Word Embeddings

Maury Courtland (PhD; www.maury.science)

September 10, 2019

30k foot background: Quantifying Words for Computational Use

30k foot background: Quantifying Words for Computational Use

- Graph-based Relationships (e.g. WordNet and SynSets)

30k foot background: Quantifying Words for Computational Use

- Graph-based Relationships (e.g. WordNet and SynSets)
  - Very informative and rich structure

30k foot background: Quantifying Words for Computational Use

- Graph-based Relationships (e.g. WordNet and SynSets)
  - Very informative and rich structure
  - Can miss non-obvious relationships

30k foot background: Quantifying Words for Computational Use

- Graph-based Relationships (e.g. WordNet and SynSets)
    - Very informative and rich structure
    - Can miss non-obvious relationships
    - Incredibly expensive to collect

30k foot background: Quantifying Words for Computational Use

- Graph-based Relationships (e.g. WordNet and SynSets)
  - Very informative and rich structure
  - Can miss non-obvious relationships
  - Incredibly expensive to collect
- Corpora Statistics

30k foot background: Quantifying Words for Computational Use

- Graph-based Relationships (e.g. WordNet and SynSets)
  - Very informative and rich structure
  - Can miss non-obvious relationships
  - Incredibly expensive to collect
- Corpora Statistics
  - Very cheap to collect (nowadays)

30k foot background: Quantifying Words for Computational Use

- Graph-based Relationships (e.g. WordNet and SynSets)
  - Very informative and rich structure
  - Can miss non-obvious relationships
  - Incredibly expensive to collect
- Corpora Statistics
  - Very cheap to collect (nowadays)
  - Prone to sampling error

30k foot background: Quantifying Words for Computational Use

- Graph-based Relationships (e.g. WordNet and SynSets)
  - Very informative and rich structure
  - Can miss non-obvious relationships
  - Incredibly expensive to collect
- Corpora Statistics
  - Very cheap to collect (nowadays)
  - Prone to sampling error
  - Not straightforward how to extract meaning

30k foot background: Quantifying Words for Computational Use

- Graph-based Relationships (e.g. WordNet and SynSets)
  - Very informative and rich structure
  - Can miss non-obvious relationships
  - Incredibly expensive to collect
- Corpora Statistics
  - Very cheap to collect (nowadays)
  - Prone to sampling error
  - Not straightforward how to extract meaning
- Human Judgements (e.g. MRC)

30k foot background: Quantifying Words for Computational Use

- Graph-based Relationships (e.g. WordNet and SynSets)
  - Very informative and rich structure
  - Can miss non-obvious relationships
  - Incredibly expensive to collect
- Corpora Statistics
  - Very cheap to collect (nowadays)
  - Prone to sampling error
  - Not straightforward how to extract meaning
- Human Judgements (e.g. MRC)
  - Very rich and abstract judgements

30k foot background: Quantifying Words for Computational Use

- Graph-based Relationships (e.g. WordNet and SynSets)
  - Very informative and rich structure
  - Can miss non-obvious relationships
  - Incredibly expensive to collect
- Corpora Statistics
  - Very cheap to collect (nowadays)
  - Prone to sampling error
  - Not straightforward how to extract meaning
- Human Judgements (e.g. MRC)
  - Very rich and abstract judgements
  - Subjective (thus susceptible to noise)

30k foot background: Quantifying Words for Computational Use

- Graph-based Relationships (e.g. WordNet and SynSets)
  - Very informative and rich structure
  - Can miss non-obvious relationships
  - Incredibly expensive to collect
- Corpora Statistics
  - Very cheap to collect (nowadays)
  - Prone to sampling error
  - Not straightforward how to extract meaning
- Human Judgements (e.g. MRC)
  - Very rich and abstract judgements
  - Subjective (thus susceptible to noise)
  - Decently expensive to collect

30k foot background: Quantifying Words for Computational Use

- Graph-based Relationships (e.g. WordNet and SynSets)
  - Very informative and rich structure
  - Can miss non-obvious relationships
  - Incredibly expensive to collect
- Corpora Statistics
  - Very cheap to collect (nowadays)
  - Prone to sampling error
  - Not straightforward how to extract meaning
- Human Judgements (e.g. MRC)
  - Very rich and abstract judgements
  - Subjective (thus susceptible to noise)
  - Decently expensive to collect
- **Choose the cheap one, grab tons of data to deal with sampling error, and figure out how to extract meaning**

Talk Overview

Talk Overview

- Word2Vec and Friends

Talk Overview

- Word2Vec and Friends
- GloVe and global techniques

Talk Overview

- Word2Vec and Friends
- GloVe and global techniques
- ELMo and deep representations

Mikolov et al. 2013a: Efficient Estimation of Word Representations in Vector Space

Background

- Motivation

Background

- Motivation
  - Most previous approaches treat words as atomic units

Background

- Motivation
  - Most previous approaches treat words as atomic units
  - Misses related nature of words (all are categorically different)

Background

- Motivation
  - Most previous approaches treat words as atomic units
  - Misses related nature of words (all are categorically different)
- Previous Approach Strengths

Background

- Motivation
  - Most previous approaches treat words as atomic units
  - Misses related nature of words (all are categorically different)
- Previous Approach Strengths
  - More data (but simple) $>$ More complexity (on less data)

Background

- Motivation
  - Most previous approaches treat words as atomic units
  - Misses related nature of words (all are categorically different)
- Previous Approach Strengths
  - More data (but simple) $>$ More complexity (on less data)
  - Simple, straightforward, and robust

Background

- Motivation
  - Most previous approaches treat words as atomic units
  - Misses related nature of words (all are categorically different)
- Previous Approach Strengths
  - More data (but simple) $>$ More complexity (on less data)
  - Simple, straightforward, and robust
- Previous Shortcomings

Background

- Motivation
  - Most previous approaches treat words as atomic units
  - Misses related nature of words (all are categorically different)
- Previous Approach Strengths
  - More data (but simple) $>$ More complexity (on less data)
  - Simple, straightforward, and robust
- Previous Shortcomings
  - Marginal data returns are decreasing

