"Vivemos em uma sociedade extremamente dependente de ciência e tecnologia, onde quase ninguém sabe coisa alguma sobre ciência e tecnologia" (Carl Sagan).

Embeddings de Texto

Paulo Ricardo Lisboa de Almeida

O Problema

“A inteligência artificial é um grande campo, e este é um grande livro. Tentamos explorar toda a extensão do assunto, que abrange lógica, probabilidade e matemática do contínuo, além de percepção, raciocínio, aprendizado, ação e, ainda, tudo o que se refere à eletrônica, ...”

Russell, S. J., Norvig, P., Davis. Artificial Intelligence: A Modern Approach. 2009.

O Problema

“A inteligência artificial é um grande campo, e este é um grande livro. Tentamos explorar toda a extensão do assunto, que abrange lógica, probabilidade e matemática do contínuo, além de percepção, raciocínio, aprendizado, ação e, ainda, tudo o que se refere à eletrônica, ...”

Como segmentar o texto?

Como representar as palavras em vetores em um espaço?

Russell, S. J., Norvig, P., Davis. Artificial Intelligence: A Modern Approach. 2009.

Solução Ingênua

O que acontece se cortarmos o texto por espaços (e afins), e dar um número para cada palavra?

```
1       2              3       4  5     6       7      8   9   4  5
A inteligência artificial é um grande campo, e este é um
   6       7          8         9       10  11    12     13    14
grande livro. Tentamos explorar toda a extensão do assunto,
 15     16       17             18       8       19      13     20
que abrange lógica, probabilidade e matemática do contínuo
```

Solução Ingênua

```
1       2              3       4  5     6       7      8    9    4  5
A inteligência artificial é um grande campo, e este é um
   6       7          8         9       10  11     12      13     14
grande livro. Tentamos explorar toda a extensão do assunto,
 15     16       17              18       8       19      13      20
que abrange lógica, probabilidade e matemática do contínuo
```

Quebrar somente em espaços é suficiente? E vírgulas? E nomes compostos (Rio de Janeiro)? E demais caracteres de quebra?

Solução Ingênua

```
1       2              3       4  5     6       7      8    9    4  5
A inteligência artificial é um grande campo, e este é um
   6       7          8         9       10  11     12      13     14
grande livro. Tentamos explorar toda a extensão do assunto,
 15      16       17              18       8       19      13      20
que abrange lógica, probabilidade e matemática do contínuo
```

Quebrar somente em espaços é suficiente? E vírgulas? E nomes compostos (Rio de Janeiro)? E demais caracteres de quebra?

O que acontece se descobrirmos ou surgir uma palavra nova?

Solução Ingênua

```
1       2              3       4  5     6       7      8    9    4  5
A inteligência artificial é um grande campo, e este é um
   6       7          8         9       10  11     12      13     14
grande livro. Tentamos explorar toda a extensão do assunto,
 15      16       17              18       8       19      13      20
que abrange lógica, probabilidade e matemática do contínuo
```

Quebrar somente em espaços é suficiente? E vírgulas? E nomes compostos (Rio de Janeiro)? E demais caracteres de quebra?

O que acontece se descobrirmos ou surgir uma palavra nova?

Por que matemática está mais longe de lógica do que de probabilidade (pela distância L2)?

Solução Ingênua

```
1       2              3       4  5     6       7      8   9   4  5
A inteligência artificial é um grande campo, e este é um
   6       7          8         9       10  11     12      13     14
grande livro. Tentamos explorar toda a extensão do assunto,
 15     16       17              18       8       19      13     20
que abrange lógica, probabilidade e matemática do contínuo
```

Quebrar somente em espaços é suficiente? E vírgulas? E nomes compostos (Rio de Janeiro)? E demais caracteres de quebra?

O que acontece se descobrirmos ou surgir uma palavra nova?

Por que matemática está mais longe de lógica do que de probabilidade (pela distância L2)?

Palavras relacionadas deveriam estar próximas no espaço (cozinha, cozinhar, cozinheiro, cozido, ...).

