Word Sense Embedded in Geometric Spaces
From Induction to Applications using Machine Learning

MIKAEL KÅGEBÄCK

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2016
Word Sense Embedded in Geometric Spaces
From Induction to Applications using Machine Learning

MIKAEL KÅGEBÄCK
Word Sense Embedded in Geometric Spaces
From Induction to Applications using Machine Learning
MIKAEL KÅGEBÄCK

ISBN 1652-876X

© MIKAEL KÅGEBÄCK, 2016

Thesis for the degree of Licentiate of Engineering
ISSN 1652-876X
Technical Report No. 156L
Department of Computer Science and Engineering

Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Sweden
Telephone: +46 (0)31-772 1000

Cover:
2D projection of geometric representations of word tokens corresponding to different senses of the word type paper.

Chalmers Reproservice
Gothenburg, Sweden 2016
To Sandra and Alva.
ABSTRACT

Words are not detached individuals but part of an interconnected web of related concepts, and to capture the full complexity of this web they need to be represented in a way that encapsulates all the semantic and syntactic facets of the language. Further, to enable computational processing they need to be expressed in a consistent manner so that common properties, e.g. plurality, are encoded in a similar way for all words sharing that property. In this thesis dense real valued vector representations, i.e. word embeddings, are extended and studied for their applicability to natural language processing (NLP). Word embeddings of two distinct flavors are presented as part of this thesis, sense aware word representations where different word senses are represented as distinct objects, and grounded word representations that are learned using multi-agent deep reinforcement learning to explicitly express properties of the physical world while the agents learn to play Guess who?. The empirical usefulness of word embeddings is evaluated by employing them in a series of NLP related applications, i.e. word sense induction, word sense disambiguation, and automatic document summarisation. The results show great potential for word embeddings by outperforming previous state-of-the-art methods in two out of three applications, and achieving a statistically equivalent result in the third application but using a much simpler model than previous work.
ACKNOWLEDGEMENTS

The writing of this thesis, and doing the research behind it, has been one of the most rewarding experiences so far in my professional career, a fact that is to a large extent due to the people that I had the pleasure of sharing this time with. Hence, I would like to take this opportunity to extend my warmest thank you.

First to my main supervisor Devdatt Dubhashi, thank you for giving me this opportunity, your guidance and inspiration, and prompt feedback 24 hours a day (yes, even in the middle of the night, this happened, not sure how you pull it off). My co-supervisors Richard Johansson and Shalom Lappin for welcome input and guidance, and Gerardo Schneider for keeping me on the narrow path in the role of examiner. Further, I would like to extend my warmest gratitude towards Richard Socher for taking the time and effort to travel here from San Fransisco and leading the discussion during my licentiate seminar.

Next my room mates, with whom I have had countless discussions on every subject worth discussing, Fredrik D. Johansson you have been a great colleague and I have learned a lot from our hours in front of the whiteboard, and Olof Mogren who co-authored my first paper and share my passion for Deep Learning. My friends and colleagues, Nina Tahmasebi for trying to teach me how to write a scientific paper, Hans Salomonsson for coding like a ninja, my star master student Emilio Jorge, and Prasanth Kolachina for digging up those hard to find references. Further, my lab mates Chien-Chung, Christos, Aristide, Vinay, Peter, Azam, Joel, and Alexander. For helping me on the administrative side I want to thank Rebecca Cyren, Jonna Amgard and Eva Axelsson, and Peter Dybjer for signing the important papers.

Finally, all my love and gratitude goes to my wonderful family for making this dream possible.
LIST OF PUBLICATIONS

This thesis is based on the following manuscripts.

Paper I
“Neural context embeddings for automatic discovery of word senses”.
*Proceedings of NAACL-HLT*, pp. 25–32

Paper II
Guess Who? and Inventing a Grounded Language as a Consequence”.
*NIPS workshop on deep reinforcement learning*

Paper III
using a Bidirectional LSTM”. *5th Workshop on Cognitive Aspects
of the Lexicon (CogALex)*. Association for Computational Linguistics

Paper IV
“Extractive summarization using continuous vector space models”.
*Proceedings of the 2nd Workshop on Continuous Vector Space Models
and their Compositionality (CVSC)@ EACL*, pp. 31–39

Paper V
Summarization by Aggregating Multiple Similarities”. *Proceedings of
Recent Advances in Natural Language Processing*, pp. 451–457

The following manuscripts have been published, but are not included in this work.

Paper VI
N. Tahmasebi, L. Borin, G. Capannini, D. Dubhashi, P. Exner, M.
Forsberg, G. Gossen, F. D. Johansson, R. Johansson, M. Kågebäck, O.
for a knowledge-based culturomics. *International Journal on Digital
**CONTRIBUTION SUMMARY**

<table>
<thead>
<tr>
<th>Paper I</th>
<th>I am the main author, developed the main technical contribution, and wrote about 50% of the manuscript and experiments.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Paper II</strong></td>
<td>I initiated the project, supervised the main author, and contributed towards the manuscript (abstract, introduction, and conclusions) and the technical contribution.</td>
</tr>
<tr>
<td><strong>Paper III</strong></td>
<td>I developed the main technical contribution and wrote 90% of the manuscript.</td>
</tr>
<tr>
<td><strong>Paper IV</strong></td>
<td>I am the main author, developed the main technical contribution, and wrote about 80% of the manuscript and experiments.</td>
</tr>
<tr>
<td><strong>Paper V</strong></td>
<td>I am the second author, contributed with the idea of multiplicative interaction between kernels, and wrote about 20% of the manuscript and experiments.</td>
</tr>
</tbody>
</table>
CONTENTS

Abstract i
Acknowledgements iii
List of publications v
Contribution summary vii
Contents ix

I Extended Summary 1

1 Introduction 3
  1.1 Main Contributions of this Thesis ........................................... 3

2 Embedding Words in Geometric Spaces 5
  2.1 Basic Vector Representations ...................................................... 5
    2.1.1 One-Hot Vectors ................................................................. 5
    2.1.2 Feature Vectors ................................................................. 5
    2.1.3 Bag-of-Words Vectors ........................................................... 6
  2.2 Dense Real Valued Vectors Representations ..................................... 6
    2.2.1 CW Vectors ............................................................................. 6
    2.2.2 Continuous Skip-gram ............................................................ 7
    2.2.3 Global Vectors for Word Representation .................................... 7
  2.3 Sense Aware Word Embeddings ....................................................... 7
    2.3.1 Instance-Context Embeddings .................................................. 8
  2.4 Embedding Grounded Concepts ...................................................... 8

3 Applications of Word Embeddings to NLP 11
  3.1 Word Sense Induction ................................................................. 11
  3.2 Word Sense Disambiguation .......................................................... 11
  3.3 Automatic Multi-Document Summarisation ...................................... 12
    3.3.1 Extractive Summarisation ....................................................... 12
    3.3.2 Comparing Sentences ............................................................. 13
    3.3.3 MULTISUM .............................................................................. 13

4 Future Direction of Research 15

References 16

II Publications 19
Part I
Extended Summary
Chapter 1

Introduction

Speakers of a language tend to have a very personal relationship to the words that make up that language. Every word has a different feel to it, that somehow encapsulates the essence of what that word means, but how do you translate this feeling into a mathematical representation that can be used in computation? When engineers design communication protocols they tend to keep the symbols orthogonal and context independent which makes the protocols compact and unambiguous. However, when humans communicate they make no such effort. Instead, several words may have related or identical meaning and most words encode different senses depending on the context in which they are being used. To further add to the complexity, these idiosyncrasies follow no well defined set of rules which makes human language very difficult to comprehend in an algorithmic way.

In this thesis different ways of representing words as dense real valued vectors are studied. Starting with neural word embeddings, which are by-products from predictive modeling of word co-occurrence statistics, that are able to encode similarities between words as distances in a geometric space. Continuing with sense aware word embeddings which provide several different representations for each word, i.e. one per word sense. Finally, a first step is taken towards grounded word representations by letting agents invent their own language to communicate concepts found in images.

To provide evidence of the usefulness of neural word embeddings in real applications, a study on the effectiveness of these representations in the following applications are conducted. (1) The automatic creation of a lexicon given a text corpus which is referred to as *Word Sense Induction* (WSI), (2) the related task of *Word Sense Disambiguation* (WSD) which is the problem of assigning a sense label, from a predefined set of senses, to a word token in a text, and (3) automatic summarisation of one or more documents by picking sentences as to cover as much of the central information in the corpus as possible, i.e. *extractive multi-document summarisation*.

1.1 Main Contributions of this Thesis

- A method for creating sense aware word embeddings, Section 2.3.1, is presented and used to do WSI, Section 3.1, on a well known dataset achieving a 33% relative
improvement over previous state-of-the-art methods. For more details on these results see Paper I.

• An end-to-end trainable multiple-agent reinforcement learning model that invents a grounded language to play Guess who?. See Section 2.4 for a short introduction or Paper II for a complete description.

• A purely learned approach to WSD, Section 3.2, that achieves results on par with state-of-the-art resource heavy systems, by leveraging GloVe vectors, Section 2.2.3, and a bidirectional long short-term memory network. See Paper III for more on these results and a detailed description of the model.

• A study on the applicability of neural word embeddings, Section 2.2, to provide a semantically aware sentence-to-sentence similarity score for use in extractive multi-document summarisation, Section 3.3. The results are disseminated in Paper IV and Paper V.
Chapter 2

Embedding Words in Geometric Spaces

Representing words as vectors has many advantages, three of them being: (1) they make it possible to encode different properties of words that may be shared between words, (2) they provide well defined distance measures, e.g. euclidean distance or cosine distance, for comparing words, and (3) they can encode correlated features by using a non orthogonal basis.

2.1 Basic Vector Representations

Before going into word embeddings some background on traditional vector space models are given in this section.

2.1.1 One-Hot Vectors

The most basic way of encoding a word in a vector space is called a one-hot encoding, i.e. a vector of the same dimensionality as the language where all but one dimension are zero and the remaining is one, the index of which encode the word type. This means that the vocabulary makes up an orthonormal basis for the vector space, which has the advantage that no assumptions about the words are being encoded in their representations. This orthonormal property makes them very useful in some applications, however, as semantic embeddings they are useless as they encode no information about the words and all words are of equal distance to each other.

2.1.2 Feature Vectors

To get a more semantically meaningful representation a second approach could be to list all known features of words and let them define the basis of the space. A word representation would then be a vector of zeros and ones indicating the absence or presence
of corresponding word feature. However, this leads to a few problems. First, it is not clear that all features are equally important which means that you will have to weight these to make the geometric distance measure meaningful and weighting features by hand with no clear objective would be highly subjective. Second, it is not possible to produce the definite list of all properties of words since language is continually changing.

2.1.3 Bag-of-Words Vectors

In order to improve scalability and objectivity we turn to hard statistics. But how do you capture word semantics in statistics? This difficult task was answers by Harris 1954 with the distributional hypothesis stating that, in the words of John Rupert Firth, You shall know a word by the company it keeps. I.e. statistics regarding which words co-occur can be used to form word representations. An early attempt at leveraging these statistics are called bag-of-words representations. These representations are related to the one-hot encodings in Section 2.1.1 as they can be formed by summing the one-hot vectors of all tokens in a corpus occurring within a given context window, e.g. three words before and after, the word type to represent. If this vector is subsequently normalized to sum to one, each dimension will indicate the probability of co-occurring with the word type corresponding to that dimension. These types of representations, when trained on a sufficiently large text corpus, will be able to enjoy all three of the advantages of vector based word representations stated in the beginning of the section. However, they suffer from one crucial deficit, the curse of dimensionality. This is because the dimensionality of the space equals the number of words in the vocabulary, which is very high for most languages, and has been shown to render geometric distance measures ineffective for measuring similarity between words in these models (Baroni, Dinu, and Kruszewski 2014). Though a lot of effort has been spent on overcoming this limitation, via different weighting and dimensionality reduction techniques, no definite answer of how to solve the problem has been found (Baroni, Dinu, and Kruszewski 2014).

2.2 Dense Real Valued Vectors Representations

In an effort to overcome the dimensionality problem of bag-of-words representations, Bengio et al. 2003 introduced a new way of leveraging co-occurrence statistics by learning to predict the context surrounding the target word using a neural network. By solving this proxy problem the network is forced into assigning similar vectors, now referred to as neural embeddings, for representing similar words. This approach has many advantages, one being that the embedding dimensionality can be chosen by the user. However, this model relied on computing a distribution over all words in the vocabulary, which is too computationally expensive to train the model on a large corpus.

2.2.1 CW Vectors

The first practical algorithm for training neural word embeddings was instead presented by Collobert and Weston 2008. This model solved the dimensionality problem by, instead
of learning the probability of each word type in the context of a target word, learning to differentiate between the correct target word and a random word given a context.