Background

- Motivation
  - Most previous approaches treat words as atomic units
  - Misses related nature of words (all are categorically different)
- Previous Approach Strengths
  - More data (but simple) $>$ More complexity (on less data)
  - Simple, straightforward, and robust
- Previous Shortcomings
  - Marginal data returns are decreasing
  - Data labeling is very expensive

Paper Goals

- Learn high quality word vectors

Paper Goals

- Learn high quality word vectors
  - Learn many types (1m+ vocab)

Paper Goals

- Learn high quality word vectors
  - Learn many types (1m+ vocab)
  - From many tokens (1B+ corpus)

Paper Goals

- Learn high quality word vectors
  - Learn many types (1m+ vocab)
  - From many tokens (1B+ corpus)
- High dimensional embedding space

Paper Goals

- Learn high quality word vectors
  - Learn many types (1m+ vocab)
  - From many tokens (1B+ corpus)
- High dimensional embedding space
  - Words can have "multiple degree of similarity"

Paper Goals

- Learn high quality word vectors
  - Learn many types (1m+ vocab)
  - From many tokens (1B+ corpus)
- High dimensional embedding space
  - Words can have "multiple degree of similarity"
- Preserve linear structure between words

Paper Goals

- Learn high quality word vectors
  - Learn many types (1m+ vocab)
  - From many tokens (1B+ corpus)
- High dimensional embedding space
  - Words can have "multiple degree of similarity"
- Preserve linear structure between words
  - Enables simple algebra $\overrightarrow{King} - \overrightarrow{Man} + \overrightarrow{Woman} \approx \overrightarrow{Queen}$

Paper Goals

- Learn high quality word vectors
  - Learn many types (1m+ vocab)
  - From many tokens (1B+ corpus)
- High dimensional embedding space
  - Words can have "multiple degree of similarity"
- Preserve linear structure between words
  - Enables simple algebra $\overrightarrow{King} - \overrightarrow{Man} + \overrightarrow{Woman} \approx \overrightarrow{Queen}$
  - Neural nets do this better than LSA (latent semantic analysis)

Paper Goals

- Learn high quality word vectors
  - Learn many types (1m+ vocab)
  - From many tokens (1B+ corpus)
- High dimensional embedding space
  - Words can have “multiple degree of similarity”
- Preserve linear structure between words
  - Enables simple algebra $\overrightarrow{King} - \overrightarrow{Man} + \overrightarrow{Woman} \approx \overrightarrow{Queen}$
  - Neural nets do this better than LSA (latent semantic analysis)
- Be relatively quick training

Paper Goals

- Learn high quality word vectors
  - Learn many types (1m+ vocab)
  - From many tokens (1B+ corpus)
- High dimensional embedding space
  - Words can have "multiple degree of similarity"
- Preserve linear structure between words
  - Enables simple algebra $\overrightarrow{King} - \overrightarrow{Man} + \overrightarrow{Woman} \approx \overrightarrow{Queen}$
  - Neural nets do this better than LSA (latent semantic analysis)
- Be relatively quick training
  - Most previous work took forever to train

Paper Goals

- Learn high quality word vectors
  - Learn many types (1m+ vocab)
  - From many tokens (1B+ corpus)
- High dimensional embedding space
  - Words can have "multiple degree of similarity"
- Preserve linear structure between words
  - Enables simple algebra $\overrightarrow{King} - \overrightarrow{Man} + \overrightarrow{Woman} \approx \overrightarrow{Queen}$
  - Neural nets do this better than LSA (latent semantic analysis)
- Be relatively quick training
  - Most previous work took forever to train
  - Neural nets train much quicker than LDA (latent dirichlet allocation)

Feedforward Neural Net Language Model (Previous work)

- Input layer:

Feedforward Neural Net Language Model (Previous work)

- Input layer:
  - 1-hot encodings (with vocab size V) of N previous words concatenated

Feedforward Neural Net Language Model (Previous work)

- Input layer:
  - 1-hot encodings (with vocab size V) of N previous words concatenated
- Projection layer:

Feedforward Neural Net Language Model (Previous work)

- Input layer:
  - 1-hot encodings (with vocab size V) of N previous words concatenated
- Projection layer:
  - Project input using a shared projection matrix into an $N \times D$ layer (D $=$ 50-200 x N)

Feedforward Neural Net Language Model (Previous work)

- Input layer:
  - 1-hot encodings (with vocab size V) of N previous words concatenated
- Projection layer:
  - Project input using a shared projection matrix into an $N \times D$ layer (D $=$ 50-200 x N)
- Hidden layer:

Feedforward Neural Net Language Model (Previous work)

- Input layer:
  - 1-hot encodings (with vocab size V) of N previous words concatenated
- Projection layer:
  - Project input using a shared projection matrix into an $N \times D$ layer (D = 50-200 x N)
- Hidden layer:
  - Densely connected from projection layer of size H (H = 50-100 x N)

Feedforward Neural Net Language Model (Previous work)

- Input layer:
  - 1-hot encodings (with vocab size V) of N previous words concatenated
- Projection layer:
  - Project input using a shared projection matrix into an $N \times D$ layer (D = 50-200 x N)
- Hidden layer:
  - Densely connected from projection layer of size H (H = 50-100 x N)
- Output layer:

Feedforward Neural Net Language Model (Previous work)

- Input layer:
  - 1-hot encodings (with vocab size V) of N previous words concatenated
- Projection layer:
  - Project input using a shared projection matrix into an $N \times D$ layer (D $=$ 50-200 x N)
- Hidden layer:
  - Densely connected from projection layer of size H (H $=$ 50-100 x N)
- Output layer:
  - Softmax (i.e. densely connected) over all vocab words (i.e. of size V)

Improvements of Current Work

- Use hierarchical (Huffman binary tree) for softmax vocab classification

Improvements of Current Work

- Use hierarchical (Huffman binary tree) for softmax vocab classification
  - Has advantage that frequent words have fewer parameters to learn (given shorter binary codes)