Transformando em Tokens (*Tokenization*)

Poderíamos considerar apenas os caracteres individuais para separar as palavras.

Transformando em Tokens (*Tokenization*)

Poderíamos considerar apenas os caracteres individuais para separar as palavras.

- Flexível, mas obriga a rede a descobrir como montar palavras válidas.

Token: grupo de poucos caracteres. Pode incluir:

- Palavras inteiras, desde que sejam comuns.
- Fragmentos de palavras.
- Caracteres.

Ideia

1. Comece com uma lista de caracteres individuais.
2. Verifique a combinação de par de tokens mais comum.
3. Adicione a combinação no dicionário de tokens.
4. Enquanto não atingir critério de parada (máximo de tokens, número de ocorrências mínimo do token mais comum ...), volte para 2.

Sennrich, Rico, Barry Haddow, and Alexandra Birch. "Neural machine translation of rare words with subword units." 2015.

Neural Machine Translation of Rare Words with Subword Units

Rico Sennrich and Barry Haddow and Alexandra Birch
School of Informatics, University of Edinburgh
{rico.sennrich,a.birch}@ed.ac.uk,bhaddow@inf.ed.ac.uk

Abstract

Neural machine translation (NMT) models typically operate with a fixed vocabulary, but translation is an open-vocabulary problem. Previous work addresses the translation of out-of-vocabulary words by backing off to a dictionary. In this paper, we introduce a simpler and more effective approach, making the NMT model capable of open-vocabulary translation by encoding rare and unknown words as sequences of subword units. This is based on the intuition that various word classes are translatable via smaller units than words, for instance names (via character copying or transliteration), compounds (via compositional translation), and cognates and loanwords (via phonological and morphological transformations). We discuss the suitability of different word segmentation techniques, including simple character n-gram models and a segmentation based on the *byte pair encoding* compression algorithm, and empirically show that subword models improve over a back-off dictionary baseline for the WMT 15 translation tasks English→German and English→Russian by up to 1.1 and 1.3 BLEU, respectively.

1 Introduction

Neural machine translation has recently shown impressive results (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015). However, the translation of rare words is an open problem. The vocabulary of neural models is typically limited to 30 000–50 000 words, but translation is an open-vocabulary problem, and especially for languages with productive word formation processes such as agglutination and compounding, translation models require mechanisms that go below the word level. As an example, consider compounds such as the German *Abwasserbehandlungsanlage* 'sewage water treatment plant', for which a segmented, variable-length representation is intuitively more appealing than encoding the word as a fixed-length vector.

For word-level NMT models, the translation of out-of-vocabulary words has been addressed through a back-off to a dictionary look-up (Jean et al., 2015; Luong et al., 2015b). We note that such techniques make assumptions that often do not hold true in practice. For instance, there is not always a 1-to-1 correspondence between source and target words because of variance in the degree of morphological synthesis between languages, like in our introductory compounding example. Also, word-level models are unable to translate or generate unseen words. Copying unknown words into the target text, as done by (Jean et al., 2015; Luong et al., 2015b), is a reasonable strategy for names, but morphological changes and transliteration is often required, especially if alphabets differ.

We investigate NMT models that operate on the level of subword units. Our main goal is to model open-vocabulary translation in the NMT network itself, without requiring a back-off model for rare words. In addition to making the translation process simpler, we also find that the subword models achieve better accuracy for the translation of rare words than large-vocabulary models and back-off dictionaries, and are able to productively generate new words that were not seen at training time. Our analysis shows that the neural networks are able to learn compounding and transliteration from subword representations.

This paper has two main contributions:

- We show that open-vocabulary neural ma-

The research presented in this publication was conducted in cooperation with Samsung Electronics Polska sp. z o.o. - Samsung R&D Institute Poland.