### 2.2.2 Continuous Skip-gram

However, it was with the *continuous Skip-gram* model by Mikolov, Chen, et al. 2013, released within the *Word2vec* package, that neural word embeddings became widely popular. The Skip-gram model is a simplified log-linear neural network, see Figure 2.2.1, that can be efficiently trained on huge amounts of data. Later the same year this model was shown by Mikolov, Yih, and Zweig 2013 to be able to capture multiple dimensions of similarity and be used to do analogy reasoning using linear vector arithmetics, e.g. \( v_{\text{king}} - v_{\text{man}} + v_{\text{woman}} \approx v_{\text{queen}} \).

### 2.2.3 Global Vectors for Word Representation

Though prediction based word embeddings quickly gained interest in the community and were fast replacing the counting based bag-of-words models, Pennington, Socher, and Manning 2014 showed that the two approaches had some complimentary properties and introduced *Global Vectors for Word Representation* (GloVe). GloVe is a hybrid approach to embedding words that combine a log-linear predictive model with counting based co-occurrence statistics to more efficiently capture global statistics, something they showed was lacking in the predictive models. As such, GloVe might represent the best of both worlds.

### 2.3 Sense Aware Word Embeddings

Though the word embeddings described in Section 2.2 has enjoyed much success they are actually founded on a false assumption, i.e. that each word has exactly one sense. This is clearly not true, e.g. the word *rock* may refer to either *music* or a *stone*. In this
Section we turn our attention to the problem of multiple senses of a single word type. To solve this we again employ the distributional hypothesis and use the embeddings of the surrounding words as a basis for a context specific word embedding tailored for a specific word token.

### 2.3.1 Instance-Context Embeddings

In Paper I, two approaches for computing sense aware word embeddings are introduced, where the first mainly provide a baseline for the second approach. The baseline system construct context dependent embeddings by averaging the word embeddings, described in Section 2.2.2, corresponding to the word tokens in their context. The drawback of the baseline approach, that we try to rectify in our second method *Instance-Context Embeddings (ICE)*, is that it attends the same amount on all words in the context even though some words are clearly more indicative for deciding the sense of a given target word. Our solution to this problem is to attend more to the words to which the Skip-gram model assigns a high probability of occurring in the target words context. This means that the words that correlate with the target word will be attended to more, but also that very common words that correlate with every word will be weighted less. This is due to the connection between the Skip-gram objective and *pointwise mutual information* showed in (Levy and Goldberg 2014), and has the effect of creating an embedding that is more stable for words sense, see Figure 2.3.1, and less affected by the noise of unrelated words, e.g. stop words or words that are rarely used together with the target word.

### 2.4 Embedding Grounded Concepts

Another aspect that is not covered by the word embeddings described in Section 2.2 is grounding. That is, the connection between the physical world and the words in a text. Grounding is not only important as a bridge to the physical world but could also aid in the understanding of the text by using generalization of concepts via physical properties.
rarely discussed in writing, e.g. that *most ground vehicles have wheels* which is apparent from images but usually not stated in written descriptions. In Paper II we take a first step towards grounded word embeddings by training agents to communicate concepts in images without any a priori shared language. i.e. they will need to create a language that encode concepts in the images in order to solve a common task. The task they are set out to solve is the game of Guess who?. A collaborative game, illustrated in Figure 2.4.1, where one player (the asking player) is tasked with figuring out which image, from a known set, that the other player (the answering player) is currently holding. To do this the asking player gets to ask questions to which the answering player will respond yes or no.

![Figure 2.4.1: Schematic illustration of our version of the Guess Who? game.](image)

To get a feeling for what concepts the agents decide to encode in their words we analyzed their interactions from a restrictive setup where the agents are only allowed two words (or questions) and the set of images only consist of two images sampled from the full set. The interactions from three such setups are tabulated in Table 2.4.1 where it can be seen that question B encoded a concept, perhaps the lack of mustache, that separated the images in the first and third setup. A deeper analysis of the result is given in Paper II.
Table 2.4.1: Final message protocols between the asking-agent and the answering-agent depending on the images the agents see.

<table>
<thead>
<tr>
<th>Asking-agent</th>
<th>Answering-agent</th>
<th>Question</th>
<th>Answer</th>
<th>Guess</th>
<th>Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>no</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>no</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>no</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>no</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>no</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>no</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Asking-agent</th>
<th>Answering-agent</th>
<th>Question</th>
<th>Answer</th>
<th>Guess</th>
<th>Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>no</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>no</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
Chapter 3

Applications of Word Embeddings to NLP

Though interesting in themselves, the main reason for the surge of interest in word embeddings is their applicability in natural language processing (NLP). Within this thesis three basic NLP application areas have been studied, and descriptions of each of them will follow.

3.1 Word Sense Induction

The first application considered is Word Sense Induction (WSI), the task of automatically creating a word sense inventory, i.e. lexicon, given a corpus. WSI is becoming an increasingly important tool for lexicographers trying to keep up with the ever increasing pace of language change. Our approach follow the work of Schütze 1998 by employing context clustering, i.e. embedding the context of tokens corresponding to a given word type and clustering them to find the different word senses. Traditionally the embeddings used have been different variations of the baseline system described in Section 2.3.1, i.e. bag-of-words representations. However, in paper Paper I we show that our proposed ICE embeddings, also described in Section 2.3.1, outperforms the traditional embeddings and achieved a relative improved over the previous state-of-the-art method of 33% on the WSI task of SemEval-2013.

3.2 Word Sense Disambiguation

The problem of assigning a word sense, from a set of predefined senses, to a word token is referred to as Word Sense Disambiguation (WSD). Traditionally WSD has been approached by modeling a fixed context window surrounding the target word, i.e. the word to disambiguate, as an unordered set. Though this may work for a large set of instances it is not difficult to find examples where the order is helpful, or even necessary,
In Paper III a sequence modeling approach is instead taken, where the order of words play an important part, and where the window is implicitly learned during training instead of defined a priori. See Figure 3.2.1 for an illustration of the model architecture.

![Figure 3.2.1: A BLSTM centered around a word at position n. Its output is fed to a neural network sense classifier consisting of one hidden layer with linear units and a softmax. The softmax selects the corresponding weight matrix and bias vector for the word at position n.](image)

The model stand in stark contrast to previous work in that it relies on no external features, e.g. part-of-speech taggers, parsers, knowledge graphs, etc., but still delivers results statistically equivalent to the best state-of-the-art systems. Further, we show that word embeddings play an essential role for the performance when trained on a limited amount of sense labeled data.

### 3.3 Automatic Multi-Document Summarisation

The amount of text being produced every day has exploded, which, if you want to follow what is being written on some topic is both a blessing and a curse. A blessing in that a much richer picture is being painted, less exposed to the subjective opinions of a few writers and able to cover more aspects in-depth. This sounds great, however, humans have a limited ability to read massive amounts of text, which means that you either have to limit yourself to the opinions of a handful producers or read a fair summary. However, manually producing such a summary is in most cases prohibitively expensive which is why automatic summarisation systems are becoming an increasingly important tool to keep up with the world.

#### 3.3.1 Extractive Summarisation

Automatic summarisation comes in two distinct flavors, abstractive and extractive. Abstractive summarisation is the more general solution where an abstract representation of the documents is created and the summary is generated based on this representation. In
contrast, extractive summarisation picks the most important sentences from the documents and put them together to form the summary, See Figure 3.3.1. Though abstractive summarisation more resemble how humans summarise text, extractive summarisation has so far been more successful at solving the task.

### 3.3.2 Comparing Sentences

Using the extractive summarisation framework presented by Lin and Bilmes 2011 provides a way of extracting sentences that are both descriptive of the document set, but also diverse within the set of extracted sentences to cover as much of the information contained in the documents as possible. However, in order to perform well, this system depends on having access to a high quality sentence-to-sentence similarity measure. In Paper IV we show that word embeddings can be used to compare sentences and provide a semantically meaningful sentence-to-sentence similarity score, but to do this we have to merge word embeddings into a sentence embedding. For this we evaluate two approaches: The first is to average the embeddings of all words in the sentence and use this as a representation. The second approach use a recursive auto encoder (RAE), proposed by Socher et al. 2011 and depicted in Figure 3.3.2, to recursively merge embeddings guided by a parse tree and finally using the root layer as a sentence representation.

### 3.3.3 MULTISUM

In Paper V we follow a similar strategy but the information from the word embeddings are combined with other measures and achieve a statistically significant improvement over the state-of-the-art on the well known dataset of Document Understanding Conference (DUC) 2004.
Figure 3.3.2: The structure of an unfolding RAE, on a three word phrase \([x_1, x_2, x_3]\). The weight matrix \(\theta_e\) is used to encode the compressed representations, while \(\theta_d\) is used to decode the representations and reconstruct the sentence.
Chapter 4

Future Direction of Research

As the licentiate thesis, to a large extent, represent a milestone on the way to a PhD, some thoughts on current and future work that will lead up to the dissertation are presented next. The general direction that is being taken is towards sequences of words and emergent properties captured through the interaction between agents. At the time of writing, this translates to the following list of ongoing projects:

Symbolic input sequence optimization  Taking an optimization approach to the sequence to sequence decoding problem by utilizing the gradient to do optimization over a one-hot input space.

Grounded word embeddings of human language  Connecting the grounded embeddings described in Section 2.4 with existing human language, to learn grounded embeddings of real words.

Waveform translation  Realizing that the models behind neural machine translation are independent of the underlying data, we try to connect the spectral voiceprint of the source sentence to the voiceprint of the target sentences directly. Though challenging, this approach has the potential of producing a far superior speech-to-speech translation system than approaches that are constraint by having to transcode the spoken language in text, since a lot of information gets lost in that step.
References


Part II
Publications
Neural context embeddings for automatic discovery of word senses

M. Kågebäck, F. Johansson, R. Johansson, and D. Dubhashi

Reprinted from Proceedings of NAACL-HLT, 2015
Neural context embeddings for automatic discovery of word senses

Mikael Kågebäck, Fredrik Johansson, Richard Johansson†, Devdatt Dubhashi
Computer Science & Engineering, Chalmers University of Technology
SE-412 96, Göteborg, Sweden
†Språkbanken, University of Gothenburg
Box 200, Göteborg, Sweden
{kageback,frejohk,dubhashi}@chalmers.se
richard.johansson@svenska.gu.se

Abstract

Word sense induction (WSI) is the problem of automatically building an inventory of senses for a set of target words using only a text corpus. We introduce a new method for embedding word instances and their context, for use in WSI. The method, Instance-context embedding (ICE), leverages neural word embeddings, and the correlation statistics they capture, to compute high quality embeddings of word contexts. In WSI, these context embeddings are clustered to find the word senses present in the text. ICE is based on a novel method for combining word embeddings using continuous Skip-gram, based on both semantic and a temporal aspects of context words. ICE is evaluated both in a new system, and in an extension to a previous system for WSI. In both cases, we surpass previous state-of-the-art, on the WSI task of SemEval-2013, which highlights the generality of ICE. Our proposed system achieves a 33% relative improvement.

1 Introduction

Ambiguity is pervasive in natural language and this is particularly true of word meaning: a word string may refer to several different concepts or senses. Word sense induction (WSI) is the problem of using a text corpus to automatically determine 1) the inventory of senses, and 2) which sense a particular occurrence of a word belongs to. This stands in contrast to the related task of word sense disambiguation (WSD), which is concerned with linking an occurrence of a word to an external sense inventory, e.g. WordNet. The result of a WSI system is a set of local sense labels, consistent within the system but not linked to a universal set of labels. A wide range of applications have been proposed where WSI could be useful, ranging from basic linguistic and lexicographical research (Nasiruddin et al., 2014), machine reading (Etzioni et al., 2006) and information retrieval (Véronis, 2004). WSI is of particular interest in situations where standard lexical resources are unreliable or inapplicable, such as when tracking changes of word meaning over time (Mitra et al., 2014).

According to the distributional hypothesis (Harris, 1954), word meaning is reflected in the set of contexts in which a word occurs. This intuition makes it natural to operationalize the meaning of a word – and of its contexts – using a vector-space representation, where geometric proximity corresponds to similarity of meaning. A common approach used in several successful WSI systems is to apply this geometric intuition and represent each context of a polysemous word as a vector, look for coherent clusters in the set of context vectors, and let these define the senses of the word. This approach was pioneered by Schütze (1998) using second order co-occurrences to construct the context representation. It is clear that in order to be useful in a WSI system, a geometric representation of context meaning must be designed in a way that makes clusters distinct.

Recently, neural embeddings, such as the popular Skip-gram model (Mikolov et al., 2013a), have proven efficient and accurate in the task of embedding words in vector spaces. As of yet, however, neural embeddings have not been considered for representing contexts in WSI. The systems that seem
most relevant in this context are those that train multi-prototype embeddings: more than one embedding per word (Huang et al., 2012). In particular, Neelakantan et al. (2014) described a modified Skip-gram algorithm that clusters instances on the fly, effectively training several vectors per word. However, whether this or any other similar approach is useful if considered as a WSI system is still an open question, since they have never been evaluated in that setting.

