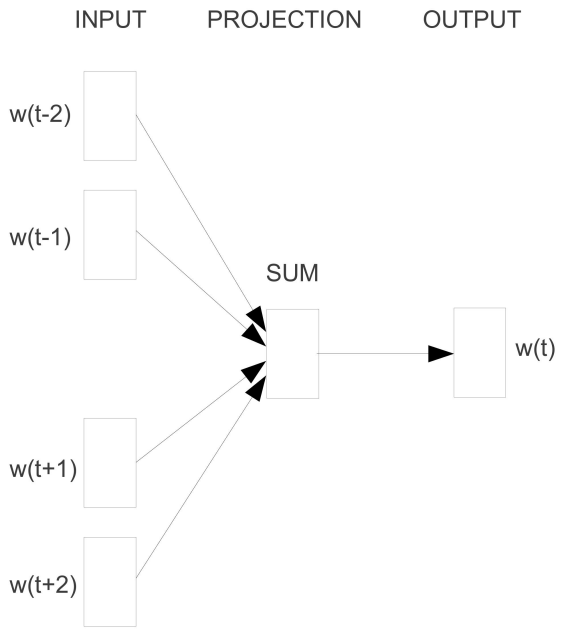
- CBoW sums the projected vector representations of the context words.

Continuous Bag-of-Words Model (CBoW)



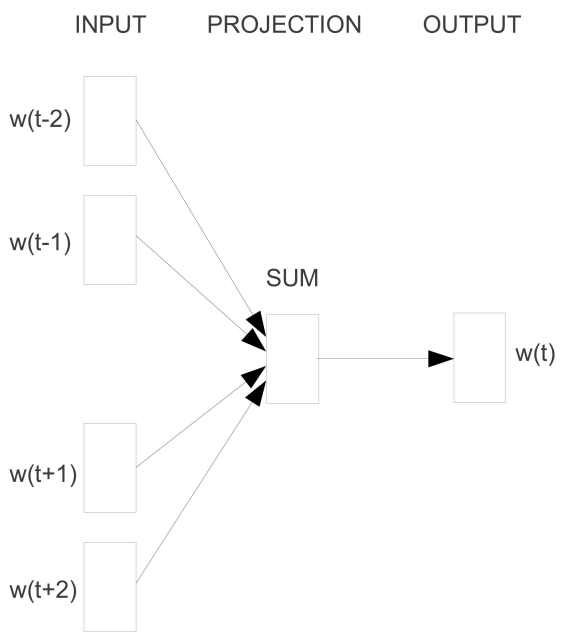
- CBoW sums the projected vector representations of the context words.
- Like BoWs, the resulting vector representation is an average vector, where word position is lost.

Continuous Bag-of-Words Model (CBoW)