Improvements of Current Work

- Use hierarchical (Huffman binary tree) for softmax vocab classification
  - Has advantage that frequent words have fewer parameters to learn (given shorter binary codes)
  - Speeds up both training and testing (avoids calculating and normalizing inner products over the **entire** vocabulary)

Improvements of Current Work

- Use hierarchical (Huffman binary tree) for softmax vocab classification
  - Has advantage that frequent words have fewer parameters to learn (given shorter binary codes)
  - Speeds up both training and testing (avoids calculating and normalizing inner products over the **entire** vocabulary)
- CBOW (Continuous Bag of Words)

Improvements of Current Work

- Use hierarchical (Huffman binary tree) for softmax vocab classification
  - Has advantage that frequent words have fewer parameters to learn (given shorter binary codes)
  - Speeds up both training and testing (avoids calculating and normalizing inner products over the **entire** vocabulary)
- CBOW (Continuous Bag of Words)
  - Projection layer encodes all words (i.e. size = D) in addition to sharing projection matrix

Improvements of Current Work

- Use hierarchical (Huffman binary tree) for softmax vocab classification
  - Has advantage that frequent words have fewer parameters to learn (given shorter binary codes)
  - Speeds up both training and testing (avoids calculating and normalizing inner products over the **entire** vocabulary)
- CBOW (Continuous Bag of Words)
  - Projection layer encodes all words (i.e. size = D) in addition to sharing projection matrix
    - Means the word vectors are averaged

Improvements of Current Work

- Use hierarchical (Huffman binary tree) for softmax vocab classification
  - Has advantage that frequent words have fewer parameters to learn (given shorter binary codes)
  - Speeds up both training and testing (avoids calculating and normalizing inner products over the **entire** vocabulary)
- CBOW (Continuous Bag of Words)
  - Projection layer encodes all words (i.e. size $= D$) in addition to sharing projection matrix
    - Means the word vectors are averaged
    - Loses positional information within the window (hence bag of words)

Improvements of Current Work

- Use hierarchical (Huffman binary tree) for softmax vocab classification
  - Has advantage that frequent words have fewer parameters to learn (given shorter binary codes)
  - Speeds up both training and testing (avoids calculating and normalizing inner products over the **entire** vocabulary)
- CBOW (Continuous Bag of Words)
  - Projection layer encodes all words (i.e. size $= D$) in addition to sharing projection matrix
    - Means the word vectors are averaged
    - Loses positional information within the window (hence bag of words)
  - Input layer uses symmetric future/past window

Improvements of Current Work

- Use hierarchical (Huffman binary tree) for softmax vocab classification
  - Has advantage that frequent words have fewer parameters to learn (given shorter binary codes)
  - Speeds up both training and testing (avoids calculating and normalizing inner products over the **entire** vocabulary)
- CBOW (Continuous Bag of Words)
  - Projection layer encodes all words (i.e. size = D) in addition to sharing projection matrix
    - Means the word vectors are averaged
    - Loses positional information within the window (hence bag of words)
  - Input layer uses symmetric future/past window
- Skip-gram

Improvements of Current Work

- Use hierarchical (Huffman binary tree) for softmax vocab classification
  - Has advantage that frequent words have fewer parameters to learn (given shorter binary codes)
  - Speeds up both training and testing (avoids calculating and normalizing inner products over the **entire** vocabulary)
- CBOW (Continuous Bag of Words)
  - Projection layer encodes all words (i.e. size = D) in addition to sharing projection matrix
    - Means the word vectors are averaged
    - Loses positional information within the window (hence bag of words)
  - Input layer uses symmetric future/past window
- Skip-gram
  - Uses current word as input to classify (using log-linear classifier) surrounding words

Improvements of Current Work

- Use hierarchical (Huffman binary tree) for softmax vocab classification
  - Has advantage that frequent words have fewer parameters to learn (given shorter binary codes)
  - Speeds up both training and testing (avoids calculating and normalizing inner products over the **entire** vocabulary)
- CBOW (Continuous Bag of Words)
  - Projection layer encodes all words (i.e. size = D) in addition to sharing projection matrix
    - Means the word vectors are averaged
    - Loses positional information within the window (hence bag of words)
  - Input layer uses symmetric future/past window
- Skip-gram
  - Uses current word as input to classify (using log-linear classifier) surrounding words
  - Uses same continuous projection layer as CBOW

Improvements of Current Work

- Use hierarchical (Huffman binary tree) for softmax vocab classification
  - Has advantage that frequent words have fewer parameters to learn (given shorter binary codes)
  - Speeds up both training and testing (avoids calculating and normalizing inner products over the **entire** vocabulary)
- CBOW (Continuous Bag of Words)
  - Projection layer encodes all words (i.e. size = D) in addition to sharing projection matrix
    - Means the word vectors are averaged
    - Loses positional information within the window (hence bag of words)
  - Input layer uses symmetric future/past window
- Skip-gram
  - Uses current word as input to classify (using log-linear classifier) surrounding words
  - Uses same continuous projection layer as CBOW
  - Sample closer words more often (choose a random number from 1 to max_context and predict that many words on each side of the input word)

Graphical Representations of the Models

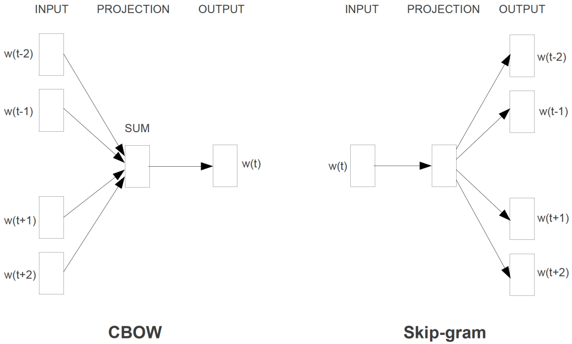


Figure 1: CBOW and Skip-gram Architectures

Testing

- Uses analogy as the testing metric (i.e. “small”:“smaller” as “big”:?) using cosine-similarity nearest neighbor

Testing

- Uses analogy as the testing metric (i.e. “small”:“smaller” as “big”:?) using cosine-similarity nearest neighbor
- Testing is all or nothing (i.e. closest vector in embedding space was either the correct answer or not)