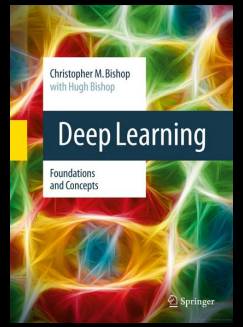
Tokens: a c d e f i k l o p P r s t

Exemplo

Peter Piper picked a peck of pickled peppers

1. Comece com uma lista de caracteres individuais.
2. Verifique a combinação de par de tokens mais comum.
3. Adicione a combinação no dicionário de tokens.
4. Enquanto não atingir critério de parada.

Bishop, C. M., Bishop, H. Deep Learning: Foundations and Concepts. 2023.



Exemplo

Tokens: a c d e f i k l o p P r s t pe

Peter Piper picked a peck of pickled peppers

Peter Piper picked a peck of pickled peppers

1. Comece com uma lista de caracteres individuais.
2. Verifique a combinação de par de tokens mais comum.
3. Adicione a combinação no dicionário de tokens.
4. Enquanto não atingir critério de parada.

Exemplo

Tokens: a c d e f i k l o p P r s t pe ck

Peter Piper picked a peck of pickled peppers

Peter Piper picked a peck of pickled peppers

Peter Piper picked a peck of pickled peppers

1. Comece com uma lista de caracteres individuais.
2. Verifique a combinação de par de tokens mais comum.
3. Adicione a combinação no dicionário de tokens.
4. Enquanto não atingir critério de parada.

Exemplo

Tokens: a c d e f i k l o p P r s t pe ck pi

Peter Piper picked a peck of pickled peppers

Peter Piper picked a peck of pickled peppers

Peter Piper picked a peck of pickled peppers

Peter Piper picked a peck of pickled peppers

1. Comece com uma lista de caracteres individuais.
2. Verifique a combinação de par de tokens mais comum.
3. Adicione a combinação no dicionário de tokens.
4. Enquanto não atingir critério de parada.

Exemplo

Tokens: a c d e f i k l o p P r s t pe ck pi per

Peter Piper picked a peck of pickled peppers

Peter Piper picked a peck of pickled peppers

Peter Piper picked a peck of pickled peppers

Peter Piper picked a peck of pickled peppers

Peter Piper picked a peck of pickled peppers

1. Comece com uma lista de caracteres individuais.
2. Verifique a combinação de par de tokens mais comum.
3. Adicione a combinação no dicionário de tokens.
4. Enquanto não atingir critério de parada.

Problemas

Como segmentar o texto? 

Como representar as palavras em vetores em um espaço?

Word2Vec

Vocabulário: a c d e f i k l o p P r s t pe ck pi per

Peter Piper picked a peck of pickled peppers

```
      a c d e f i k l o p P r s t pe ck pi per
x0 = [0 0 0 0 0 0 0 0 0 1 0 0 0 0  0  0  0]
x1 = [0 0 0 1 0 0 0 0 0 0 0 0 0 0  0  0  0]
x2 = [0 0 0 0 0 0 0 0 0 0 0 0 1 0  0  0  0]
…
```

One-hot encoding

Word2Vec

Vocabulário: a c d e f i k l o p P r s t pe ck pi per

Peter Piper picked a peck of pickled peppers

```
      a c d e f i k l o p P r s t pe ck pi per
x0 = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0  0  0  0]
x1 = [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0  0  0  0]
x2 = [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0  0  0  0]
…
```