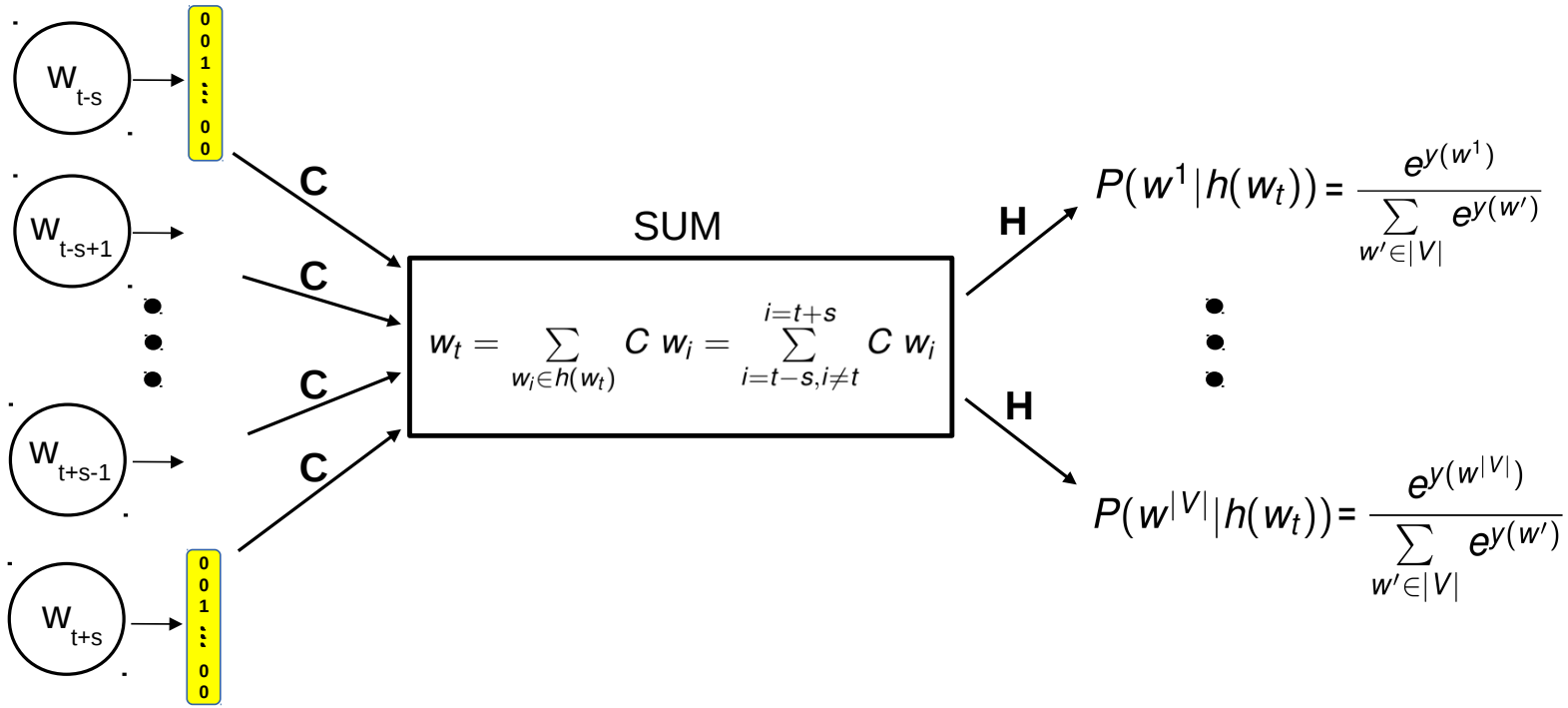
- CBoW sums the projected vector representations of the context words.
- Like BoWs, the resulting vector representation is an average vector, where word position is lost.
- Unlike BoWs, the projection layer produces continuous vector representations, so the name CBoW.

Continuous Bag-of-Words Model (CBoW)



- CBoW sums the projected vector representations of the context words.
- Like BoWs, the resulting vector representation is an average vector, where word position is lost.
- Unlike BoWs, the projection layer produces continuous vector representations, so the name CBoW.
- The CBoW output vector w(t) is connected through a single proyection or layer to the soft-max output nodes.

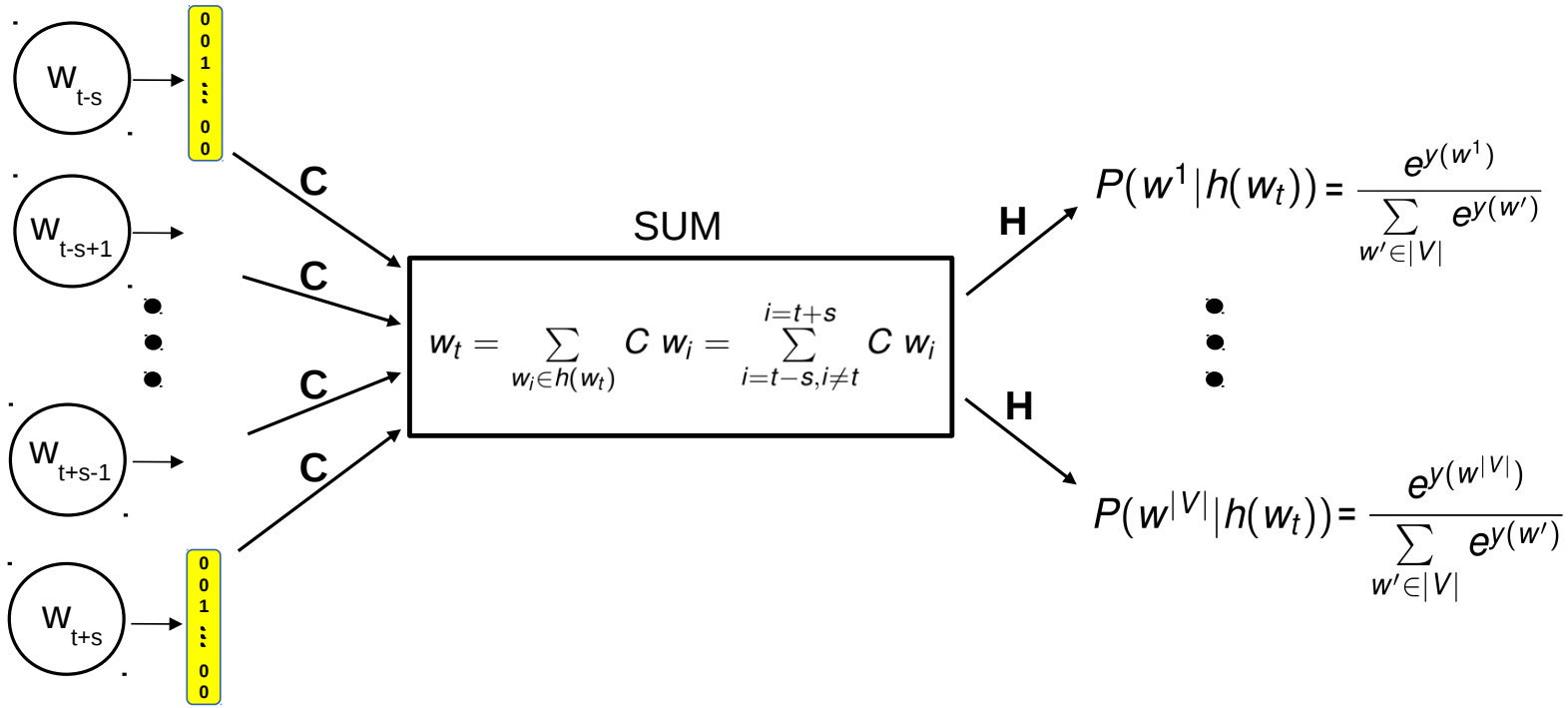
Continuous Bag-of-Words Model (CBoW)