Testing

- Uses analogy as the testing metric (i.e. “small”:“smaller” as “big”:?) using cosine-similarity nearest neighbor
- Testing is all or nothing (i.e. closest vector in embedding space was either the correct answer or not)
- Well-trained models capture notion of capital (“France”:“Paris” as “Germany”:“Berlin”)

Test Set Dimensions

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

Figure 2: Example Relationships in the Test Set

Testing Results

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

Figure 3: Results of Various Approaches

Dimensionality vs. Tokens vs. Time Tradeoff

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

Figure 4: Performance of Various Training Corpora

Dimensions Captured

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Figure 5: Analogies in the Test Set

- Scores about 60% using their criteria

Dimensions Captured

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Figure 5: Analogies in the Test Set

- Scores about 60% using their criteria
- Can improve by 10% (absolute) by averaging relationship vectors

Mikolov et al. 2013b: Distributed Representations of Words and Phrases and their Compositionality

Motivation

- Improved accuracy and training speedup using subsampling

Motivation

- Improved accuracy and training speedup using subsampling
- Better and faster Noise Contrastive Estimation (NCE) compared with hierarchical softmax

Motivation

- Improved accuracy and training speedup using subsampling
- Better and faster Noise Contrastive Estimation (NCE) compared with hierarchical softmax
- Non-compositionality of phrases (i.e. $\overrightarrow{Boston\ Globe} \neq \overrightarrow{Boston} + \overrightarrow{Globe}$)

Motivation

- Improved accuracy and training speedup using subsampling
- Better and faster Noise Contrastive Estimation (NCE) compared with hierarchical softmax
- Non-compositionality of phrases (i.e. $\overrightarrow{Boston\ Globe} \neq \overrightarrow{Boston} + \overrightarrow{Globe}$)
  - Test on these new phrasal relations: -"Montreal":"Montreal Canadiens"::"Toronto":"Toronto Maple Leafs"

Motivation

- Improved accuracy and training speedup using subsampling
- Better and faster Noise Contrastive Estimation (NCE) compared with hierarchical softmax
- Non-compositionality of phrases (i.e. $\overrightarrow{Boston\ Globe} \neq \overrightarrow{Boston} + \overrightarrow{Globe}$)
  - Test on these new phrasal relations: -"Montreal":"Montreal Canadiens"::"Toronto":"Toronto Maple Leafs"
- Explore meaningful simple vector additions

Motivation

- Improved accuracy and training speedup using subsampling
- Better and faster Noise Contrastive Estimation (NCE) compared with hierarchical softmax
- Non-compositionality of phrases (i.e. $\overrightarrow{Boston\ Globe} \neq \overrightarrow{Boston} + \overrightarrow{Globe}$)
  - Test on these new phrasal relations: -“Montreal”:“Montreal Canadiens”::“Toronto”:“Toronto Maple Leafs”
- Explore meaningful simple vector additions
  - $\overrightarrow{Russia} + \overrightarrow{river} = \overrightarrow{Volga\ River}$

Motivation

- Improved accuracy and training speedup using subsampling
- Better and faster Noise Contrastive Estimation (NCE) compared with hierarchical softmax
- Non-compositionality of phrases (i.e. $\overrightarrow{Boston\ Globe} \neq \overrightarrow{Boston} + \overrightarrow{Globe}$)
  - Test on these new phrasal relations: -“Montreal”:“Montreal Canadiens”::“Toronto”:“Toronto Maple Leafs”
- Explore meaningful simple vector additions
  - $\overrightarrow{Russia} + \overrightarrow{river} = \overrightarrow{Volga\ River}$
  - $\overrightarrow{Germany} + \overrightarrow{capital} = \overrightarrow{Berlin}$

Negative Sampling

- Task is to differentiate real co-occurrence pairs from fake ones (GAN-esque)

Negative Sampling

- Task is to differentiate real co-occurrence pairs from fake ones (GAN-esque)
- Approximately maximizes the softmax (easier to compute than Hier. Softmax)

Negative Sampling

- Task is to differentiate real co-occurrence pairs from fake ones (GAN-esque)
- Approximately maximizes the softmax (easier to compute than Hier. Softmax)
  - Can be simplified further given task demands

Negative Sampling

- Task is to differentiate real co-occurrence pairs from fake ones (GAN-esque)
- Approximately maximizes the softmax (easier to compute than Hier. Softmax)
  - Can be simplified further given task demands
- $$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I}) \right]$$

Negative Sampling

- Task is to differentiate real co-occurrence pairs from fake ones (GAN-esque)
- Approximately maximizes the softmax (easier to compute than Hier. Softmax)
  - Can be simplified further given task demands
- $$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I}) \right]$$
  - Replaces $log\ P(w_O|w_I)$ in the Skip-gram objective

Negative Sampling

- Task is to differentiate real co-occurrence pairs from fake ones (GAN-esque)
- Approximately maximizes the softmax (easier to compute than Hier. Softmax)
  - Can be simplified further given task demands
-

$$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I}) \right]$$

  - Replaces $log\ P(w_O|w_I)$ in the Skip-gram objective
  - For each word w_I:

Negative Sampling

- Task is to differentiate real co-occurrence pairs from fake ones (GAN-esque)
- Approximately maximizes the softmax (easier to compute than Hier. Softmax)
  - Can be simplified further given task demands
-

$$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I}) \right]$$

  - Replaces $log\ P(w_O|w_I)$ in the Skip-gram objective
  - For each word w_I:
  - Distinguish target word, w_O from fake samples from the noise distribution $P_n(w)$ using logistic regression for k negative samples for each 1 real sample

Negative Sampling

- Task is to differentiate real co-occurrence pairs from fake ones (GAN-esque)
- Approximately maximizes the softmax (easier to compute than Hier. Softmax)
  - Can be simplified further given task demands
- $$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I}) \right]$$
  - Replaces $log\ P(w_O|w_I)$ in the Skip-gram objective
  - For each word w_I:
  - Distinguish target word, w_O from fake samples from the noise distribution $P_n(w)$ using logistic regression for k negative samples for each 1 real sample
  - Experimental results show k=5–20 are useful for small datasets, 2–5 for large