We make the following contributions: (1) We define the Instance-context embedding (ICE), a novel way for representing word instances and their context. ICE combines vectors representing context words using a novel weighting schema consisting of a semantic component, and a temporal component, see Section 3. (2) We propose two methods for using our embeddings in word sense induction, see Section 4. The first adopts a batch clustering scheme, where senses are induced after the word embeddings are computed. The number of senses is automatically chosen, based on data. The second extends an existing method for simultaneous embedding and clustering of words (Neelakantan et al., 2014). We show that our extension substantially improves the model. (3) We evaluate both proposed methods in the WSI task. We show that the two components of our proposed weighting schema both contribute to an increased overall performance. Further, we compare our method to state-of-the-art methods on Task 13 of SemEval-2013, achieving a 33% relative improvement see, Section 6.

2 Context clustering

Context clustering is an approach to WSI in which each instance of a word is represented by its context, embedded in a geometric space. These context embeddings are then clustered to form centroids representing the different senses of the target word. The context clustering approach was pioneered by Schütze (1998) who used second order co-occurrences to construct the context embedding. In this setting, the output of a WSI system is a set \( S_w = \{ s_{w,1}, \ldots, s_{w,k} \} \) of \( k \) locally defined senses of a word \( w \), with corresponding sense embeddings \( s_{w,j} \). We refer to \( S_w \) as the induced sense inventory of \( w \). The WSI problem is often paired with the related task of word sense disambiguation (WSD), concerned with linking a previously unseen occurrence of a word to an existing sense inventory. Given an instance \( w_i \), of a possibly polysemous word, let its context be represented by an embedding, \( c_i \). The sense of \( w_i \) is determined by finding the nearest neighbor to \( c_i \), in the sense inventory \( S_{w_i} \).

\[
\text{sense}(w_i) = \arg \min_{j : s_j \in S_{w_i}} d(c_i, s_j),
\]

where \( d(\cdot, \cdot) \) is some distance function. In this work, \( d \) is the cosine distance \( d(x, y) = 1 - \frac{x^T y}{\|x\| \|y\|} \). We proceed to review distributed word embeddings, used in this work to create context embeddings.

2.1 Distributed word embeddings

A word embedding is a continuous vector representation that captures semantic and syntactic information about a word. Such representations are often based on the distributional hypothesis of Harris (1954), stating that the meaning of a word is largely determined by the contexts in which it appears. For word embeddings, this is realized by assigning similar embeddings to words that appear in similar contexts. These representations can be used to unveil multiple dimensions of similarity between words, such as number, topic and gender (Mikolov et al., 2013b). Word embeddings computed using neural networks were introduced by Bengio et al. (2003) and are often called neural word embeddings.

Continuous Skip-gram is an algorithm for computing word embeddings that was introduced by Mikolov et al. (2013a). This model has received a lot of attention recently, being one of the models used in the software package word2vec (Mikolov, 2013). The model is trained to predict the context surrounding a given target word. Each word \( w \) is represented by two vectors, one for when the word is the target, denoted \( u_w \), and one for when it is in the context of another word, denoted \( v_w \).

We follow the interpretation of the negative sampling method for Skip-gram in Levy and Goldberg (2014). Let \( D \) denote the observed data, as a set of pairs of target and context words. Then, the probability of observing the pair \((w_i, w_j)\) of a context
word \( c \) and target word \( i \) in the data is,
\[
p((w_c, w_i) \in D) = \frac{1}{1 + e^{-\mathbf{v}_c^T \mathbf{u}_i}},
\]
(2)
where \( \mathbf{u}_i \) is the vector representation of the target word \( w_i \) and \( \mathbf{v}_c \) is the vector representation of the context word \( w_c \). The vectors \( \mathbf{u}_i \) and \( \mathbf{v}_c \) are referred to as word embeddings and context-word embeddings respectively. Training of the Skip-gram model with negative sampling corresponds to finding embeddings that maximize \( p((w_c, w_i) \in D) \) for observed context pairs and \( p((w_c, w_i) \notin D) \) for random (negative) context pairs. This is usually achieved using stochastic gradient descent.

### 2.2 Clustering word instances

Clustering of vector-observations is a well-studied subject. Perhaps the most widely used algorithm for this purpose, \( k \)-means clustering, embodies many of the intuitions and difficulties of the problem. In our setting, the vectors to cluster represent instances of a single word and \( k \) corresponds to the number of senses of the word. Clearly, \( k \) is highly dependent on the word, and is not easily set by hand. Although many algorithms have been proposed to solve the problem for a given \( k \), choosing \( k \) itself remains a problem in its own right. The frequently used Gap statistic (Tibshirani et al., 2000) gives a method for solving this problem. Unfortunately, it can be prohibitively slow for use in repeated clustering of large numbers of points, as the method relies on Monte Carlo simulations. Pham et al. (2005) proposed an alternative method in which a function defined by the cluster distortion for different values of \( k \), is used to evaluate cluster quality.

In the setting described above, the embeddings are assumed to be computed before clustering into senses. In contrast, Multi-sense Skip-gram (MSSG) (Neelakantan et al., 2014) attempts to learn several embeddings of a word, one for each of its different senses, by extending the Skip-gram method of Mikolov et al. (2013a). This involves a simultaneous embedding and clustering of word instances. A drawback is that their method limits the training of multi-sense embeddings to the \( M \) most common words, forcing a complete re-training of the model should a new word of interest appear.

### 3 Instance-context embeddings

We propose a new method for creating context embeddings for WSI. The embeddings are based on word embeddings and context-word embeddings computed using the Skip-gram model as described in Section 2.1. Our method differs from previous approaches in that it assigns different weights to the context words based on their influence on the meaning of the target word.

More precisely, the context embedding \( (c) \) for word instance \( i \) is computed as the weighted average of the context-word embeddings representing surrounding words
\[
c_i = \frac{1}{Z} \sum_{-T \leq c \leq T, c \neq 0} \psi_{i,c} \mathbf{v}_c.
\]
(3)
Here, \( \psi_{i,c} \) is the weight for context word \( c \), \( \mathbf{v}_c \) is the context-word embedding for the same word and \( T \) is the number of words, to the left and right, which are considered part of the context of target word \( i \). \( Z \) is a normalizing factor to put \( c_i \) on the unit sphere.

Perhaps the simplest weighting schema is the uniform, or non-informative schema, \( \psi^{\text{uniform}}_{i,c} = \frac{1}{2T} \forall i,c \). Context embeddings using uniform weights were used in the Multi-Sense Skip-Gram (MSSG) model by Neelakantan et al. (2014) for computing sense embeddings. However, in the context of WSI it is not hard to imagine a situation where an informed weighted sum would perform better. For example, in the phrase “the rock band” the word “band” is clearly more indicative for the sense of ”rock” than the word ”the”, and should therefore have a larger impact on the instance representation. To address this caveat we propose context embeddings based on a novel weighting schema, \textit{Instance-context embeddings} (ICE), that leverages co-occurrence statistics naturally captured by the Skip-gram model.

#### 3.1 Semantic context weights

The first component of ICE is based on the assumption that context words that strongly correlate with the target word is more important for the meaning of the target word. In the example phrase from Section 3, the word “band” is clearly a strong indicator for the presence of the word ”rock”, while the word
"the" occurs everywhere in English text and will therefore not have a strong correlation with "rock". To leverage this idea, we use the Skip-gram output probability, see (2), to weight context words by

$$\psi_{t,c}^\text{semantic} = \frac{1}{1 + e^{-v_c^Tu_t}},$$

where $v_c$ is the context-word embedding for the word $c$, and $u_t$ is the word embedding of target word $t$. Using $\psi^\text{semantic}$ in (3) has the effect of assigning bigger importance to context words that have a semantic relation to the target word. Context words that are not useful in characterizing the sense of the target are weighted less. This is in stark contrast to the uniform weighting schema.

Levy and Goldberg (2014) discovered an interesting connection between the Skip-gram model and Pointwise Mutual Information (PMI) (Church and Hanks, 1990). Consider the optimizers of the Skip-gram objective, word and context-word embeddings, $u_t, v_c$, trained using $k$ negative samples. Levy and Goldberg showed that for sufficiently large dimensionality, these vectors satisfy the following relation, $u_t^Tv_c = \text{PMI}(w_t, w_c) - \log k$. Let $\sigma(\cdot)$ be the logistic function. For vectors satisfying the conditions stated above, we have $\psi_{t,c}^\text{semantic} = \sigma(\text{PMI}(w_t, w_c) - \log k)$, establishing a connection between the semantic weights applied to Skip-gram embeddings, and PMI, a function frequently used for measuring word similarity (Pantel and Lin, 2002).

### 3.2 Temporal context weights

Window functions are used to extract local information from a sequence. In the context of NLP this translates to extracting a phrase of a given length from a larger text. The most common window function used in WSI is the rectangular window function, where $T$ words are extracted from each side of the target word. However, this approach is not optimal. In part, because it ignores the distance between the target word and the context word, but also because the sharp border makes the approach more noisy with respect to the chosen $T$.

To address these issues we instead apply a triangular window function to the context. This is inspired by the Skip-gram model, where this is achieved by uniformly sampling the context width $\in \{1 \ldots T\}$. In our model we weight the context words according to target word distance as

$$\psi_{t,c}^\text{temporal} = \frac{1}{T} \max(0, T - |t - c|).$$

### 3.3 Instance-context embeddings (ICE)

Finally, by combining the results of Section 3.1 and 3.2 we arrive at the definition of our proposed weighting schema

$$\psi_{t,c}^\text{ice} = \psi_{t,c}^\text{semantic} \psi_{t,c}^\text{temporal}.$$  

### 4 Word sense induction using ICE

We devise two methods for performing word sense induction using ICE. The first is based on the $k$-means clustering algorithm. Here, word and context-word embeddings are computed using Skipgram. Then, context embeddings are computed for all instances of a word, according to (3), and clustered using $k$-means, with Pham’s heuristic for choosing $k$ (Pham et al., 2005), to form centroids representing word senses. As clustering is performed in batch, after embedding, we refer to this method as ICE-kmeans.

The second method is an extension of the MSSG model (Neelakantan et al., 2014), in which we during training of the model embed word instances using ICE. This improves the disambiguation performed at every iteration of MSSG. As this method performs the clustering in an online fashion, we refer to this method as ICE-online. For this, we have modified the code provided by Jeevan Shankar¹.

### 5 Evaluation

We evaluate our methods for word sense induction on shared task 13 of SemEval-2013, *Word Sense Induction for Graded and Non-Graded Senses* (Jurgen and Klapaftis, 2013). Henceforth, we let “SemEval-2013” refer to this specific task. We also investigate the influence of our weighting schema on both methods. Further, we study qualitative properties of the word instance embeddings produced by our method.

¹https://bitbucket.org/jeevan_shankar/multi-sense-skipgram/
5.1 SemEval-2013, Task 13

The SemEval-2013 (test) data contains 4664 instances, each inflections of one of 50 lemmas (Jurgen and Klapaftis, 2013). The competition included both single-sense instances and instances with a graded mixture of senses. Because the manual annotations were deemed too poor, only 10% of instances were labeled with multiple senses (Jurgen and Klapaftis, 2013), which led the organizers to publish results both for all instances, and for single-sense instances only. For this reason, we consider only single-sense instances. Each instance is represented by a phrase, annotated with part-of-speech (POS) tags, comprising the word for which to determine the sense, and its context.

The rules of SemEval-2013 allowed the use of a specific corpus, ukWaC, for training of the submitted models. We have cleaned this corpus, removing formatting and making it lowercase. We extract common $n$-grams from the corpus and include them as entities in our vocabulary, e.g. Kuala Lumpur → Kuala Lumpur. Frequency thresholds were set to 10 times for $n = 1$, 20 times for $n = 2$, and 50 times for $n \in \{3, 4\}$. Longer phrases are not considered. Following SemEval-2013, we evaluate systems for unsupervised WSI using two different scores, Fuzzy B-Cubed (FBC) and Fuzzy Normalized Mutual Information (FNMI) (Jurgen and Klapaftis, 2013). FBC compares two fuzzy covers, clusterings of the data with partial memberships, on a per-item basis. The score is sensitive to cluster size skew. FNMI is a generalization of normalized mutual information for fuzzy covers. It measures the dependence between two clusterings independently of cluster sizes. As a final, combined score, we compute the harmonic mean (HM) of FBC and FNMI. To allow direct comparison with published results, we use the fuzzy measures even though we only consider single-sense instances.

We compare our results to two baselines from SemEval-2013. “One sense” predicts that all instances have the same sense. “One per instance” predicts that every instance has its own sense.