$$C \in \mathbb{R}^{|V| x m}, H \in \mathbb{R}^{m x |V|}$$

- The goal of the training step is to learn weight matrices C and H.

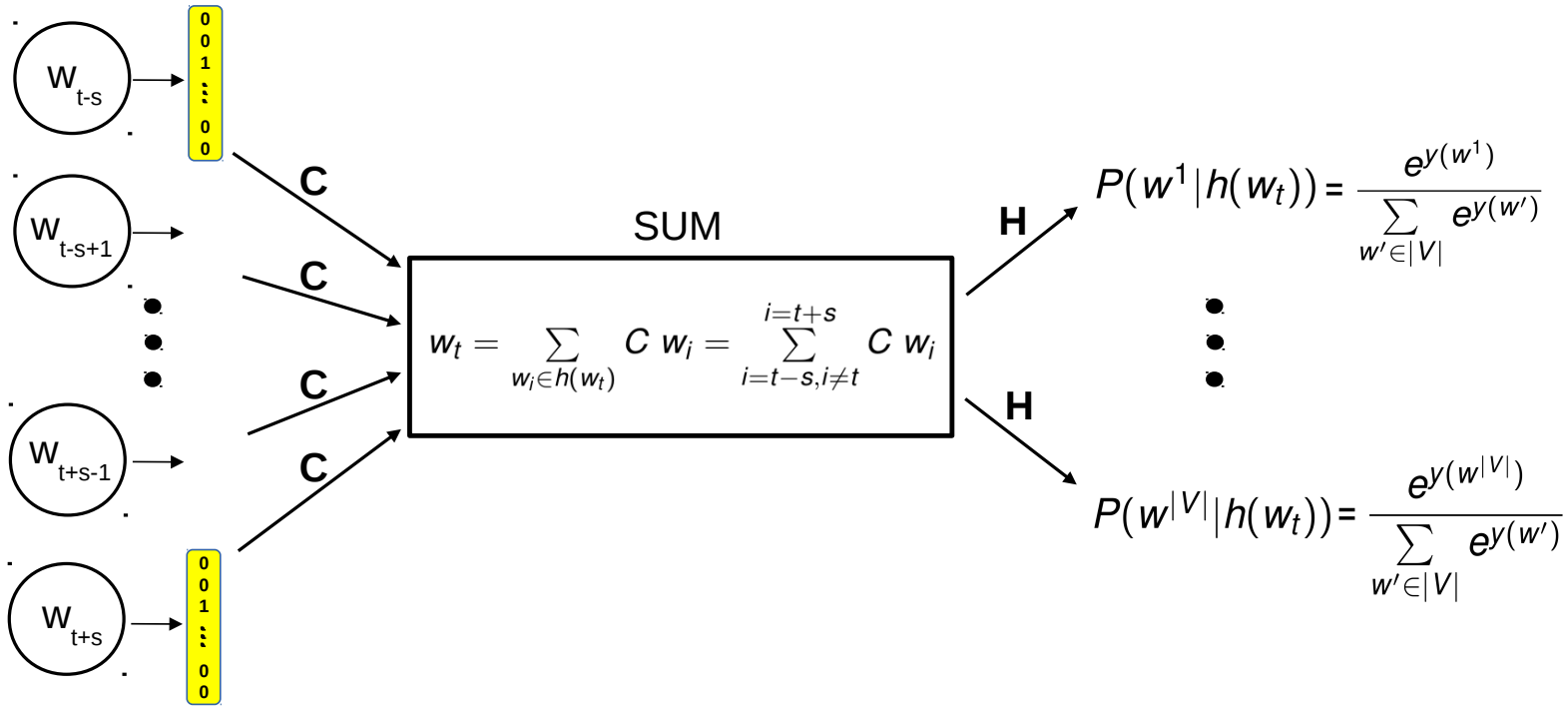
Continuous Bag-of-Words Model (CBoW)