Negative Sampling

- Task is to differentiate real co-occurrence pairs from fake ones (GAN-esque)
- Approximately maximizes the softmax (easier to compute than Hier. Softmax)
  - Can be simplified further given task demands
- $$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I}) \right]$$
  - Replaces $log\ P(w_O|w_I)$ in the Skip-gram objective
  - For each word w_I:
  - Distinguish target word, w_O from fake samples from the noise distribution $P_n(w)$ using logistic regression for k negative samples for each 1 real sample
  - Experimental results show k=5–20 are useful for small datasets, 2–5 for large
- Read Goldberg & Levy 2014 for great derivation of the method

Subsampling of Frequent Words

- Overly common words (e.g. “the”, “a”, etc.) are too common, providing less information value (as we’ll see in GloVe)

Subsampling of Frequent Words

- Overly common words (e.g. "the", "a", etc.) are too common, providing less information value (as we'll see in GloVe)
- They therefore become saturated easily (very soon into training) due to exposure frequency

Subsampling of Frequent Words

- Overly common words (e.g. “the”, “a”, etc.) are too common, providing less information value (as we’ll see in GloVe)
- They therefore become saturated easily (very soon into training) due to exposure frequency
- So subsample to counterbalance frequency effects:

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

Subsampling of Frequent Words

- Overly common words (e.g. “the”, “a”, etc.) are too common, providing less information value (as we'll see in GloVe)
- They therefore become saturated easily (very soon into training) due to exposure frequency
- So subsample to counterbalance frequency effects:

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

  - where $f(w_i)$ is word frequency and t is a threshold (around 10^-5)

Results

Method	Time [min]	Syntactic [%]	Semantic [%]	Total accuracy [%]
NEG-5	38	63	54	59
NEG-15	97	63	58	**61**
HS-Huffman	41	53	40	47
NCE-5	38	60	45	53
The following results use 10^{-5} subsampling				
NEG-5	14	61	58	60
NEG-15	36	61	61	**61**
HS-Huffman	21	52	59	55

Figure 6: Better than the original Word2Vec (faster too)

Pennington et al. 2014: GloVe: Global Vectors for Word Representation

Background and Drawbacks

- Two main approaches to learn embeddings

Background and Drawbacks

- Two main approaches to learn embeddings

1) Global matrix factorization techniques (e.g. LSA)

Background and Drawbacks

- Two main approaches to learn embeddings

1) Global matrix factorization techniques (e.g. LSA)
   - Efficiently makes use of global statistical information

Background and Drawbacks

- Two main approaches to learn embeddings

1) Global matrix factorization techniques (e.g. LSA)
    - Efficiently makes use of global statistical information
    - Poor at capturing syntactic/semantic relationships = bad vector space geometry

Background and Drawbacks

- Two main approaches to learn embeddings

1) Global matrix factorization techniques (e.g. LSA)
   - Efficiently makes use of global statistical information
   - Poor at capturing syntactic/semantic relationships = bad vector space geometry
2) Local context window techniques (e.g. Word2Vec)

Background and Drawbacks

- Two main approaches to learn embeddings

1) Global matrix factorization techniques (e.g. LSA)
   - Efficiently makes use of global statistical information
   - Poor at capturing syntactic/semantic relationships = bad vector space geometry
2) Local context window techniques (e.g. Word2Vec)
   - Good at capturing similarity = good vector space geometry

Background and Drawbacks

- Two main approaches to learn embeddings

1) Global matrix factorization techniques (e.g. LSA)
   - Efficiently makes use of global statistical information
   - Poor at capturing syntactic/semantic relationships = bad vector space geometry
2) Local context window techniques (e.g. Word2Vec)
   - Good at capturing similarity = good vector space geometry
   - Idiosyncratic and underutitilze corpus statistics due to separate context training

Matrix Factorization Approaches

- Use low-rank approximations to decompose large statistics matrices

Matrix Factorization Approaches

- Use low-rank approximations to decompose large statistics matrices
- Statistics can vary based on application domain

Matrix Factorization Approaches

- Use low-rank approximations to decompose large statistics matrices
- Statistics can vary based on application domain
  - LSA is term-document matrix (words X document_count)

Matrix Factorization Approaches

- Use low-rank approximations to decompose large statistics matrices
- Statistics can vary based on application domain
  - LSA is term-document matrix (words X document_count)
  - HAL (Hyperspace Analogue to Language) is term-term cooccurrence

Matrix Factorization Approaches

- Use low-rank approximations to decompose large statistics matrices
- Statistics can vary based on application domain
  - LSA is term-document matrix (words X document_count)
  - HAL (Hyperspace Analogue to Language) is term-term cooccurrence
- Main drawback is frequent words (e.g. “the”, “and”) have disproportionate effect unrelated to meaningful semantic relationships

Matrix Factorization Approaches

- Use low-rank approximations to decompose large statistics matrices
- Statistics can vary based on application domain
  - LSA is term-document matrix (words X document_count)
  - HAL (Hyperspace Analogue to Language) is term-term cooccurrence
- Main drawback is frequent words (e.g. "the", "and") have disproportionate effect unrelated to meaningful semantic relationships
- Several transformations had been introduced to address this issue and reduce the dynamic range while preserving information

GloVe: ***Glo***bal ***Ve***ctors

- Conditional probability matrix P (assymetric):

GloVe: ***Glo***bal ***Ve***ctors

- Conditional probability matrix P (assymetric):
  - Derived from the co-occurrence matrix X (symmetric)

GloVe: ***Glo***bal ***Ve***ctors

- Conditional probability matrix P (assymetric):
  - Derived from the co-occurrence matrix X (symmetric)
  - $P_{ij} = P(j|i) = X_{ij}/X_i$

GloVe: ***Glo***bal ***Ve***ctors

- Conditional probability matrix P (assymetric):
  - Derived from the co-occurrence matrix X (symmetric)
  - $P_{ij} = P(j|i) = X_{ij}/X_i$
  - i.e. the number of times i and j cooccur over the number of times i cooccurs with anything

GloVe: ***Glo***bal ***Ve***ctors

- Conditional probability matrix P (assymetric):
  - Derived from the co-occurrence matrix X (symmetric)
  - $P_{ij} = P(j|i) = X_{ij}/X_i$
  - i.e. the number of times i and j cooccur over the number of times i cooccurs with anything
- Addresses the disproportional effect of frequent words (e.g. "the") by using conditional probabilities

GloVe: ***Glo***bal ***Ve***ctors

- Conditional probability matrix P (assymetric):
  - Derived from the co-occurrence matrix X (symmetric)
  - $P_{ij} = P(j|i) = X_{ij}/X_i$
  - i.e. the number of times i and j cooccur over the number of times i cooccurs with anything
- Addresses the disproportional effect of frequent words (e.g. “the”) by using conditional probabilities
- Allows relationship to other words to dictate meaning which is intepretable using probe words...