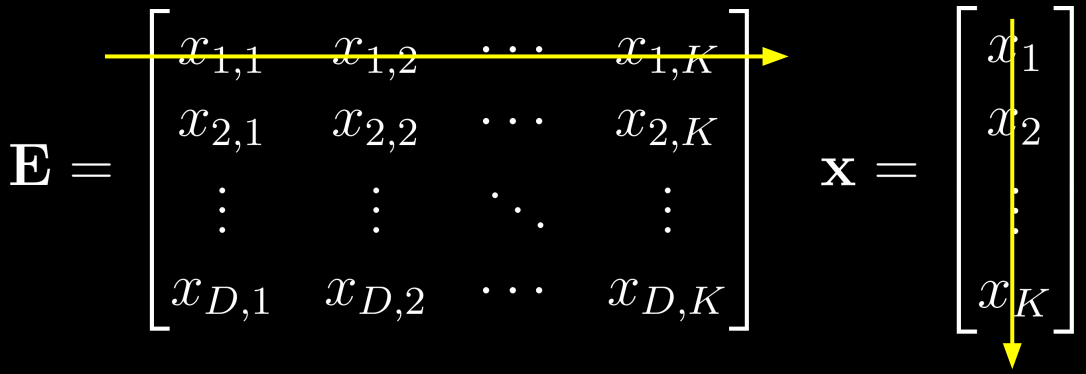
Utilizamos uma matriz de embedding $\mathbf{E}$, de tamanho $D \times K$.
K é o tamanho do Vocabulário
D é o número de dimensões do embedding.

Cada vetor é projetado no espaço de embedding através de $\mathbf{v}_i = \mathbf{E}\mathbf{x}_i$.

Exemplo

$$\mathbf{E} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,K} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ x_{D,1} & x_{D,2} & \cdots & x_{D,K} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_K \end{bmatrix}$$

Exemplo



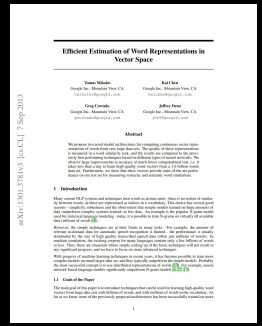
Devido ao one-hot encoding, o vetor resultante é dado pela coluna correspondente em $\mathbf{E}$, sem precisar realizar cálculos!

Word2vec

Uma técnica clássica para aprender a matriz $\mathbf{E}$ é utilizar o word2vec.

- MLP de duas camadas.
  - Entrada e saída do tamanho do vocabulário.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. Efficient estimation of word representations in vector space. 2013.



Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

Abstract

We propose two new model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

1 Introduction

Many current NLP systems and techniques treat words as atomic units - there is no notion of similarity between words, as these are represented as indices in a vocabulary. This choice has several good reasons - simplicity, robustness and the observation that simple models trained on huge amounts of data outperform complex systems trained on less data. An example is the popular N-gram model used for statistical language modeling - today, it is possible to train N-grams on virtually all available data (trillions of words [3]).

However, the simple techniques are at their limits in many tasks. For example, the amount of relevant in-domain data for automatic speech recognition is limited - the performance is usually dominated by the size of high quality transcribed speech data (often just millions of words). In machine translation, existing corpora of parallel sentences contain up to a few billion words or less. Thus, there are situations where simple scaling up of the basic techniques will not result in any significant progress, and we have to focus on more advanced techniques.

With progress of machine learning techniques in recent years, it has become possible to train more complex models on much larger data set, and they typically outperform the simple models. Probably the most successful concept is to use distributed representations of words [10]. For example, neural network based language models significantly outperform N-gram models [1] [2] [5].

1.1 Goals of the Paper

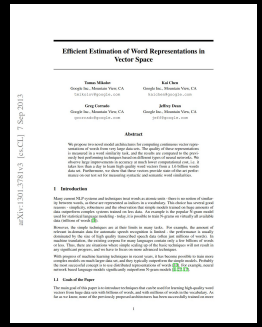
The main goal of this paper is to introduce techniques that can be used for learning high-quality word vectors from huge data sets with billions of words, and with millions of words in the vocabulary. As far as we know, none of the previously proposed architectures has been successfully trained on more

1

arXiv:1301.3781v3 [cs.CL] 7 Sep 2013

Word2vec

Mikolov, T., Chen, K., Corrado, G., & Dean, J. Efficient estimation of word representations in vector space. 2013.



Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

1 Introduction

Many current NLP systems and techniques treat words as atomic units - there is no notion of similarity between words, as these are represented as indices in a vocabulary. This choice has several good reasons - simplicity, robustness and the observation that simple models trained on huge amounts of data outperform complex systems trained on less data. An example is the popular N-gram model used for statistical language modeling - today it is possible to train N-grams on virtually all available data (trillions of words [3]).