$$C \in \mathbb{R}^{|V| x m}, H \in \mathbb{R}^{m x |V|}$$

- The goal of the training step is to learn weight matrices C and H.
- In particular, we are interested on matrix C that has on its columns the |V| word embedded vectors.

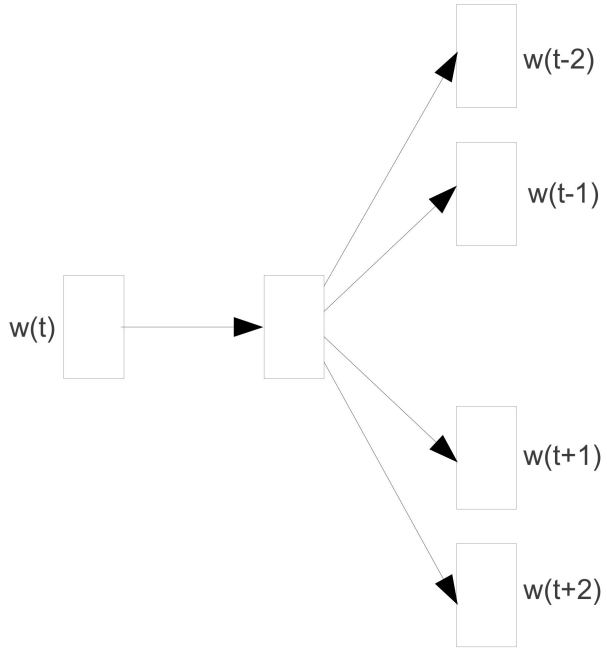
Continuous Bag-of-Words Model (CBoW)



$$C \in \mathbb{R}^{|V| x m}, H \in \mathbb{R}^{m x |V|}$$

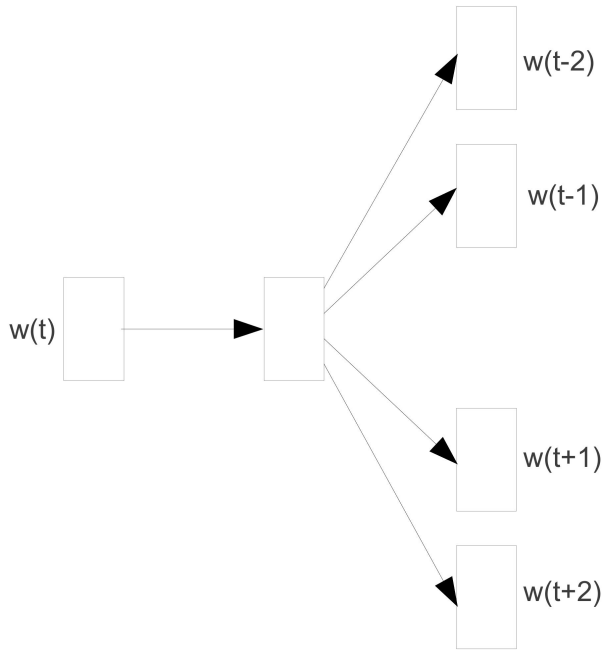
- The goal of the training step is to learn weight matrices C and H.
- In particular, we are interested on matrix C that has on its columns the |V| word embedded vectors.
- Notice that there is not a hidden layer, but a direct bilinear connection H between the projection layer and softmax outputs.

Continuous Skip-gram Model



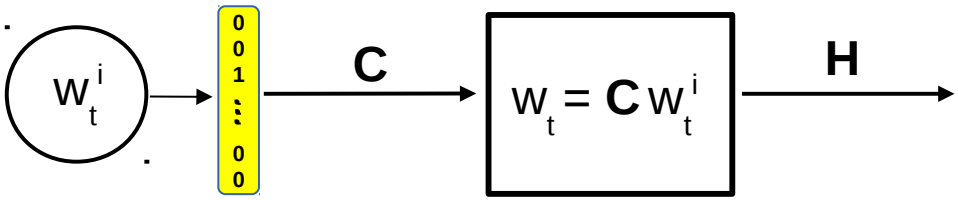
- Given a target word w (usually the center word), the Skip-gram model predicts possible context words within a certain range surrounding word w.