Probability Relationships of Terms

Table 1: Co-occurrence probabilities for target words *ice* and *steam* with selected context words from a 6 billion token corpus. Only in the ratio does noise from non-discriminative words like *water* and *fashion* cancel out, so that large values (much greater than 1) correlate well with properties specific to ice, and small values (much less than 1) correlate well with properties specific of steam.

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Figure 7: Probability Relationships of “Ice” and “Steam”

- Allows differentiation along meaningful dimensions:

Probability Relationships of Terms

Table 1: Co-occurrence probabilities for target words *ice* and *steam* with selected context words from a 6 billion token corpus. Only in the ratio does noise from non-discriminative words like *water* and *fashion* cancel out, so that large values (much greater than 1) correlate well with properties specific to ice, and small values (much less than 1) correlate well with properties specific of steam.

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Figure 7: Probability Relationships of “Ice” and “Steam”

- Allows differentiation along meaningful dimensions:
  - Both mutually related (“water”) and unrelated (“fashion”) terms have similar ratios

Developing the Learning Objective

- Want to capture the information in P_{ik}/P_{jk}

Developing the Learning Objective

- Want to capture the information in P_{ik}/P_{jk}
  - It is a function of 3 terms: i, j, and k
    $F(w_i, w_j, \tilde{w}_k) = P_{ik}/P_{jk}$

Developing the Learning Objective

- Want to capture the information in P_{ik}/P_{jk}
  - It is a function of 3 terms: i, j, and k
    $F(w_i, w_j, \tilde{w}_k) = P_{ik}/P_{jk}$
  - Where $w \in \mathbb{R}^d$ are word vectors and $\tilde{w} \in \mathbb{R}^d$ is a separate context vector

Developing the Learning Objective

- Want to capture the information in P_{ik}/P_{jk}
  - It is a function of 3 terms: i, j, and k
    $F(w_i, w_j, \tilde{w}_k) = P_{ik}/P_{jk}$
  - Where $w \in \mathbb{R}^d$ are word vectors and $\tilde{w} \in \mathbb{R}^d$ is a separate context vector
- F could be **so** many things, so we impose constraints:

Developing the Learning Objective

- Want to capture the information in P_{ik}/P_{jk}
  - It is a function of 3 terms: i, j, and k
    $F(w_i, w_j, \tilde{w}_k) = P_{ik}/P_{jk}$
  - Where $w \in \mathbb{R}^d$ are word vectors and $\tilde{w} \in \mathbb{R}^d$ is a separate context vector
- F could be **so** many things, so we impose constraints:
  1) Linear structure on the vector space:
     $F(w_i - w_j, \tilde{w}_k) = P_{ik}/P_{jk}$

Developing the Learning Objective

- Want to capture the information in P_{ik}/P_{jk}
  - It is a function of 3 terms: i, j, and k
    $F(w_i, w_j, \tilde{w}_k) = P_{ik}/P_{jk}$
  - Where $w \in \mathbb{R}^d$ are word vectors and $\tilde{w} \in \mathbb{R}^d$ is a separate context vector
- F could be **so** many things, so we impose constraints:
  1) Linear structure on the vector space:
     $F(w_i - w_j, \tilde{w}_k) = P_{ik}/P_{jk}$
  2) Simplify the mapping from vectors (arguments) to scalar (output):
     $F((w_i - w_j)^T \tilde{w}_k) = P_{ik}/P_{jk}$

Developing the Learning Objective

- Want to capture the information in P_{ik}/P_{jk}
  - It is a function of 3 terms: i, j, and k
    $F(w_i, w_j, \tilde{w}_k) = P_{ik}/P_{jk}$
  - Where $w \in \mathbb{R}^d$ are word vectors and $\tilde{w} \in \mathbb{R}^d$ is a separate context vector
- F could be **so** many things, so we impose constraints:
  1) Linear structure on the vector space:
     $F(w_i - w_j, \tilde{w}_k) = P_{ik}/P_{jk}$
  2) Simplify the mapping from vectors (arguments) to scalar (output):
     $F((w_i - w_j)^T \tilde{w}_k) = P_{ik}/P_{jk}$
     - also isolates vector space dimensions

Developing the Learning Objective

- Want to capture the information in P_{ik}/P_{jk}
  - It is a function of 3 terms: i, j, and k
    $F(w_i, w_j, \tilde{w}_k) = P_{ik}/P_{jk}$
  - Where $w \in \mathbb{R}^d$ are word vectors and $\tilde{w} \in \mathbb{R}^d$ is a separate context vector
- F could be **so** many things, so we impose constraints:
  1) Linear structure on the vector space:
     $F(w_i - w_j, \tilde{w}_k) = P_{ik}/P_{jk}$
  2) Simplify the mapping from vectors (arguments) to scalar (output):
     $F((w_i - w_j)^T \tilde{w}_k) = P_{ik}/P_{jk}$
     - also isolates vector space dimensions
  3) Symmetry between context and word vectors (an arbitrary distinction):
     $w_i^T \tilde{w}_k + b_i + \tilde{b}_k = log(X_{ik})$

We're almost there...

- But this model weights all co-occurrences equally (and is undefined when $X_{ik} = 0$)

We're almost there...