5.2 Experimental setup

For ICE-kmeans, we train a 300 dimensional Skip-gram model on the ukWaC corpus using standard parameter settings. I.e. context width set to 20 (10 before and 10 after), and 10 negative samples. We let the model iterate over the training data 9 times to improve the embeddings. For sense induction, we sample 1800 instances of very target word at random, from the ukWaC corpus. Using more instances did not improve the results in our experiments, however, for larger datasets this might not be valid. To remain general, we opted not to use the POS tags available in ukWaC, even though using them might have improved the result. Also, due to the noisy nature of the corpus, we exclude contexts where more than 30% of the words contain non-alphabetic characters. We cluster the selected instances using $k$-means clustering with the heuristic of Pham et al. (2005) for choosing $k$. For both ICE-kmeans and ICE-online, when computing the ICE vectors, the context width for was set to 20 when using the full schema, see (6), and 10 otherwise, as the full schema is less sensitive to irrelevant context words. For the MSSG part of ICE-online, we use the parameters reported in Neelakantan et al. (2014).

5.3 Current state-of-the-art

We compare the performance of our system to that of state-of-the-art systems for WSI.

First, we compare to the systems with the current best results on SemEval 2013 task 13 for single-sense word instances, AI-KU and unimelb. AI-KU (Baskaya et al., 2013) uses an approach based on substitute word vectors, inferred using a statistical language model. AI-KU achieved the highest FNMI score of the systems submitted to SemEval-2013. unimelb (Lau et al., 2013), who achieved the highest FBC score at SemEval-2013, is a system based on the topic model Latent Dirichlet Allocation and its non-parametric equivalent, Hierarchical Dirichlet Processes. Word instances are clustered based on the topic distributions inferred by the model.

The related problem of training neural embeddings of polysemous words was addressed by Huang et al. (2012) and subsequently by Neelakantan et al. (2014) with the model Multi-sense Skip-gram (MSSG), see Section 2.2. As a second experiment we extend MSSG for WSI. MSSG has not previously been used for WSI, however it produces one word embedding for each word sense, and performs a simple disambiguation procedure during training.
MSSG is thus a natural candidate for comparison. We use the standard variant of MSSG, as it achieved the best overall results in the original paper Neelakantan et al. (2014). MSSG disambiguates instances by assigning them to the sense with embedding closest to the average context-word vector of the instance, i.e. using uniform weighting. We use the parameters reported in Neelakantan et al. (2014), with the number of senses per word set to 3. MSSG takes a parameter $M$ specifying the number of words for which multiple sense vectors are computed. Like in Neelakantan et al. (2014), we set this parameter to $M = 30000$. We note that only 43 out of 50 lemmas in SemEval-2013 were in the $M$ most common words assigned multiple vectors by the MSSG methods. For the remaining 7 words, a single sense was predicted. Making sure all relevant words are included is not trivial in practice, without knowledge of the test set, as the training time of the model depends greatly upon $M$.

6 Results

We report the results of all experiments below.

6.1 Qualitative evaluation of instance vectors

Consider the word “paper”. WordNet (Miller, 1995) lists seven senses of “paper” as a noun: 1) a medium for written communication, 2) an essay (especially one written as an assignment), 3) a scholarly article describing the results of observations or stating hypotheses, 4) a daily or weekly publication on folded sheets; contains news and articles and advertisements, 5) a business firm that publishes newspapers, 6) a material made of cellulose pulp derived mainly from wood or rags or certain grasses, 7) the physical object that is the product of a newspaper publisher. Assigning an instance to one of these senses can be challenging even for a human reader.

The word “paper” is one of the 50 lemmas in the SemEval-2013 evaluation data with corresponding instances that cover six of the senses listed in WordNet. In Figure 1, we show context embeddings for these instances, plotted using the dimensionality reduction tool t-SNE (Van der Maaten and Hinton, 2008). Figure 1a represents context embeddings computed using a uniform average, and Figure 1b plots the instance context embeddings computed with using ICE, as described in Section 3. The colors and markers correspond to gold-standard WordNet annotations provided by SemEval. The size of a marker in Figure 1b is proportional to the average ICE weight of words in the context of an instance and is indicative of the confidence in the instance vector. A low average ICE weight indicates that the context is not predictive of the target word.

For the senses, “material”, “scholarly article”, “newspaper” and “essay”, the instances in Figure 1b are noticeably more clustered than in Figure 1a. This shows that the senses of these words are better represented using ICE weighting for context embeddings than a uniform schema.

6.2 SemEval WSI results

The results of the WSI evaluation on shared task 13 of SemEval-2013 are presented in Table 1. Here, our system ICE-kmeans, and our MSSG extension ICE-online, use the ICE weighting schema, see (6). MSSG is the system presented in Neelakantan et al. (2014) without modifications. AI-KU and unimelb represent the best systems submitted to SemEval-2013, and AI-KU the current state-of-the-art in WSI.

First, we note that ICE-kmeans achieves the overall best results with respect to both scores, corresponding to a relative improvement of 31.1% in FNMI and 15.9% in FBC. Further we note that the previous best FBC and FNMI belong to different methods. This is important since, as with precision and recall, achieving a high score in one of these measures can be achieved using a trivial baseline, see the first two methods in Table 1. Hence, a better benchmark, analogue to the $F_1$-score, is the harmonic mean (HM) of the two complementary scores. Considering this our results are even more impressive with a 33% relative improvement.

6.3 Semantic and temporal component of ICE

We evaluate the impact of using context embeddings based on the different weighting schemas defined in Section 3, over embeddings based on uniform weights. The results are presented, as harmonic mean and relative improvement over previous state-of-the-art AI-KU, in Table 2.

First, we note that both variants of our full system (ICE) offers a substantial relative improvement over AI-KU. We note that the results are always
Figure 1: Context embeddings for instances of the noun “paper” in the SemEval-2013 test data, plotted using t-SNE. The legend refers to WordNet gold standard embeddings.

<table>
<thead>
<tr>
<th>Method</th>
<th>FBC(%)</th>
<th>FNMI(%)</th>
<th>HM</th>
</tr>
</thead>
<tbody>
<tr>
<td>One sense</td>
<td>57.0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>One per instance</td>
<td>0</td>
<td>4.8</td>
<td>0</td>
</tr>
<tr>
<td>Unimelb</td>
<td>44.1</td>
<td>3.9</td>
<td>7.2</td>
</tr>
<tr>
<td>AI-KU</td>
<td>35.1</td>
<td>4.5</td>
<td>8.0</td>
</tr>
<tr>
<td>MSSG</td>
<td>45.9</td>
<td>3.7</td>
<td>6.8</td>
</tr>
<tr>
<td>ICE-online</td>
<td>48.7</td>
<td>5.5</td>
<td>9.9</td>
</tr>
<tr>
<td>ICE-kmeans</td>
<td>51.1</td>
<td>5.9</td>
<td>10.6</td>
</tr>
</tbody>
</table>

Table 1: Results for single-sense instances on the WSI task of SemEval-2013. HM is the harmonic mean of FBC and FNMI.

7 Conclusion

We have presented Instance-context embedding (ICE), a method for embedding word instances and their context for use in word sense induction (WSI). At the heart of the system are instance representations based on neural embeddings of context words, combined using a novel weighting schema.

We have shown that ICE is successful in representing instances of polysemous words, not just in our own WSI system, but in an extension of an existing model as well. In an evaluation on the WSI task of SemEval-2013, our system beat previous state-of-the-art methods, achieving a 33% relative improvement. Further, we have established the benefits of using ICE over a uniform weighting schema, by showing empirically that each of its components contribute to a more accurate system.

Acknowledgments

The authors would like to acknowledge the project Towards a knowledge-based culturomics supported by a framework grant from the Swedish Research Council (2012–2016; dnr 2012-5738), and the project Data-driven secure business intelligence grant IIS11-0089 from the Swedish Foundation for Strategic Research (SSF). The third author was supported by Swedish Research Council grant 2013–4944, Distributional methods to represent the meaning of frames and constructions.

References

<table>
<thead>
<tr>
<th>Method</th>
<th>Uniform HM</th>
<th>Imprv. (%)</th>
<th>ICE (Sem. only) HM</th>
<th>Imprv. (%)</th>
<th>ICE HM</th>
<th>Imprv. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICE-online</td>
<td>6.8†</td>
<td>-14†</td>
<td>7.9</td>
<td>-1.1</td>
<td>9.9</td>
<td>24</td>
</tr>
<tr>
<td>ICE-kmeans</td>
<td>7.6</td>
<td>-5.2</td>
<td>9.6</td>
<td>20</td>
<td>10.6</td>
<td>33</td>
</tr>
</tbody>
</table>

Table 2: Impact of the two different components of ICE on SemEval-2013 task 13 (single-sense). Imprv. is the relative improvement over the state-of-the-art system AI-KU (with an HM of 8.0). †Gray numbers correspond to the original MSSG system. The weighting schemas are: Uniform, Section 3, ICE (Semantic only), Eq (4), and ICE, Eq (6).
Learning to Play Guess Who? and Inventing a Grounded Language as a Consequence

E. Jorge, M. Kågebäck, and E. Gustavsson

Reprinted from NIPS workshop on deep reinforcement learning, 2016
Abstract

Learning your first language is an incredible feat and not easily duplicated. Doing this using nothing but a few pictureless books, a corpus, would likely be impossible even for humans. As an alternative we propose to use situated interactions between agents as a driving force for communication, and the framework of Deep Recurrent Q-Networks (DRQN) for learning a common language grounded in the provided environment. We task the agents with interactive image search in the form of the game Guess Who?. The images from the game provide a non trivial environment for the agents to discuss and a natural grounding for the concepts they decide to encode in their communication. Our experiments show that it is possible to learn this task using DRQN and even more importantly that the words the agents use correspond to physical attributes present in the images that make up the agents environment.

1 Introduction

Human language closely interacts with the world it is used to describe. Language arose as a way of transmitting knowledge about the state of the world between the people that live in it, and it evolves as the world changes over time. Severing this link, by analysing text as a stand alone artifact, leads to problems with grounding of concepts and effectively eliminates exploratory mapping of the language. In contrast, when two humans communicate they generally do so in connection to the environment and in both directions, which provides the necessary grounding of concepts but also immediate feedback on every utterance. The importance of feedback for human language learners was shown by Sachs et al. in [1]. This paper describes the case of Jim, a hearing child that was brought up by two deaf parents and had to learn to speak from watching television without any supervision or feedback. These circumstances severely delayed his acquisition of language and he did not learn to speak properly until after intervention from the outside, which highlights the importance of guided exploration of the language via synthesis and feedback for mastering a language.

In this paper we investigate if a grounded language can emerge by letting two agents invent their own language to solve a common problem using DRQN. A first step in this direction was taken by Foerster et al. in [2, 3] where agents learn to communicate using binary messages to solve riddles. However, our goal is to enable the agents to evolve a richer language and to facilitate that we propose to use images as conversation pieces. More precisely we let two agents play the game of Guess Who? which forces the agents to come up with grounded words that represent characteristics of objects in images in order to win the game. The model we propose is similar to the Differentiable inter-agent learning (DIAL) model presented in [2], see Section 2.3 for more details, but differs in some key areas. (1) Instead of communicating using bits we generalise the model to handle orthogonal messages of arbitrary dimension to enable vocabularies of arbitrary size, (2) by gradually increasing the noise.
on the communication channel we ensure that the agents learn a symbolic language but with less negative impact on the convergence rate during training and with a positive impact on the learning capacity (for more on the model see Section 4) and (3) in our model no parameters are shared between the agents since this is more reasonable from a human perspective.

The main contributions of this paper include:

- An end-to-end trainable multiple-agent reinforcement learning model that learns a near optimal strategy for playing Guess who? without sharing any parameters between agents and with no predefined communications protocol.
- An analysis of the invented language that shows how the words are grounded in the concepts visible in the images.
- Experiments that show how increasing levels of noise in the communication channel leads to an improved training speed and learning capacity (compared to constant noise) while retaining the ability to learn discrete symbols.
- Finally, we generalise DIAL to use orthogonal messages of arbitrary dimension, to more closely resemble human language which encompasses hundreds of thousands of words, and we show that this improve the performance of the system as well as make it more interpretable.

2 Background

The results in this paper rely mainly on the theory of reinforcement learning, deep Q-networks, and the concept of differentiable inter-agent learning. In this section short descriptions of these methodologies are presented.

2.1 Reinforcement learning

In a traditional single-agent reinforcement learning (RL) framework an agent observes the current state \( s_t \in S \) at each time step \( t \), takes action \( u_t \in U \) according to some policy \( \pi \), receives the reward \( r_t \), and transitions according to some probability distribution (depending on the current state and action) to a new state \( s_{t+1} \in S \). The objective of the agent is to maximize the expected discounted future sum of rewards \( R_t = \sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_{\tau} \), where \( \gamma \in [0, 1] \) is a discount factor that trades-off the importance of immediate and future rewards. For a specific policy \( \pi \), the value of a state-action pair is defined as \( Q^\pi(s, u) = \mathbb{E}[R_t | s_t = s, u_t = u] \). The optimal value function \( Q^*(s, u) = \max_{\pi} Q^\pi(s, u) \) is called the Q-value of the state-action pair \( (s, t) \) and obeys the Bellman optimality equation \( Q^*(s, t) = \mathbb{E}[r + \gamma \max_{u'} Q^*(s', u') | s, u] \). Whenever an agent employs the optimal strategy the agent is guaranteed to achieve the highest expected sum of discounted rewards.