Continuous Skip-gram Model



- Given a target word w (usually the center word), the Skip-gram model predicts possible context words within a certain range surrounding word w.
- More distance words are usually less related to the central one. The Skip-gram model takes account of this fact by sampling less from these words, so they have less relevance in the parameter estimation.

Continuous Skip-gram Model

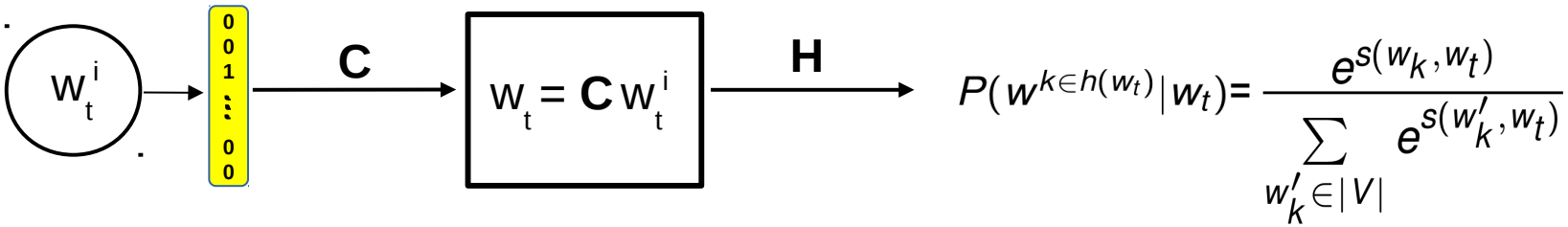


$$P(w^{k \in h(w_t)} | w_t) = \frac{e^{s(w_k, w_t)}}{\sum_{w'_k \in |V|} e^{s(w'_k, w_t)}}$$

where $s(w_k, w_t) = w_k^T w_t$. $C \in \mathbb{R}^{|V| x m}$, $H \in \mathbb{R}^{m x |V|}$

- The goal of the training step is to learn weight matrices C and H.

Continuous Skip-gram Model



where $s(w_k, w_t) = w_k^T w_t$. $C \in \mathbb{R}^{|V| x m}$, $H \in \mathbb{R}^{m x |V|}$

- The goal of the training step is to learn weight matrices C and H.
- Notice that there is not a hidden layer, but a direct linear connection H between the projection layer and softmax outputs.

CBoW and Skip-gram models: Learning

- CBoW parameter learning is similar to previous NPLM. For the experiments reported, they use stochastic gradient descent and backpropagation. Learning rate starts at 0.025 and decreases linearly, so that it approaches zero at the end of the last training epoch.

CBoW and Skip-gram models: Learning

- CBoW parameter learning is similar to previous NPLM. For the experiments reported, they use stochastic gradient descent and backpropagation. Learning rate starts at 0.025 and decreases linearly, so that it approaches zero at the end of the last training epoch.
- Skip-gram parameter learning is more complex, it is based in a technique called Noise Contrastive Estimation or NCE. The main idea is to estimate parameters by learning to discriminate between true and noisy data instances.

CBoW and Skip-gram models: Learning

- CBoW parameter learning is similar to previous NPLM. For the experiments reported, they use stochastic gradient descent and backpropagation. Learning rate starts at 0.025 and decreases linearly, so that it approaches zero at the end of the last training epoch.
- Skip-gram parameter learning is more complex, it is based in a technique called Noise Contrastive Estimation or NCE. The main idea is to estimate parameters by learning to discriminate between true and noisy data instances.
- Optimized code for computing the CBOW and skip-gram models is available online: `https://code.google.com/p/word2vec/`

CBoW and Skip-gram models: Learning

- CBoW parameter learning is similar to previous NPLM. For the experiments reported, they use stochastic gradient descent and backpropagation. Learning rate starts at 0.025 and decreases linearly, so that it approaches zero at the end of the last training epoch.
- Skip-gram parameter learning is more complex, it is based in a technique called Noise Contrastive Estimation or NCE. The main idea is to estimate parameters by learning to discriminate between true and noisy data instances.
- Optimized code for computing the CBOW and skip-gram models is available online: `https://code.google.com/p/word2vec/`
- A version trained for Spanish can be found online at: `https://github.com/uchile-nlp/spanish-word-embeddings`

Results

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

Goal: Evaluate capacity to identify semantic and syntactic word relationships.