However, the simple techniques are at their limits in many tasks. For example, the amount of relevant in-domain data for automatic speech recognition is limited - the performance is usually dominated by the size of high quality transcribed speech data (often just millions of words). In machine translation, the existing corpora for many languages contain only a few billions of words or less. Thus, there are situations where simple scaling up of the basic techniques will not result in any significant progress, and we have to focus on more advanced techniques.

With progress of machine learning techniques in recent years, it has become possible to train more complex models on much larger data set, and they typically outperform the simple models. Probably the most successful concept is to use distributed representations of words [10]. For example, neural network based language models significantly outperform N-gram models [1] [2] [5].

1.1 Goals of the Paper

The main goal of this paper is to introduce techniques that can be used for learning high-quality word vectors from huge data sets with billions of words, and with millions of words in the vocabulary. As far as we know, none of the previously proposed architectures has been successfully trained on more

1

arXiv:1301.3781v3 [cs.CL] 7 Sep 2013

Uma técnica clássica para aprender a matriz $\mathbf{E}$ é utilizar o word2vec.

- MLP de duas camadas.
  - Entrada e saída do tamanho do vocabulário.
- Conjunto de treinamento construído considerando uma janela deslizante de palavras de tamanho M.
  - Comum utilizar M=5.

Word2vec - Treinamento

Continuous bag of words

Considerando M=5

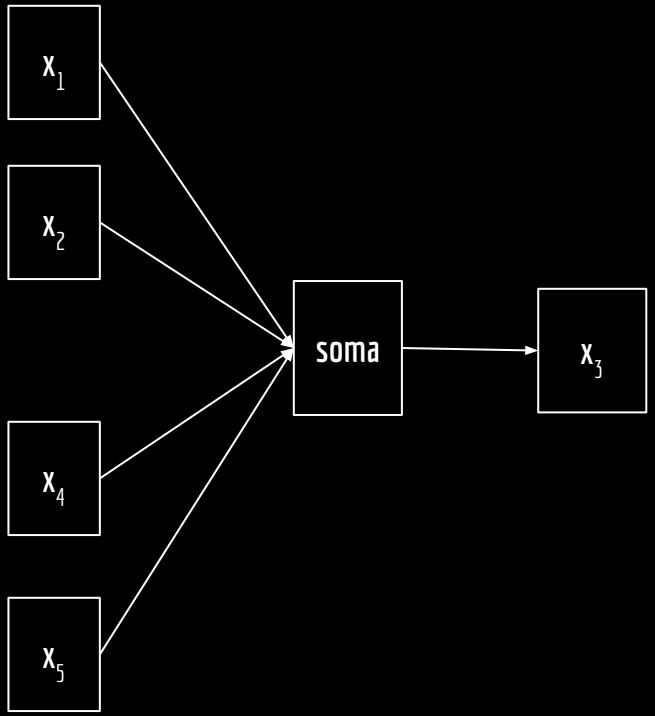
Dada uma string de treinamento, são enviados os tokens de índice 1,2,4 e 5.

O objetivo de rede é prever o token central (índice 3).

A função de loss é a soma das entradas.

A soma deve ser igual ao token central.

Word2vec - Treinamento



Self-supervised Learning

Treinamento auto-supervisionado.

Não precisamos rotular os dados, já que o treino é realizado ocultando parte do dado existente.

Mas note que o texto original precisa fazer sentido. Logo, alguém precisou escrever!

Word2vec

Depois de treinada, a transposta dos pesos da segunda camada da rede são usados como a Matriz $\mathbf{E}$.

Word2vec

Depois de treinada, a transposta dos pesos da segunda camada da rede são usados como a Matriz $\mathbf{E}$.

Os pesos da primeira camada são usados se usarmos o método skip-grams.

Pesquise como funciona o método baseado em skip-grams.