To find the Q-values various RL algorithms have previously been used where the most successful ones are the iterative methods Q-Learning and Profit Sharing. RL can be extended to cooperative multi-agent settings where each agent \( a \) observes a global state \( s_t \), selects individual actions \( u_t^a \), and then receives a team reward \( r_t \), shared among all agents. When all agents do not have full observability, i.e., when the agents do not observe the entire environment, the global state \( s_t \) is hidden and the agents only receive observations \( o_t \) that are correlated with the state \( s_t \).

2.2 Deep Q-Networks

The space of state-action pairs is, in many applications, so large that storing and updating the Q values for each state-action pair is computationally intractable. One solution for this dimensionality problem is to employ the concept of Deep-Q-Networks (DQN) (for a more thorough description, see [4]). The idea of DQN is to represent the Q-function by using a neural network parameterised by \( \theta \), i.e., to find \( Q(s, u; \theta) \) which approximates the value \( Q^*(s, u) \) for all state-action pairs. The network is optimised by minimising the loss function \( L_i(\theta_i) = \mathbb{E}[(y_{i}^{DQN} - Q(s, u; \theta_i))^2] \), at iteration \( i \), with \( y_{i}^{DQN} = r + \gamma \max_{u'} Q(s', u'; \theta^-) \), where \( \theta^- \) are the parameters of a target network which is fixed for a number of iterations. The actions chosen during the training of the network are determined by an \( \epsilon \)-greedy policy that selects the action that maximises the Q-value for the current state with probability \( 1 - \epsilon \) and chooses an action randomly with probability \( \epsilon \). When agents only have partial observability, Hausknecht and Stone ([5]) propose to use an approach called Deep Recurrent Q-Networks (DRQN)
where, instead of approximating the Q-values with a feed-forward network, they approximate the Q-values with a recurrent neural network that can maintain an internal state which aggregates the observations over time. This is modelled by adding an input $h_{t-1}$ to the network that represents the hidden state of the network.

### 2.3 Differentiable inter-agent learning

In earlier work where communication is a part of the RL setting, messages are seen as actions and are selected via Q-functions and action policies such as $\epsilon$-greedy. In [2], Foerster et al. introduce the idea of centralised training but decentralised execution, i.e., the agents are trained together but evaluated separately. They introduce the concept of Differentiable Inter-Agent Learning (DIAL) where messages are allowed to be continuous during training, but need to be discrete during evaluation, which allows gradients to propagate between the agents through the messages in training. This gives the agents more feedback and thus reduces the amount of learning required through trial-and-error. To reduce the discretisation error that could occur from the discretisation of messages in the evaluation phase the messages are processed by a dicreitise/regularise unit (DRU). During centralised learning the DRU regularises the messages according to $\text{DRU}(m_t^a) = \text{Logistic}(N(m_t^a; \sigma))$, where $m_t^a$ is the message of agent $a$ in time step $t$, and during decentralised execution $\text{DRU}(m_t^a) = 1(\|m_t^a\| > 0)$. The added noise $\sigma$ during the training phase pushes the messages towards the ends of the logistic function and therefore will force the agents to send almost discrete messages. The change of setup also requires a slight change in how the model is trained; the loss with respect to the $Q$-function is the same as in the DQN case but the gradient term for $m_t^a$ is the error backpropagated through the message from the recipient to the sender. More algorithm details can be found in [2, Appendix A]. Using DIAL instead of traditional independent $Q$-learning techniques is shown to be beneficial, both in terms of learning speed and in terms of ability to learn.

### 3 Guess Who?

The task we consider in this paper is a version of the popular guessing game *Guess Who?*. In *Guess Who?* two players are each assigned one character from a set of 24 characters. Each character has an image and the goal of the game is to figure out which of the 24 characters the other player has. The players take turns asking questions based on the visual appearance of the characters. The questions have to be answered truthfully with *yes* or *no* until one player knows which character the other player was assigned.

The version of Guess Who? that we consider is slightly different. One important difference is that we remove the competing factor and have one agent be the *asking-agent* ($\alpha = 1$) and the other the *answering-agent* ($\alpha = 2$). This means that one agent specialises in asking questions and the other in answering them, and the two of them collaborate with the objective of solving the task. Another difference is that instead of finding the correct character amongst the full set of characters a subset is sampled and given as observation input to the asking agent; in our experiments we used a subset of two or four images. The answering agent only observes the image of its assigned character from the subset observed by the asking-agent.

In Figure 1 an example of our version of Guess Who? is illustrated. The asking-agent has four images and the answering-agent has the (correct) image which is the third image of the asking-agent. The asking-agent sends its first question ($m_1^1$) and receives the answer ($m_1^2$). Then another round of question and answer occurs and finally the asking-agent guesses which of the images the answering-agent holds.

In each time step, the agents take turns sending and receiving messages. This means that in each step either the answering-agent’s receives a question or the asking-agent gets an answer. The questions that the asking-agent may send are limited to a discrete set of questions and the answers the answering-agent can send are limited to two answers (*yes* or *no*). Instead of playing the game with a variable amount of time steps a fix amount of steps is played. For the case when the asking-agent observes two images the agent is only allowed to ask one question ($m_1^1$) and receive one answer ($m_2^1$), and for the case when the asking-agent observes four images, two questions ($m_1^1, m_1^2$) and two answers ($m_2^1, m_2^2$) are allowed. After all answers have been received the asking-agent has to guess which of the images the answering-agent has. The guess is considered as an action taken by the asking-agent and is represented by $u^\downarrow \in \{1, \ldots, n\}$, where $n$ is the number of images the agent holds. If $u^\downarrow$ is equal to
the index of the character that the asking-agent has, a reward of 1 is given to both agents, otherwise they receive 0.

When the number of question-answer rounds is limited it is not always possible to win each game since only a fixed number of messages are available and only yes or no answers are permitted. If two messages are available it is essentially possible to partition the set of images into four parts. With four messages this partition can be made into 16 parts. If there are fewer parts than the size of the image set it means that several images will be partitioned into the same part, yielding in the same answers to the questions posed and therefore making them indistinguishable for the model. This implies that the maximum average reward for a game where the asking-agent holds two images from a total of 24 classes (as in Guess Who?) and two different messages are allowed is only 0.89 (for derivations see Appendix A). The equivalent score in a four image game with two rounds of questions would be 0.71 (for derivations see Appendix A). However, since our agents have recurrent networks it is possible that the questions in each round have different meanings (otherwise there would be no point in asking the same question in the second round as in the first round) such that the set can be partitioned into more parts, implying that the maximum average reward is higher than 0.71.

4 Model

The architecture of the model is presented in this section and a schematic illustration of the model is shown in Figure 2. Each agent \((a = 1, 2)\) consists of a recurrent neural network (RNN) that is unrolled for \(T\) time steps with an internal state \(h\), an input network, and an output network. The input network takes the information available, \((o^a_t, \hat{m}^{a'}_{t-1}, u^a_{t-1})\) and transforms it into a 256 embedding \(z^a_t\). The observation \(o^a_t\) of agent \(a\) at time step \(t\) is passed through a 2-layer multilayer perceptron (MLP) of size \([\#\text{colours} \times |\text{image}|, 128, 256]\). The regularised message \(\hat{m}^{a'}_{t-1}\) from the other agent, \(a'\), is passed through an MLP of size \([|M|, 256]\). The previous action \(u^a_{t-1}\) for agent \(a\) is passed through a lookup-embedding of size 256. These embeddings are added element-wise to give a final embedding \(z^a_t\). The embedding \(z^a_t\) is then processed by a 2-layer RNN with gated recurrent units (GRU) [6] of size \([256, 256]\) to give the output embedding \(h^a_{2,t}\). The output embedding \(h^a_{2,t}\) is then processed by an MLP to give \(Q_{a,\alpha,\beta} = MLP[256, 256, U + M](h^a_{2,t})\). The \(Q\)-function \(Q_{a,\alpha}(u)\) is used to generate the action \(u^a_t\) according to an \(\epsilon\)-greedy policy. In our model we utilise messages in a one-hot encoding which are then passed to the other agent for the next time step. While using a one-hot encoding does have disadvantages in terms of scalability (since a binary encoding of length \(n\) gives \(2^n\) different possible messages instead of only \(n\) in the one-hot encoding) we found that it gave improved results and makes it easier to study the underlying effect behind each message which is clouded by the varying closeness between messages that occurs in binary encoding. It also makes
Figure 2: Architecture of the model. The time dimension (question-answer rounds in the game) of the RNN goes from top to bottom. The green boxes illustrate the internal state of the network.

sense from a human perspective since we communicate through words which are non-binary. The message $m^t_a$ is passed through a variant of DRU as described in section 2.3 to generate a one-hot encoding using $\hat{m}^t_a = \text{DRU}(m^t_a) = \text{Softmax}(N(m^t_a, \sigma_{\text{episode}}))$ in the training case where $N(\mu, \sigma)$ is the $|m|$ dimensional normal distribution. During evaluation $\hat{m}^t_a(i) = \{1$ if $i = \text{argmax}(m), 0$ otherwise $\}$ is used instead.

Batch normalization (see, [7]) is performed in the MLP for the image embedding and on the incoming messages $m^t_{a-1}$. During testing non-stochastic versions of batch normalization is used which implies that running averages of values observed during training are used instead of those from the batch.

Updates of the parameters of the network are performed as described in section 2.3 with more details available in [2, Appendix A].

5 Increasing noise

Inspired by curriculum learning described in [8], where Bengio et al. show that gradually increasing difficulty of tasks leads to improved learning, we introduce the usage of increasing noise to DIAL. We found that using a fix value for the noise $\sigma$ led to an unwelcome trade off. If $\sigma$ is too large there is a risk of the model never learning anything. If $\sigma$ is too small you get an excellent training error (where continuous messages are allowed) but bad testing error due to the model over-encoding information in the messages leading to a large discretisation error. Our solution to this is to allow the noise to linearly increase over the epochs. This enables the model to learn quickly in the beginning but to punish the over-encoding of information more and more as the training progresses. This can be compared to human speech where a person who is beginning to learn a new language requires an almost noise free surrounding to understand a message while a proficient speaker would have no problem understanding a message in a more noisy environment such as listening to the radio or having a conversation by a busy road.

6 Experiments

To validate our ideas we run a series of experiments on our version of Guess Who? to see how well it performs but also what the model learns. Further, we evaluate the effectiveness of increasing noise by comparing it to experiments with non-increasing noise.

For each episode we randomly select two or four different images from the pool of 24 Guess Who? images (which are down-sampled to $32 \times 32$ pixels). The model is trained with $\sigma$ increasing linearly from 0.1 to 1. We evaluate the model performance with two, four, or eight different messages available to the asking agent in a one-hot embedding, and the answers yes/no available to the answering agent.
The performance of the model when the asking-agent has two images, one round of question-answer is performed, and two, four, or eight different messages are allowed. The results are averaged over five runs.

The performance of the model when the asking-agent has four images, two rounds of question-answer are performed, and two, four, or eight different messages are allowed. The results are medians of twelve runs.

Figure 3: Performance of the model when allowing different number of messages. The dashed grey lines represents the baseline performance where the asking-agent guesses randomly.

The game goes on for three time steps (i.e., one question, one answer, and one guess) for two images and five time steps (i.e., two questions and two answers) for four images.

The experiments use a $\epsilon$-greedy policy with $\epsilon = 0.05$, a discount factor $\gamma = 1$, and each epoch performs parallel episodes using a batch size of 32. The target network $\theta^-$ is updated every 100 epochs. Unless stated otherwise, we let $\sigma$ increase linearly from 0.1 to 1; this keeps the average $\sigma$ close to what was found to work on a task with similar tasks in [2], where they used $\sigma = 0.5$. We use a small dropout parameter of 0.1 to be able to maintain network size relatively small. The optimization is done by rms-prop with a learning rate of $5 \times 10^{-4}$. Each experiment is run multiple times and the results averaged. We intend to publish the code for all experiments online.

7 Results

7.1 Performance

In Figure 3a the average performance (over five runs) is illustrated in the case where the asking-agent sees two images, here it is clear that more messages are crucial for better performance on two images. The score for two messages is close to the theoretical bound of 0.89 for two messages (as discussed in Section 3) but is surpassed when using four and eight messages.

The performance for four images is show using median values. This is due to the fact that one of the runs with four messages fails to learn anything and remains at baseline performance, additionally there are three runs (two with four messages and one with eight messages) that learn a semi-successful protocol before returning to the baseline. The performance over twelve runs can be seen in Figure 3b. It can be seen that the performance obtained on four images is quite similar independently of how many questions are available, specially when taking into account that there is some instability in the training. In the case of four images with two messages some of the evaluations achieve a score that is above 0.71 which would be the best possible if the questions in both asking phases are interpreted the same (as discussed in Section 3). This shows that the agents learn to change the interpretation of questions depending on if it is the first or the second question or depending on what question was asked previously.