- Test set contains 8869 semantic and 10675 syntactic questions (Semantic-Syntactic dataset).

Results

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

Goal: Evaluate capacity to identify semantic and syntactic word relationships.

- Test set contains 8869 semantic and 10675 syntactic questions (Semantic-Syntactic dataset).
- Relations are divided in 5 types of semantic relations and 9 types of syntactic relations. Table shows two examples from each type of relations.

Results

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

Goal: Evaluate capacity to identify semantic and syntactic word relationships.

- Test set contains 8869 semantic and 10675 syntactic questions (Semantic-Syntactic dataset).
- Relations are divided in 5 types of semantic relations and 9 types of syntactic relations. Table shows two examples from each type of relations.
- Training set is given by Google News corpus, which contains 6B tokens.

Results

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

Goal: Evaluate capacity to identify semantic and syntactic word relationships.

- Test set contains 8869 semantic and 10675 syntactic questions (Semantic-Syntactic dataset).
- Relations are divided in 5 types of semantic relations and 9 types of syntactic relations. Table shows two examples from each type of relations.
- Training set is given by Google News corpus, which contains 6B tokens.
- For training set, they restrict the vocabulary size to 1 million most frequent words.

Results

- For the evaluation they answer questions. Ex. "What is the word that is similar to small in the same sense as biggest is similar to big?"

Results

- For the evaluation they answer questions. Ex. "What is the word that is similar to small in the same sense as biggest is similar to big?"
- They answer each question by performing simple algebraic operations with the vector representation of words.

Results

- For the evaluation they answer questions. Ex. “What is the word that is similar to small in the same sense as biggest is similar to big?”
- They answer each question by performing simple algebraic operations with the vector representation of words.
- Ex. To find a word that is similar to small in the same sense as biggest is similar to big, they compute vector X=vector(“biggest”)-vector(“big”) + vector(“small”).

Results

- For the evaluation they answer questions. Ex. "What is the word that is similar to small in the same sense as biggest is similar to big?"
- They answer each question by performing simple algebraic operations with the vector representation of words.
- Ex. To find a word that is similar to small in the same sense as biggest is similar to big, they compute vector X=vector("biggest")-vector("big") + vector("small").
- Then, they search in the vector space for the word closest to X measured by cosine distance, and use it as the answer to the question (we discard the input question words during this search).

Results

- For the evaluation they answer questions. Ex. "What is the word that is similar to small in the same sense as biggest is similar to big?"
- They answer each question by performing simple algebraic operations with the vector representation of words.
- Ex. To find a word that is similar to small in the same sense as biggest is similar to big, they compute vector X=vector("biggest")-vector("big") + vector("small").
- Then, they search in the vector space for the word closest to X measured by cosine distance, and use it as the answer to the question (we discard the input question words during this search).
- Question is assumed to be correctly answered only if the closest word to the vector computed using the above method is exactly the same as the correct word in the question

Results

- For the evaluation they answer questions. Ex. "What is the word that is similar to small in the same sense as biggest is similar to big?"
- They answer each question by performing simple algebraic operations with the vector representation of words.
- Ex. To find a word that is similar to small in the same sense as biggest is similar to big, they compute vector X=vector("biggest")-vector("big") + vector("small").
- Then, they search in the vector space for the word closest to X measured by cosine distance, and use it as the answer to the question (we discard the input question words during this search).
- Question is assumed to be correctly answered only if the closest word to the vector computed using the above method is exactly the same as the correct word in the question
- Synonyms are thus counted as mistakes. This could be improved in the future using some more elaborated language information.

Results

Model	Vector Dimensionality	Training words	Accuracy [%]		
			Semantic	Syntactic	Total
Collobert-Weston NNLM	50	660M	9.3	12.3	11.0
Turian NNLM	50	37M	1.4	2.6	2.1
Turian NNLM	200	37M	1.4	2.2	1.8
Mnih NNLM	50	37M	1.8	9.1	5.8
Mnih NNLM	100	37M	3.3	13.2	8.8
Mikolov RNNLM	80	320M	4.9	18.4	12.7
Mikolov RNNLM	640	320M	8.6	36.5	24.6
Huang NNLM	50	990M	13.3	11.6	12.3
Our NNLM	20	6B	12.9	26.4	20.3
Our NNLM	50	6B	27.9	55.8	43.2
Our NNLM	100	6B	34.2	**64.5**	50.8
CBOW	300	783M	15.5	53.1	36.1
Skip-gram	300	783M	**50.0**	55.9	**53.3**

- Table compares results on Semantic-Syntactic dataset for different publicly available algorithms to obtain word vector embedding.