- But this model weights all co-occurrences equally (and is undefined when $X_{ik} = 0$)
  - Just zero entries are 75–95% of X

We're almost there...

- But this model weights all co-occurrences equally (and is undefined when $X_{ik} = 0$)
  - Just zero entries are 75–95% of X
  - Rare co-occurrences are highly susceptible to noise

We're almost there...

- But this model weights all co-occurrences equally (and is undefined when $X_{ik} = 0$)
  - Just zero entries are 75–95% of X
  - Rare co-occurrences are highly susceptible to noise
    - Amplified given the use of ratios

We're almost there...

- But this model weights all co-occurrences equally (and is undefined when $X_{ik} = 0$)
  - Just zero entries are 75–95% of X
  - Rare co-occurrences are highly susceptible to noise
    - Amplified given the use of ratios
- So they cast the objective function as a least squares problem

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

We're almost there...

- But this model weights all co-occurrences equally (and is undefined when $X_{ik} = 0$)
  - Just zero entries are 75–95% of X
  - Rare co-occurrences are highly susceptible to noise
    - Amplified given the use of ratios
- So they cast the objective function as a least squares problem

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

  - where the weighting function $f(X_{ij})$ is:

$$f(x) = \begin{cases} (x/x_{\max})^\alpha & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$

Weighting Function

- Satisfies:

Weighting Function

- Satisfies:
  1) $f(0) = 0$, taking care of the ill-defined nature

Weighting Function

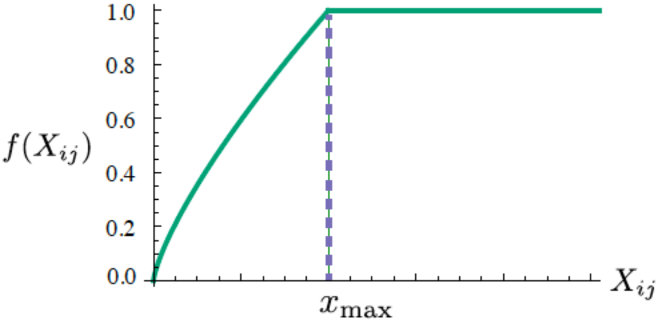
- Satisfies:
  1) $f(0) = 0$, taking care of the ill-defined nature
  2) $f(x)$ is non-decreasing, discounting rare occurrences

Weighting Function

- Satisfies:
  1) $f(0) = 0$, taking care of the ill-defined nature
  2) $f(x)$ is non-decreasing, discounting rare occurrences
  3) $f(x)$ is not huge for large xs, deflating common occurrences

Weighting Function

- Satisfies:
  1) $f(0) = 0$, taking care of the ill-defined nature
  2) $f(x)$ is non-decreasing, discounting rare occurrences
  3) $f(x)$ is not huge for large xs, deflating common occurrences
- and looks like:



Results for word analogy: SOTA(ish)

Table 2: Results on the word analogy task, given as percent accuracy. Underlined scores are best within groups of similarly-sized models; bold scores are best overall. HPCA vectors are publicly available[2]; (i)vLBL results are from (Mnih et al., 2013); skip-gram (SG) and CBOW results are from (Mikolov et al., 2013a,b); we trained SG† and CBOW† using the `word2vec` tool[3]. See text for details and a description of the SVD models.

Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	<u>67.5</u>	<u>54.3</u>	<u>60.3</u>
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	<u>64.8</u>	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	<u>80.8</u>	61.5	<u>70.3</u>
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW†	300	6B	63.6	<u>67.4</u>	65.7
SG†	300	6B	73.0	66.0	69.1
GloVe	300	6B	<u>77.4</u>	67.0	<u>71.7</u>
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	<u>**81.9**</u>	<u>**69.3**</u>	<u>**75.0**</u>

Figure 8: GloVe Analogy Results

Results for named entity recognition: SOTA(ish)

Table 4: F1 score on NER task with 50d vectors. *Discrete* is the baseline without word vectors. We use publicly-available vectors for HPCA, HSMN, and CW. See text for details.

Model	Dev	Test	ACE	MUC7
Discrete	91.0	85.4	77.4	73.4
SVD	90.8	85.7	77.3	73.7
SVD-S	91.0	85.5	77.6	74.3
SVD-L	90.5	84.8	73.6	71.5
HPCA	92.6	**88.7**	81.7	80.7
HSMN	90.5	85.7	78.7	74.7
CW	92.2	87.4	81.7	80.2
CBOW	93.1	88.2	82.2	81.1
GloVe	**93.2**	88.3	**82.9**	**82.2**

Figure 9: Named Entity Recognition on Various Tasks

Peters et al. 2018: Deep contextualized word representations (aka ELMo)

Background and Approach

- Wants to capture both syntax and semantics as previously done

Background and Approach

- Wants to capture both syntax and semantics as previously done
- Wants to capture variance across contexs (i.e. polysemy) *not* previously done

Background and Approach

- Wants to capture both syntax and semantics as previously done
- Wants to capture variance across contexs (i.e. polysemy) *not* previously done
  - Tokens are assigned representations based on entire sentence

Background and Approach

- Wants to capture both syntax and semantics as previously done
- Wants to capture variance across contexs (i.e. polysemy) *not* previously done
  - Tokens are assigned representations based on entire sentence
  - Uses bidirectional LSTM with language modeling (LM) objective

Background and Approach

- Wants to capture both syntax and semantics as previously done
- Wants to capture variance across contexs (i.e. polysemy) *not* previously done
    - Tokens are assigned representations based on entire sentence
    - Uses bidirectional LSTM with language modeling (LM) objective
    - Thus, ELMo (***E***mbeddings from ***L***anguage ***Mo***dels)

Strengths

- ELMo representations are deep (a function of all biLM layers, not just top layer)

Strengths

- ELMo representations are deep (a function of all biLM layers, not just top layer)
- Allows distributed meaning encoding

Strengths

- ELMo representations are deep (a function of all biLM layers, not just top layer)
- Allows distributed meaning encoding
  - Higher states capture context-dependent meaning (good for WSD)