7.2 Understanding the questions

To better understand the nature of the questions that the asking-agent asks we sample a few episodes from a model with two images and two available questions to see how the games play out, i.e., what
Table 1: Final message protocols between the asking-agent and the answering-agent depending on
the images the agents see.

<table>
<thead>
<tr>
<th>Asking-agent</th>
<th>Answering-agent</th>
<th>Question</th>
<th>Answer</th>
<th>Guess</th>
<th>Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Image A]</td>
<td>![Image B]</td>
<td>B</td>
<td>yes</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>![Image A]</td>
<td>![Image B]</td>
<td>B</td>
<td>no</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>![Image A]</td>
<td>![Image B]</td>
<td>B</td>
<td>yes</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>![Image A]</td>
<td>![Image B]</td>
<td>B</td>
<td>no</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>![Image A]</td>
<td>![Image B]</td>
<td>A</td>
<td>no</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>![Image A]</td>
<td>![Image B]</td>
<td>A</td>
<td>no</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>![Image A]</td>
<td>![Image B]</td>
<td>A</td>
<td>no</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>![Image A]</td>
<td>![Image B]</td>
<td>A</td>
<td>no</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>![Image A]</td>
<td>![Image B]</td>
<td>B</td>
<td>yes</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>![Image A]</td>
<td>![Image B]</td>
<td>B</td>
<td>no</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>![Image A]</td>
<td>![Image B]</td>
<td>B</td>
<td>yes</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>![Image A]</td>
<td>![Image B]</td>
<td>B</td>
<td>no</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

The message protocol between the agents is depending on the images they see. The message protocol is illustrated in Table 1. In the table it can be seen that the asking-agents questions depend on what images it has, the answers it gets from the answering-agent, and its interpretation of the answer (its guess). One can see that in some cases it receives the same answer even when the answering-agent has different images, this leads to the answering-agent sometimes making a guess on the wrong image and receiving zero reward.

The answers to each question for the different images is illustrated in Figure 4. This shows that the two images that it has trouble separating (see the ones yielding zero reward in Table 1) have the same answers to both questions (No to question A and Yes to question B), as such it is impossible for the asking-agent to distinguish between the two. This illustrates the problem with a limited set of messages which leads to only being able to partition the set in a few ways. Judging from the appearances of the images, it appears that the question A can be interpreted as something along the lines of Is the top of his/her head very light coloured or very darkly coloured?. Question B is harder to interpret but it could be something along the lines of Are his/her cheeks and ears free from a surrounding object (such as hair or sideburns), and if this is not the case is the surrounding object very lightly coloured?.

7.3 Increasing noise

To evaluate the effectiveness of increasing the noise σ during training we compare results on Guess Who? using four images and with eight possible messages and an increasing σ with a constant σ = 0.5 (which was used in [2]) but also with σ = {0, 0.1, 1}. In Figure 5 it is clearly visible that there is a significant advantage in using variable noise compared to constant noise. The model learns both faster and achieves better performance compared to the models trained with constant noise.

8 Related works

Previous work has been on communication in a multi-agent reinforcement learning [2, 3, 9]. In [9] the predator-prey environment is studied where agents learn a communication protocol consisting of bits. There it is shown that while longer messages (i.e. larger bandwidth) may take longer to learn they generally give improved results. In [2, 3] the authors solve puzzles with multiple agents with one-bit communication.
Figure 4: A partition of the Guess Who? images defined by the answers the answering-agent gives for each question (for the case when two questions are allowed).

Figure 5: The performance of the model for different noise levels ($\sigma = \{0, 0.1, 0.5, 1.0\}$), and $\sigma$ varying linearly from 0.1 to 1.0.) The dashed grey line represents the baseline performance where the asking-agent guesses randomly. The results are averaged over twelve runs for the increasing $\sigma$ and five runs for constant $\sigma$.

9 Conclusions

In this paper we have shown that a DRQN can learn to play Guess Who?. Further, by careful analysis of the results we could establish a grounding connection between the words used by the agents and characteristics of objects in images from the game. Finally, we showed that regulating the level of noise in the communications channel could have a large impact on the training speed as well as the final performance of the system.

Acknowledgments

The authors would like to acknowledge the project Towards a knowledge-based culturomics supported by a framework grant from the Swedish Research Council (2012–2016; dnr 2012-5738). The authors would also like to thank Jakob Foerster and Yannis M. Assael for the insightful discussions.
References


A Derivations for optimal scoring with two messages

When trying to correctly differentiate between two images from a pool of 24 images using one out of two available question the probability of guessing the correct image can be seen as the following: Let image A be the correct image and B be the incorrect image from the pool.

\[
P(\text{Correct}) = \frac{P(\text{Correct} | \text{Separable}) \cdot P(\text{Separable})}{P(\text{Separable})} = P(\text{Correct} | \text{Separable}) + P(\text{Correct} | \text{Not separable}) \cdot P(\text{Not separable})
\]

(1)

Images are separable if the image A and B are not in the same subset of images divided up by the questions. Since a subset that divides the set up equally is optimal this means that the 24 images are divided up such that there is 6 images in each subset and \( P(\text{Separable}) = 1 - \frac{5}{23} = \frac{18}{23} \). If they are inseparable asking agent will have to perform a random guess. This gives equation (1) \( = 1 \cdot \frac{18}{23} + \frac{1}{2} \cdot \frac{5}{23} \approx 0.89 \).

For the case with four images it is slightly different. The following calculations assume that two questions are asked but that those are the only two questions available such that the two questions available in the first question phase are the same as in the second phase. This means that the optimal procedure will be to ask one of them in the first phase and the other in the second phase. In this case the calculations are as above but taking into consideration that there may be multiple images in the correct subset. We then get

\[
P(\text{Correct}) = \frac{P(\text{Correct} | \text{Separable}) \cdot P(\text{Separable})}{P(\text{Separable})} = P(\text{Correct} | \text{Separable}) + P(\text{Correct} | \text{Not separable with one}) \cdot P(\text{Not separable with one}) + P(\text{Correct} | \text{Not separable with two}) \cdot P(\text{Not separable with two}) + P(\text{Correct} | \text{Not separable with three}) \cdot P(\text{Not separable with three})
\]

\[
= \frac{\binom{4}{2} \binom{18}{3} + \frac{1}{2} \binom{4}{1} \binom{18}{2} + \frac{1}{4} \binom{4}{1} \binom{18}{1} + \frac{1}{16} \binom{4}{1} \binom{18}{0}}{\binom{23}{3}} \approx 0.71
\]
Word Sense Disambiguation using a Bidirectional LSTM

M. Kågebäck and H. Salomonsson

Reprinted from 5th Workshop on Cognitive Aspects of the Lexicon (CogALEx), 2016
Word Sense Disambiguation using a Bidirectional LSTM

Mikael Kågebäck, Hans Salomonsson∗
Computer Science & Engineering, Chalmers University of Technology
SE-412 96, Göteborg, Sweden
{kageback,hans.salomonsson}@chalmers.se

Abstract

In this paper we present a clean, yet effective, model for word sense disambiguation. Our approach leverage a bidirectional long short-term memory network which is shared between all words. This enables the model to share statistical strength and to scale well with vocabulary size. The model is trained end-to-end, directly from the raw text to sense labels, and makes effective use of word order. We evaluate our approach on two standard datasets, using identical hyperparameter settings, which are in turn tuned on a third set of held out data. We employ no external resources (e.g. knowledge graphs, part-of-speech tagging, etc), language specific features, or hand crafted rules, but still achieve statistically equivalent results to the best state-of-the-art systems, that employ no such limitations.

1 Introduction

Words are in general ambiguous and can have several related or unrelated meanings depending on context. For instance, the word rock can refer to both a stone and a music genre, but in the sentence "Without the guitar, there would be no rock music" the sense of rock is no longer ambiguous. The task of assigning a word token in a text, e.g. rock, to a well defined word sense in a lexicon is called word sense disambiguation (WSD). From the rock example above it is easy to see that the context surrounding the word is what disambiguates the sense. However, it may not be so obvious that this is a difficult task. To see this, consider instead the phrase "Solid rock" where changing the order of words completely changes the meaning, or "Hard rock crushes heavy metal" where individual words seem to indicate stone but together they actually define the word token as music. With this in mind, our thesis is that to do WSD well we need to go beyond bag of words and into the territory of sequence modeling.

Improved WSD would be beneficial to many natural language processing (NLP) problems, e.g. machine translation (Vickrey et al., 2005), information Retrieval, information Extraction (Navigli, 2009), and sense aware word representations (Neelakantan et al., 2015; Kågebäck et al., 2015; Nieto Piña and Johansson, 2015; Bovi et al., 2015). However, though much progress has been made in the area, many current WSD systems suffer from one or two of the following deficits. (1) Disregarding the order of words in the context which can lead to problems as described above. (2) Relying on complicated and potentially language specific hand crafted features and resources, which is a big problem particularly for resource poor languages. We aim to mitigate these problems by (1) modeling the sequence of words surrounding the target word, and (2) refrain from using any hand crafted features or external resources and instead represent the words using real valued vector representation, i.e. word embeddings. Using word embeddings has previously been shown to improve WSD (Taghipour and Ng, 2015; Johansson and Nieto Piña, 2015). However, these works did not consider the order of words or their operational effect on each other.

∗Authors contributed equally.

This work is licenced under a Creative Commons Attribution 4.0 International License. License details: http://creativecommons.org/licenses/by/4.0/
1.1 The main contributions of this work include:

• A purely learned approach to WSD that achieves results on par with state-of-the-art resource heavy systems, employing e.g. knowledge graphs, parsers, part-of-speech tagging, etc.

• Parameter sharing between different word types to make more efficient use of labeled data and make full vocabulary scaling plausible without the number of parameters exploding.

• Empirical evidence that highlights the importance of word order for WSD.

• A WSD system that, by using no explicit window, is allowed to combine local and global information when deducing the sense.

2 Background

In this section we introduce the most important underlying techniques for our proposed model.

2.1 Bidirectional LSTM

Long short-term memory (LSTM) is a gated type of recurrent neural network (RNN). LSTMs were introduced by Hochreiter and Schmidhuber (1997) to enable RNNs to better capture long term dependencies when used to model sequences. This is achieved by letting the model copy the state between timesteps without forcing the state through a non-linearity. The flow of information is instead regulated using multiplicative gates which preserves the gradient better than e.g. the logistic function. The bidirectional variant of LSTM, (BLSTM) (Graves and Schmidhuber, 2005) is an adaptation of the LSTM where the state at each time step consist of the state of two LSTMs, one going left and one going right. For WSD this means that the state has information about both preceding words and succeeding words, which in many cases are absolutely necessary to correctly classify the sense.

2.2 Word embeddings by GloVe

Word embeddings is a way to represent words as real valued vectors in a semantically meaningful space. Global Vectors for Word Representation (GloVe), introduced by Pennington et al. (2014) is a hybrid approach to embedding words that combine a log-linear model, made popular by Mikolov et al. (2013), with counting based co-occurrence statistics to more efficiently capture global statistics. Word embeddings are trained in an unsupervised fashion, typically on large amounts of data, and is able to capture fine grained semantic and syntactic information about words. These vectors can subsequently be used to initialize the input layer of a neural network or some other NLP model.

3 The Model

Given a document and the position of the target word, i.e. the word to disambiguate, the model computes a probability distribution over the possible senses corresponding to that word. The architecture of the model, depicted in Figure 1, consist of a softmax layer, a hidden layer, and a BLSTM. See Section 2.1 for more details regarding the BLSTM. The BLSTM and the hidden layer share parameters over all word types and senses, while the softmax is parameterized by word type and selects the corresponding weight matrix and bias vector for each word type respectively. This structure enables the model to share statistical strength across different word types while remaining computationally efficient even for a large total number of senses and realistic vocabulary sizes.

3.1 Model definition

The input to the BLSTM at position $n$ in document $D$ is computed as

$$x_n = W^x v(w_n), n \in \{1, \ldots, |D|\}.$$

Here, $v(w_n)$ is the one-hot representation of the word type corresponding to $w_n \in D$. A one-hot representation is a vector with dimension $V$ consisting of $|V| - 1$ zeros and a single one which index
indicate the word type. This will have the effect of picking the column from $W^x$ corresponding to that word type. The resulting vector is referred to as a word embedding. Further, $W^x$ can be initialized using pre-trained word embeddings, to leverage large unannotated datasets. In this work GloVe vectors are used for this purpose, see Section 4.1 for details.