Results

Model	Vector Dimensionality	Training words	Accuracy [%]		
			Semantic	Syntactic	Total
Collobert-Weston NNLM	50	660M	9.3	12.3	11.0
Turian NNLM	50	37M	1.4	2.6	2.1
Turian NNLM	200	37M	1.4	2.2	1.8
Mnih NNLM	50	37M	1.8	9.1	5.8
Mnih NNLM	100	37M	3.3	13.2	8.8
Mikolov RNNLM	80	320M	4.9	18.4	12.7
Mikolov RNNLM	640	320M	8.6	36.5	24.6
Huang NNLM	50	990M	13.3	11.6	12.3
Our NNLM	20	6B	12.9	26.4	20.3
Our NNLM	50	6B	27.9	55.8	43.2
Our NNLM	100	6B	34.2	**64.5**	50.8
CBOW	300	783M	15.5	53.1	36.1
Skip-gram	300	783M	**50.0**	55.9	**53.3**

- Table compares results on Semantic-Syntactic dataset for different publicly available algorithms to obtain word vector embedding.
- CBOW and Skip-gram model are trained on subset of the Google News data.

Results

Model	Vector Dimensionality	Training words	Accuracy [%]		
			Semantic	Syntactic	Total
Collobert-Weston NNLM	50	660M	9.3	12.3	11.0
Turian NNLM	50	37M	1.4	2.6	2.1
Turian NNLM	200	37M	1.4	2.2	1.8
Mnih NNLM	50	37M	1.8	9.1	5.8
Mnih NNLM	100	37M	3.3	13.2	8.8
Mikolov RNNLM	80	320M	4.9	18.4	12.7
Mikolov RNNLM	640	320M	8.6	36.5	24.6
Huang NNLM	50	990M	13.3	11.6	12.3
Our NNLM	20	6B	12.9	26.4	20.3
Our NNLM	50	6B	27.9	55.8	43.2
Our NNLM	100	6B	34.2	**64.5**	50.8
CBOW	300	783M	15.5	53.1	36.1
Skip-gram	300	783M	**50.0**	55.9	**53.3**

- Table compares results on Semantic-Syntactic dataset for different publicly available algorithms to obtain word vector embedding.
- CBOW and Skip-gram model are trained on subset of the Google News data.
- CBoW training is about 1 day, Skip-gram is about 3 days.

Results

Model	Vector Dimensionality	Training words	Accuracy [%]		
			Semantic	Syntactic	Total
Collobert-Weston NNLM	50	660M	9.3	12.3	11.0
Turian NNLM	50	37M	1.4	2.6	2.1
Turian NNLM	200	37M	1.4	2.2	1.8
Mnih NNLM	50	37M	1.8	9.1	5.8
Mnih NNLM	100	37M	3.3	13.2	8.8
Mikolov RNNLM	80	320M	4.9	18.4	12.7
Mikolov RNNLM	640	320M	8.6	36.5	24.6
Huang NNLM	50	990M	13.3	11.6	12.3
Our NNLM	20	6B	12.9	26.4	20.3
Our NNLM	50	6B	27.9	55.8	43.2
Our NNLM	100	6B	34.2	**64.5**	50.8
CBOW	300	783M	15.5	53.1	36.1
Skip-gram	300	783M	**50.0**	55.9	**53.3**

- Table compares results on Semantic-Syntactic dataset for different publicly available algorithms to obtain word vector embedding.
- CBOW and Skip-gram model are trained on subset of the Google News data.
- CBoW training is about 1 day, Skip-gram is about 3 days.
- Note that in the semantic relationships, the skip-gram model substantially outperforms the CBoW model, any guess?

Results

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

- They perform a big test using the 6B word corpus on a distributed framework called DistBelief.

Results

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

- They perform a big test using the 6B word corpus on a distributed framework called DistBelief.
- They use mini-batch asynchronous gradient descent and Adagrad algorithm to adaptively control the learning rate.

Results

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

- They perform a big test using the 6B word corpus on a distributed framework called DistBelief.
- They use mini-batch asynchronous gradient descent and Adagrad algorithm to adaptively control the learning rate.
- They use 50 to 100 model replicas during the training.

Results

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

- They perform a big test using the 6B word corpus on a distributed framework called DistBelief.
- They use mini-batch asynchronous gradient descent and Adagrad algorithm to adaptively control the learning rate.
- They use 50 to 100 model replicas during the training.
- The number of CPU cores reported is an estimate (cluster is shared with other tasks).