Semântica

Paris está para França assim como Itália está para ...

v(Paris) - **v**(França) + **v**(Itália) ≃ **v**(Roma)

Visualização

Veja um exemplo de embeddings em https://projector.tensorflow.org

Problemas

Como segmentar o texto? 

Como representar as palavras em vetores em um espaço? 

Uso

Os embeddings são comumente usados atualmente como um pré-processamento.

Entrada para uma rede profunda, como redes baseadas em transformers (próximas aulas).

Podemos usar de forma fixa (pré-treinado), ou como uma camada inicial treinável.

Ideias

As ideias são aplicáveis em outros tipos de dados.

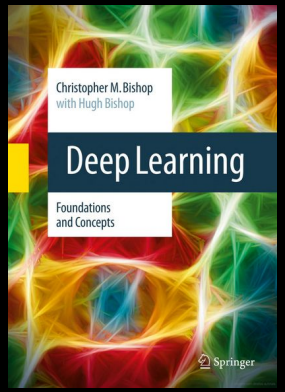
- Imagens.
- Áudio.
- Códigos fonte.
- ...

Exercícios

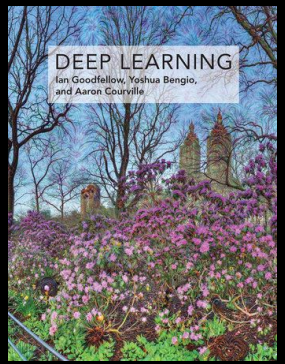
1. Execute os exemplos disponibilizados no Google Colab.

Referências

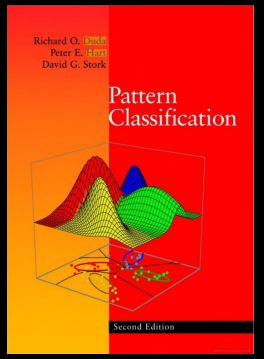
Bishop, C. M., Bishop, H. Deep Learning: Foundations and Concepts. 2023.



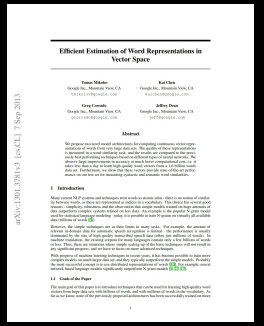
Goodfellow, I., Bengio, Y., Courville, A. Deep Learning. 2016.



Duda, R. O., Hart, P. E., Stork, D. G. Pattern Classification. 2012.



Mikolov, T., Chen, K., Corrado, G., & Dean, J. Efficient estimation of word representations in vector space. 2013.


Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kai@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

1 Introduction

Many current NLP systems and techniques treat words as atomic units - there is no notion of similarity between words, as these are represented as indices in a vocabulary. This choice has several good reasons - simplicity, robustness and the observation that simple models trained on huge amounts of data outperform complex systems trained on less data. An example is the popular N-gram model used for statistical language modeling - today it is possible to train N-grams on virtually all available data (trillions of words [3]).

However, the simple techniques are at their limits in many tasks. For example, the amount of relevant in-domain data for automatic speech recognition is limited - the performance is usually dominated by the size of high quality transcribed speech data (often just millions of words). In machine translation, the existing corpora for many languages contain only a few billions of words or less. Thus, there are situations where simple scaling up of the basic techniques will not result in any significant progress, and we have to focus on more advanced techniques.

With progress of machine learning techniques in recent years, it has become possible to train more complex models on much larger data sets, and they typically outperform the simple models. Probably the most successful concept is to use distributed representations of words [10]. For example, neural network based language models significantly outperform N-gram models [1] [2] [5].

1.1 Goals of the Paper

The main goal of this paper is to introduce techniques that can be used for learning high-quality word vectors from huge data sets with billions of words, and with millions of words in the vocabulary. As far as we know, none of the previously proposed architectures has been successfully trained on more

1

arXiv:1301.3781v3 [cs.CL] 7 Sep 2013

JURAFSKY, Daniel; MARTIN, James H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models. 3. Disponível em: https://web.stanford.edu/~jurafsky/slp3.

Licença