The model output, 
\[
y(n) = \text{softmax}(W_{ay}^a y_n + b_{ay}^a),
\]
is the predicted distribution over senses for the word at position $n$, where $W_{ay}^a$ and $b_{ay}^a$ are the weights and biases for the softmax layer corresponding to the word type at position $n$. Hence, each word type will have its own softmax parameters, with dimensions depending on the number of senses of that particular word. Further, the hidden layer $a$ is computed as
\[
a = W^{ha}[h_{n-1}^L; h_{n+1}^R] + b^{ha}
\]
where $[h_{n-1}^L; h_{n+1}^R]$ is the concatenated outputs of the right and left traversing LSTMs of the BLSTM at word $n$. $W^{ha}$ and $b^{ha}$ are the weights and biases for the hidden layer.

**Loss function** The parameters of the model, $\Omega = \{W^x, \Theta_{BLSTM}, W^{ha}, b^{ha}, \{W_{ay}^a, b_{ay}^a\}_{yw \in V}\}$, are fitted by minimizing the cross entropy error
\[
L(\Omega) = -\sum_{i \in \mathcal{I}} \sum_{j \in S(w_i)} t_{i,j} \log y_j(i)
\]
over a set of sense labeled tokens with indices $\mathcal{I} \subset \{1, \ldots, |\mathcal{C}|\}$ within a training corpus $\mathcal{C}$, each labeled with a target sense $t_i$, $\forall i \in \mathcal{I}$.

### 3.2 Dropword

*Dropword* is a regularization technique very similar to *word dropout* introduced by Iyyer et al. (2015). Both methods are word level generalizations of dropout (Srivastava et al., 2014) but in word dropout the word is set to zero while in dropword it is replaced with a $<\text{dropped}>$ tag. The tag is subsequently treated just like any other word in the vocabulary and has a corresponding word embedding that is trained. This process is repeated over time, so that the words dropped change over time. The motivation for doing dropword is to decrease the dependency on individual words in the training context. This technique can be generalized to other kinds of sequential inputs, not only words.
4 Experiments

To evaluate our proposed model we perform the lexical sample task of SensEval 2 (SE2) (Kilgarriff, 2001) and SensEval 3 (SE3) (Mihalcea et al., 2004), part of the SensEval (Kilgarriff and Palmer, 2000) workshops organized by Special Interest Group on the Lexicon at ACL. For both instances of the task training and test data are supplied, and the task consist of disambiguating one indicated word in a context. The words to disambiguate are sampled from the vocabulary to give a range of low, medium and high frequency words, and a gold standard sense label is supplied for training and evaluation.

4.1 Experimental settings

The hyperparameter settings used during the experiments, presented in Table 1, were tuned on a separate validation set with data picked from the SE2 training set. The source code, implemented using TensorFlow (Abadi et al., 2015), has been released as open source.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Range searched</th>
<th>Value used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedding size</td>
<td>{100, 200}</td>
<td>100</td>
</tr>
<tr>
<td>BLSTM hidden layer size</td>
<td>[50, 100]</td>
<td>2 * 74</td>
</tr>
<tr>
<td>Dropout on word embeddings</td>
<td>[0, 50%]</td>
<td>50%</td>
</tr>
<tr>
<td>Dropout on the LSTM output</td>
<td>[0, 70%]</td>
<td>50%</td>
</tr>
<tr>
<td>Dropout on the hidden layer α</td>
<td>[0, 70%]</td>
<td>50%</td>
</tr>
<tr>
<td>Dropword</td>
<td>[0, 20%]</td>
<td>10%</td>
</tr>
<tr>
<td>Gaussian noise added to input</td>
<td>[0, 0.4]</td>
<td>∼ N(0, 0.2σ_i)</td>
</tr>
<tr>
<td>Optimization algorithm</td>
<td>-</td>
<td>Stochastic gradient descent</td>
</tr>
<tr>
<td>Momentum</td>
<td>-</td>
<td>0.1</td>
</tr>
<tr>
<td>Initial learning rate</td>
<td>-</td>
<td>2.0</td>
</tr>
<tr>
<td>Learning rate decay</td>
<td>-</td>
<td>0.96</td>
</tr>
<tr>
<td>Embedding initialization</td>
<td>-</td>
<td>GloVe</td>
</tr>
<tr>
<td>Remaining parameters initialized</td>
<td>-</td>
<td>∈ U(−0.1, 0.1)</td>
</tr>
</tbody>
</table>

Table 1: Hyperparameter settings used for both experiments and the ranges that were searched during tuning. "-" indicates that no tuning were performed on that parameter.

Embeddings The embeddings are initialized using a set of freely available GloVe vectors trained on Wikipedia and Gigaword. Words not included in this set are initialized from $N(0, 0.1)$. To keep the input noise proportional to the embeddings it is scaled by $σ_i$ which is the standard deviation in embedding dimension $i$ for all words in the embeddings matrix, $W^x$. $σ_i$ is updated after each weight update.

Data preprocessing The only preprocessing of the data that is conducted is replacing numbers with a < number > tag. This result in a vocabulary size of $|V| = 50817$ for SE2 and $|V| = 37998$ for SE3. Words not present in the training set are considered unknown during test. Further, we limit the size of the context to max 140 words centered around the target word to facilitate faster training.

4.2 Results

The results of our experiments and the state-of-the-art are shown in Table 2. 100JHU(R) was developed by Yarowsky et al. (2001) and achieved the best score on the English lexical sample task of SE2 with a F1 score of 64.2. Their system utilized a rich feature space based on raw words, lemmas, POS tags, bag-of-words, bi-gram, and tri-gram collocations, etc. as inputs to an ensemble classifier. htsa3 by Grozea (2004) was the winner of the SE3 lexical sample task with a F1 score of 72.9. This system was based mainly on raw words, lemmas, and POS tags. These were used as inputs to a regularized least square
classifier. IMS+adapted CW is a more recent system, by Taghipour and Ng (2015), that uses separately trained word embeddings as input. However, it also relies on a rich set of other features including POS tags, collocations and surrounding words to achieve their reported result.

Our proposed model achieves the top score on SE2 and are tied with IMS+adapted CW on SE3. Moreover, we see that dropword consistently improves the results on both SE2 and SE3. Randomizing the order of the input words yields a substantially worse result, which provides evidence for our hypothesis that the order of the words are significant. We also see that the system effectively makes use of the information in the pre-trained word embeddings and that they are essential to the performance of our system on these datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>SE2</th>
<th>SE3</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLSTM (our proposed model)</td>
<td>66.9</td>
<td>73.4</td>
</tr>
<tr>
<td>100JHU(R)</td>
<td>64.2</td>
<td>-</td>
</tr>
<tr>
<td>htsa3</td>
<td>-</td>
<td>72.9</td>
</tr>
<tr>
<td>IMS+adapted CW</td>
<td>66.2</td>
<td>73.4</td>
</tr>
<tr>
<td>BLSTM without dropword</td>
<td>66.5</td>
<td>72.9</td>
</tr>
<tr>
<td>BLSTM without GloVe</td>
<td>54.6</td>
<td>59.0</td>
</tr>
<tr>
<td>BLSTM, randomized word order</td>
<td>58.8</td>
<td>64.7</td>
</tr>
</tbody>
</table>

Table 2: Results for Senseval 2 and 3 on the English lexical sample task.

5 Conclusions & future work

We presented a BLSTM based model for WSD that was able to effectively exploit word order and achieve results on state-of-the-art level, using no external resources or handcrafted features. As a consequence, the model is largely language independent and applicable to resource poor languages. Further, the system was designed to generalize to full vocabulary WSD by sharing most of the parameters between words.

For future work we would like to provide more empirical evidence for language independence by evaluating on several different languages, and do experiments on large vocabulary all words WSD, where every word in a sentence is disambiguated. Further, we plan to experiment with unsupervised pre-training of the BLSTM, encouraged by the substantial improvement achieved by incorporating word embeddings.

Acknowledgments

The authors would like to acknowledge the project Towards a knowledge-based culturomics supported by a framework grant from the Swedish Research Council (2012–2016; dnr 2012-5738). Further, the work was partly funded by the BigData@Chalmers initiative. Finally, the authors would like to thank the reviewers for their insightful and constructive feedback.

References


Extractive summarization using continuous vector space models

M. Kågebäck, O. Mogren, N. Tahmasebi, and D. Dubhashi

Reprinted from Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)@ EACL, 2014
Abstract
Automatic summarization can help users extract the most important pieces of information from the vast amount of text digitized into electronic form everyday. Central to automatic summarization is the notion of similarity between sentences in text. In this paper we propose the use of continuous vector representations for semantically aware representations of sentences as a basis for measuring similarity. We evaluate different compositions for sentence representation on a standard dataset using the ROUGE evaluation measures. Our experiments show that the evaluated methods improve the performance of a state-of-the-art summarization framework and strongly indicate the benefits of continuous word vector representations for automatic summarization.

1 Introduction
The goal of summarization is to capture the important information contained in large volumes of text, and present it in a brief, representative, and consistent summary. A well written summary can significantly reduce the amount of work needed to digest large amounts of text on a given topic. The creation of summaries is currently a task best handled by humans. However, with the explosion of available textual data, it is no longer financially possible, or feasible, to produce all types of summaries by hand. This is especially true if the subject matter has a narrow base of interest, either due to the number of potential readers or the duration during which it is of general interest. A summary describing the events of World War II might for instance be justified to create manually, while a summary of all reviews and comments regarding a certain version of Windows might not. In such cases, automatic summarization is a way forward.

In this paper we introduce a novel application of continuous vector representations to the problem of multi-document summarization. We evaluate different compositions for producing sentence representations based on two different word embeddings on a standard dataset using the ROUGE evaluation measures. Our experiments show that the evaluated methods improve the performance of a state-of-the-art summarization framework which strongly indicate the benefits of continuous word vector representations for this task.

2 Summarization
There are two major types of automatic summarization techniques, extractive and abstractive. Extractive summarization systems create summaries using representative sentences chosen from the input while abstractive summarization creates new sentences and is generally considered a more difficult problem.

Figure 1: Illustration of Extractive Multi-Document Summarization.

For this paper we consider extractive multi-document summarization, that is, sentences are chosen for inclusion in a summary from a set of documents $D$. Typically, extractive summarization techniques can be divided into two components, the summarization framework and the similarity measures used to compare sentences. Next
we present the algorithm used for the framework and in Sec. 2.2 we discuss a typical sentence similarity measure, later to be used as a baseline.

2.1 Submodular Optimization

Lin and Bilmes (2011) formulated the problem of extractive summarization as an optimization problem using monotone nondecreasing submodular set functions. A submodular function $F$ on the set of sentences $V$ satisfies the following property: for any $A \subseteq B \subseteq V \setminus \{v\}$, $F(A + \{v\}) - F(A) \geq F(B + \{v\}) - F(B)$ where $v \in V$. This is called the diminishing returns property and captures the intuition that adding a sentence to a small set of sentences (i.e., summary) makes a greater contribution than adding a sentence to a larger set. The aim is then to find a summary that maximizes diversity of the sentences and the coverage of the input text. This objective function can be formulated as follows:

$$\mathcal{F}(S) = \mathcal{L}(S) + \lambda \mathcal{R}(S)$$

where $S$ is the summary, $\mathcal{L}(S)$ is the coverage of the input text, $\mathcal{R}(S)$ is a diversity reward function. The $\lambda$ is a trade-off coefficient that allows us to define the importance of coverage versus diversity of the summary. In general, this kind of optimization problem is NP-hard, however, if the objective function is submodular there is a fast scalable algorithm that returns an approximation with a guarantee. In the work of Lin and Bilmes (2011) a simple submodular function is chosen:

$$\mathcal{L}(S) = \sum_{i \in V} \min \{ \sum_{j \in S} \text{Sim}(i, j), \alpha \sum_{j \in V} \text{Sim}(i, j) \}$$

(1)

The first argument measures similarity between sentence $i$ and the summary $S$, while the second argument measures similarity between sentence $i$ and the rest of the input $V$. $\text{Sim}(i, j)$ is the similarity between sentence $i$ and sentence $j$ and $0 \leq \alpha \leq 1$ is a threshold coefficient. The diversity reward function $\mathcal{R}(S)$ can be found in (Lin and Bilmes, 2011).

2.2 Traditional Similarity Measure

Central to most extractive summarization systems is the use of sentence similarity measures ($\text{Sim}(i, j)$) in Eq. 1. Lin and Bilmes measure similarity between sentences by representing each sentence using $tf$-$idf$ (Salton and McGill, 1986) vectors and measuring the cosine angle between vectors. Each sentence is represented by a word vector $w = (w_1, \ldots, w_N)$ where $N$ is the size of the vocabulary. Weights $w_k$ correspond to the $tf$-$idf$ value of word $k$ in the sentence $i$. The weights $\text{Sim}(i, j)$ used in the $\mathcal{L}$ function in Eq. 1 are found using the following similarity measure.

\[ \text{Sim}(i, j) = \frac{\sum_{w \in i} tf_{w,i} \times tf_{w,j} \times idf^2_w \sqrt{\sum_{w \in i} tf^2_{w,i} \times idf^2_w} \sqrt{\sum_{w \in j} tf^2_{w,j} \times idf^2_w}}{\sqrt{\sum_{w \in i} tf^2_{w,i} \times idf^2_w} \sqrt{\sum_{w \in j} tf^2_{w,j} \times idf^2_w}} \]

(2)

where $tf_{w,i}$ and $tf_{w,j}$ are the number of occurrences of $w$ in sentence $i$ and $j$, and idf$_w$ is the inverse document frequency (idf) of $w$.