Results

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

- They perform a big test using the 6B word corpus on a distributed framework called DistBelief.
- They use mini-batch asynchronous gradient descent and Adagrad algorithm to adaptively control the learning rate.
- They use 50 to 100 model replicas during the training.
- The number of CPU cores reported is an estimate (cluster is shared with other tasks).
- Note that training of NNLM with 1000-dimensional vectors would take too long to complete.

Results: Additive Compositionality

- Previous examples show that the word2vect embeddings exhibit a linear structure that makes it possible to perform analogical reasoning using vector arithmetics.

Results: Additive Compositionality

- Previous examples show that the word2vect embeddings exhibit a linear structure that makes it possible to perform analogical reasoning using vector arithmetics.
- Element-wise vector additions provide Additive Compositionality of semantic concepts. Following table shows some examples:

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

Results: Additive Compositionality

- Figure shows 2D PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities.

Results: Additive Compositionality

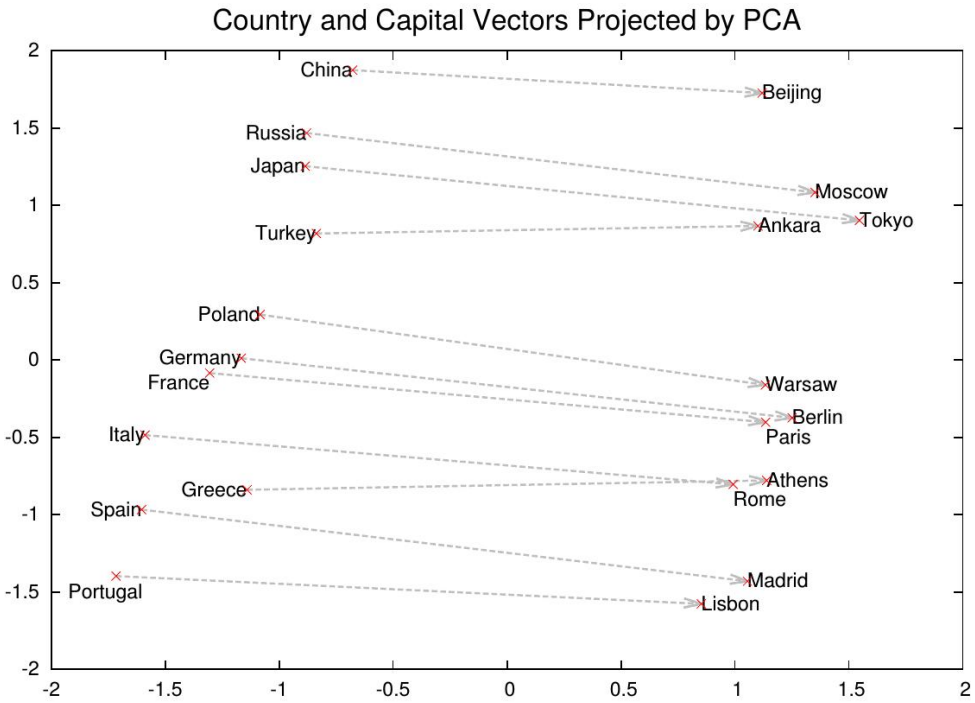
- Figure shows 2D PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities.
- Note that during training, the algorithm did not receive any supervised information about what a capital city means.

Results: Additive Compositionality

- Figure shows 2D PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities.
- Note that during training, the algorithm did not receive any supervised information about what a capital city means.
- However, the model is able to automatically organize concepts and learn implicitly the relationships between them.

Results: Additive Compositionality

- Figure shows 2D PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities.
- Note that during training, the algorithm did not receive any supervised information about what a capital city means.
- However, the model is able to automatically organize concepts and learn implicitly the relationships between them.
- Any intuition of why we obtain this behavior?.



The amazing power of word vectors

- "Tweet2Vec: Character-Based Distributed Representations for Social Media", Dhingra et al. 2016.
- "Visual Word2Vec (vis-w2v): Learning Visually Grounded Word Embeddings Using Abstract Scenes", Kottur et. al, 2015.
- "E-commerce in Your Inbox: Product Recommendations at Scale (Prod-2-Vec)", Grbovic et al. 2016.
- "Meta-Prod2Vec - Product Embeddings Using Side-Information for Recommendation", Vasile et al. 2016. (Criteo)
- "Item2Vec: Neural Item Embedding for Collaborative Filtering", Barkan et. al, 2016