Strengths

- ELMo representations are deep (a function of all biLM layers, not just top layer)
- Allows distributed meaning encoding
  - Higher states capture context-dependent meaning (good for WSD)
  - Lower states capture aspects of syntax (good for POS tagging)

Strengths

- ELMo representations are deep (a function of all biLM layers, not just top layer)
- Allows distributed meaning encoding
  - Higher states capture context-dependent meaning (good for WSD)
  - Lower states capture aspects of syntax (good for POS tagging)
  - With access to all this information, downstream models can select the relevant dimensions of information for the task (i.e. they are transferable)

Method

- Start with forward and backward LMs

Method

- Start with forward and backward LMs
  - Forward model: $p(t_1, t_2, ..., t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, ..., t_{k-1})$

Method

- Start with forward and backward LMs
  - Forward model: $p(t_1, t_2, ..., t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, ..., t_{k-1})$
  - Backward model: $p(t_1, t_2, ..., t_N) = \prod_{k=1}^{N} p(t_k | t_{k+1}, t_{k+2}, ... t_N)$

Method

- Start with forward and backward LMs
  - Forward model: $p(t_1, t_2, ..., t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, ..., t_{k-1})$
  - Backward model: $p(t_1, t_2, ..., t_N) = \prod_{k=1}^{N} p(t_k | t_{k+1}, t_{k+2}, ...t_N)$
- Jointly maximize their log likelihoods:

Method

- Start with forward and backward LMs
  - Forward model: $p(t_1, t_2, ..., t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, ..., t_{k-1})$
  - Backward model: $p(t_1, t_2, ..., t_N) = \prod_{k=1}^{N} p(t_k | t_{k+1}, t_{k+2}, ...t_N)$
- Jointly maximize their log likelihoods:
  - $$\sum_{k=1}^{N} \Big(\log p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \Big)$$

ELMo's Magic

- ...

ELMo's Magic

- ...
- Just combine all the representations computed for each token into a single vector!

each token t_k, a L-layer biLM computes a set of $2L+1$ representations

$$\begin{aligned} R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \ldots, L\} \\ &= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\}, \end{aligned}$$

where $\mathbf{h}_{k,0}^{LM}$ is the token layer and $\mathbf{h}_{k,j}^{LM} = [\overrightarrow{\mathbf{h}}_{k,j}^{LM}; \overleftarrow{\mathbf{h}}_{k,j}^{LM}]$, for each biLSTM layer.

Choosing a Useful Representation

- Simplest case is to simply select the top layer (as in previous work)

Choosing a Useful Representation

- Simplest case is to simply select the top layer (as in previous work)
- ***Or***, for each task, adjust the weighting function of each layer:

$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}.$$

Choosing a Useful Representation

- Simplest case is to simply select the top layer (as in previous work)
- ***Or***, for each task, adjust the weighting function of each layer:

$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}.$$

  - where s^{task} are softmax-normalized weights and γ^{task} allows the downstream model to scale the ELMo vector.

Choosing a Useful Representation

- Simplest case is to simply select the top layer (as in previous work)
- ***Or***, for each task, adjust the weighting function of each layer:

$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}.$$

  - where s^{task} are softmax-normalized weights and γ^{task} allows the downstream model to scale the ELMo vector.
  - sometimes it helps to apply layer normalization before weighting

Applying the Representations Downstream

- Use the pre-trained biLM and a specified architecture for target task

Applying the Representations Downstream

- Use the pre-trained biLM and a specified architecture for target task
  1) Run pre-trained (frozen) biLM and store representations for each word

Applying the Representations Downstream

- Use the pre-trained biLM and a specified architecture for target task
  1) Run pre-trained (frozen) biLM and store representations for each word
  2) Let target task model learn a linear combination of these representations

Applying the Representations Downstream

- Use the pre-trained biLM and a specified architecture for target task
  1) Run pre-trained (frozen) biLM and store representations for each word
  2) Let target task model learn a linear combination of these representations
- Can include (by concatenation) ELMo representations both in the input (concatenated with a context-independent embedding) and output (concatenated with the top hidden layer)

Applying the Representations Downstream

- Use the pre-trained biLM and a specified architecture for target task
  1) Run pre-trained (frozen) biLM and store representations for each word
  2) Let target task model learn a linear combination of these representations
- Can include (by concatenation) ELMo representations both in the input (concatenated with a context-independent embedding) and output (concatenated with the top hidden layer)
- Authors find it helpful to add "a moderate amount" of dropout

Applying the Representations Downstream

- Use the pre-trained biLM and a specified architecture for target task
  1) Run pre-trained (frozen) biLM and store representations for each word
  2) Let target task model learn a linear combination of these representations
- Can include (by concatenation) ELMo representations both in the input (concatenated with a context-independent embedding) and output (concatenated with the top hidden layer)
- Authors find it helpful to add “a moderate amount” of dropout
- Also some cases regularization helped (adding $\lambda||w||_2^2$ to the loss)

Results: SOTA!

Task	Previous SOTA		Our baseline	ELMo + baseline	Increase (absolute/ relative)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

Table 1: Test set comparison of ELMo enhanced neural models with state-of-the-art single model baselines across six benchmark NLP tasks. The performance metric varies across tasks – accuracy for SNLI and SST-5; F_1 for SQuAD, SRL and NER; average F_1 for Coref. Due to the small test sizes for NER and SST-5, we report the mean and standard deviation across five runs with different random seeds. The "increase" column lists both the absolute and relative improvements over our baseline.

Figure 10: ELMo's Improvements Across the board

Analysis

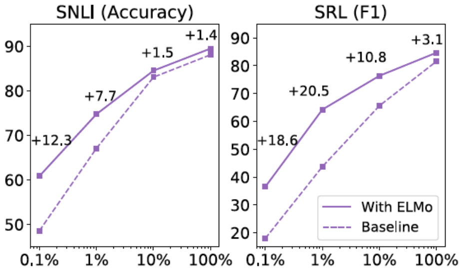


Figure 1: Comparison of baseline vs. ELMo performance for SNLI and SRL as the training set size is varied from 0.1% to 100%.

Figure 11: ELMo is very useful for small training corpora