In order to have a high similarity between sentences using the above measure, two sentences must have an overlap of highly scored $tf$-$idf$ words. The overlap must be exact to count towards the similarity, e.g. the terms The US President and Barack Obama in different sentences will not add towards the similarity of the sentences. To capture deeper similarity, in this paper we will investigate the use of continuous vector representations for measuring similarity between sentences. In the next sections we will describe the basics needed for creating continuous vector representations and methods used to create sentence representations that can be used to measure sentence similarity.

3 Background on Deep Learning

Deep learning (Hinton et al., 2006; Bengio, 2009) is a modern interpretation of artificial neural networks (ANN), with an emphasis on deep network architectures. Deep learning can be used for challenging problems like image and speech recognition (Krizhevsky et al., 2012; Graves et al., 2013), as well as language modeling (Mikolov et al., 2010), and in all cases, able to achieve state-of-the-art results.

Inspired by the brain, ANNs use a neuron-like construction as their primary computational unit. The behavior of a neuron is entirely controlled by its input weights. Hence, the weights are where the information learned by the neuron is stored. More precisely the output of a neuron is computed as the weighted sum of its inputs, and squeezed into the interval $[0, 1]$ using a sigmoid function:

\[ y_i = g(\theta^T x) \]

(3)

\[ g(z) = \frac{1}{1 + e^{-z}} \]

(4)
Figure 2: FFNN with four input neurons, one hidden layer, and 1 output neuron. This type of architecture is appropriate for binary classification of some data $x \in \mathbb{R}^4$, however depending on the complexity of the input, the number and size of the hidden layers should be scaled accordingly.

where $\theta_i$ are the weights associated with neuron $i$ and $x$ is the input. Here the sigmoid function ($g$) is chosen to be the logistic function, but it may also be modeled using other sigmoid shaped functions, e.g. the hyperbolic tangent function.

The neurons can be organized in many different ways. In some architectures, loops are permitted. These are referred to as recurrent neural networks. However, all networks considered here are non-cyclic topologies. In the rest of this section we discuss a few general architectures in more detail, which will later be employed in the evaluated models.

3.1 Feed Forward Neural Network

A feed forward neural network (FFNN) (Haykin, 2009) is a type of ANN where the neurons are structured in layers, and only connections to subsequent layers are allowed, see Fig 2. The algorithm is similar to logistic regression using non-linear terms. However, it does not rely on the user to choose the non-linear terms needed to fit the data, making it more adaptable to changing datasets. The first layer in a FFNN is called the input layer, the last layer is called the output layer, and the interim layers are called hidden layers. The hidden layers are optional but necessary to fit complex patterns.

Training is achieved by minimizing the network error ($E$). How $E$ is defined differs between different network architectures, but is in general a differentiable function of the produced output and the expected output. In order to minimize this function the gradient $\frac{\partial E}{\partial \Theta}$ first needs to be calculated, where $\Theta$ is a matrix of all parameters, or weights, in the network. This is achieved using backpropagation (Rumelhart et al., 1986). Secondly, these gradients are used to minimize $E$ using e.g. gradient descent. The result of this processes is a set of weights that enables the network to do the desired input-output mapping, as defined by the training data.

3.2 Auto-Encoder

An auto-encoder (AE) (Hinton and Salakhutdinov, 2006), see Fig. 3, is a type of FFNN with a topology designed for dimensionality reduction. The input and the output layers in an AE are identical, and there is at least one hidden bottleneck layer that is referred to as the coding layer. The network is trained to reconstruct the input data, and if it succeeds this implies that all information in the data is necessarily contained in the compressed representation of the coding layer.

A shallow AE, i.e. an AE with no extra hidden layers, will produce a similar code as principal component analysis. However, if more layers are added, before and after the coding layer, non-linear manifolds can be found. This enables the network to compress complex data, with minimal loss of information.

3.3 Recursive Neural Network

A recursive neural network (RvNN), see Fig. 4, first presented by Socher et al. (2010), is a type of feed forward neural network that can process data through an arbitrary binary tree structure, e.g. a
binary parse tree produced by linguistic parsing of a sentence. This is achieved by enforcing weight constraints across all nodes and restricting the output of each node to have the same dimensionality as its children.

The input data is placed in the leaf nodes of the tree, and the structure of this tree is used to guide the recursion up to the root node. A compressed representation is calculated recursively at each non-terminal node in the tree, using the same weight matrix at each node. More precisely, the following formulas can be used:

\[ z_p = \theta^T_p [x_l; x_r] \]  
\[ y_p = g(z_p) \]

where \( y_p \) is the computed parent state of neuron \( p \), and \( z_p \) the induced field for the same neuron. \([x_l; x_r]\) is the concatenation of the state belonging to the right and left sibling nodes. This process results in a fixed length representation for hierarchical data of arbitrary length. Training of the model is done using backpropagation through structure, introduced by Goller and Kuchler (1996).

4 Word Embeddings

Continuous distributed vector representation of words, also referred to as word embeddings, was first introduced by Bengio et al. (2003). A word embedding is a continuous vector representation that captures semantic and syntactic information about a word. These representations can be used to unveil dimensions of similarity between words, e.g., singular or plural.

4.1 Collobert & Weston

Collobert and Weston (2008) introduce an efficient method for computing word embeddings, in this work referred to as \( CW \) vectors. This is achieved firstly, by scoring a valid n-gram (\( x \)) and a corrupted n-gram (\( \bar{x} \)) (where the center word has been randomly chosen), and secondly, by training the network to distinguish between these two n-grams. This is done by minimizing the hinge loss

\[
\max(0, 1 - s(x) + s(\bar{x})) \]

where \( s \) is the scoring function, i.e., the output of a FFNN that maps between the word embeddings of an n-gram to a real valued score. Both the parameters of the scoring function and the word embeddings are learned in parallel using backpropagation.

4.2 Continuous Skip-gram

A second method for computing word embeddings is the Continuous Skip-gram model, see Fig. 5, introduced by Mikolov et al. (2013a). This model is used in the implementation of their word embeddings tool Word2Vec. The model is trained to predict the context surrounding a given word. This is accomplished by maximizing the objective function

\[
\frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j}|w_t) 
\]

where \( T \) is the number of words in the training set, and \( c \) is the length of the training context. The probability \( p(w_{t+j}|w_t) \) is approximated using the hierarchical softmax introduced by Bengio et al. (2002) and evaluated in a paper by Morin and Bengio (2005).

5 Phrase Embeddings

Word embeddings have proven useful in many natural language processing (NLP) tasks. For summarization, however, sentences need to be compared. In this section we present two different methods for deriving phrase embeddings, which in Section 5.3 will be used to compute sentence to sentence similarities.

5.1 Vector addition

The simplest way to represent a sentence is to consider it as the sum of all words without regarding word orders. This was considered by...
Mikolov et al. (2013b) for representing short phrases. The model is expressed by the following equation:

\[ x_p = \sum_{x_w \in \{\text{sentence}\}} x_w \]  

where \( x_p \) is a phrase embedding, and \( x_w \) is a word embedding. We use this method for computing phrase embeddings as a baseline in our experiments.

### 5.2 Unfolding Recursive Auto-encoder

The second model is more sophisticated, taking into account also the order of the words and the grammar used. An unfolding recursive auto-encoder (RAE) is used to derive the phrase embedding on the basis of a binary parse tree. The unfolding RAE was introduced by Socher et al. (2011) and uses two RvNNs, one for encoding the compressed representations, and one for decoding them to recover the original sentence, see Figure 6. The network is subsequently trained by minimizing the reconstruction error.

Forward propagation in the network is done by recursively applying Eq. 5a and 5b for each triplet in the tree in two phases. First, starting at the center node (root of the tree) and recursively pulling the data from the input. Second, again starting at the center node, recursively pushing the data towards the output. Backpropagation is done in a similar manner using backpropagation through structure (Goller and Kuchler, 1996).

### 5.3 Measuring Similarity

Phrase embeddings provide semantically aware representations for sentences. For summarization, we need to measure the similarity between two representations and will make use of the following two vector similarity measures. The first similarity measure is the cosine similarity, transformed to the interval of \([0, 1]\)

\[ \text{Sim}(i, j) = \left( \frac{x_i^T x_j}{\|x_j\| \|x_j\|} + 1 \right) / 2 \]  

where \( x \) denotes a phrase embedding The second similarity is based on the complement of the Euclidean distance and computed as:

\[ \text{Sim}(i, j) = 1 - \frac{1}{\max_{k,n} \sqrt{\|x_k - x_n\|^2}} \sqrt{\|x_j - x_i\|^2} \]  

### 6 Experiments

In order to evaluate phrase embeddings for summarization we conduct several experiments and compare different phrase embeddings with \(tf-idf\) based vectors.

#### 6.1 Experimental Settings

Seven different configuration were evaluated. The first configuration provides us with a baseline and is denoted \(\text{Original}\) for the Lin-Bilmes method described in Sec. 2.1. The remaining configurations comprise selected combinations of word embeddings, phrase embeddings, and similarity measures.
The first group of configurations are based on
vector addition using both Word2Vec and CW vec-
tors. These vectors are subsequently compared us-
ing both cosine similarity and Euclidean distance.
The second group of configurations are built upon
recursive auto-encoders using CW vectors and are
also compared using cosine similarity as well as Euclidean distance.

The methods are named according to:
VectorType, EmbeddingMethod, SimilarityMethod,
e.g. W2V_Add_Cos for Word2Vec vectors com-
bined using vector addition and compared using
cosine similarity.

To get an upper bound for each ROUGE score
an exhaustive search were performed, where each
possible pair of sentences were evaluated, and
maximized w.r.t. the ROUGE score.

6.2 Dataset and Evaluation
The Opinosis dataset (Ganesan et al., 2010) con-
sists of short user reviews in 51 different top-
ics. Each of these topics contains between 50 and
575 sentences and are a collection of user reviews
made by different authors about a certain charac-
teristic of a hotel, car or a product (e.g. "Loca-
tion of Holiday Inn, London" and "Fonts, Ama-
zon Kindle"). The dataset is well suited for multi-
document summarization (each sentence is con-
sidered its own document), and includes between
4 and 5 gold-standard summaries (not sentences
chosen from the documents) created by human au-
thors for each topic.

Each summary is evaluated with ROUGE, that
works by counting word overlaps between gener-
sated summaries and gold standard summaries. Our
results include R-1, R-2, and R-SU4, which counts
matches in unigrams, bigrams, and skip-bigrams
respectively. The skip-bigrams allow four words
in between (Lin, 2004).

The measures reported are recall (R), precision
(P), and F-score (F), computed for each topic indi-
vidually and averaged. Recall measures what frac-
tion of a human created gold standard summary
that is captured, and precision measures what frac-
tion of the generated summary that is in the gold
standard. F-score is a standard way to combine
recall and precision, computed as $F = 2 \frac{P \times R}{P + R}$.

6.3 Implementation
All results were obtained by running an imple-
mentation of Lin-Bilmes submodular optimization
summarizer, as described in Sec. 2.1. Also, we
have chosen to fix the length of the summaries
to two sentences because the length of the gold-
standard summaries are typically around two sen-
tences. The CW vectors used were trained by
Turian et al. (2010)$^1$, and the Word2Vec vectors
by Mikolov et al. (2013b)$^2$. The unfolding RAE
used is based on the implementation by Socher
et al. (2011)$^3$, and the parse trees for guiding
the recursion was generated using the Stanford
Parser (Klein and Manning, 2003)$^4$.

6.4 Results
The results from the ROUGE evaluation are com-
piled in Table 1. We find for all measures (recall,
precision, and F-score), that the phrase embed-
ddings outperform the original Linn-Bilmes. For re-
call, we find that CW_Add_Cos achieves the high-
est result, while for precision and F-score the
CW_Add_Euc perform best. These results are con-
sistent for all versions of ROUGE scores reported
(1, 2 and SU4), providing a strong indication for
phrase embeddings in the context of automatic
summarization.

Unfolding RAE on CW vectors and vector addi-
tion on W2V vectors gave comparable results
w.r.t. each other, generally performing better than
original Linn-Bilmes but not performing as well as
vector addition of CW vectors.

The results denoted OPT in Table 1 describe
the upper bound score, where each row repres-
ts optimal recall and F-score respectively. The
best results are achieved for R-1 with a max-
imum recall of 57.86%. This is a consequence of
hand created gold standard summaries used in the
evaluation, that is, we cannot achieve full recall
or F-score when the sentences in the gold stan-
dard summaries are not taken from the underly-
ing documents and thus, they can never be fully
matched using extractive summarization. R-2 and
SU4 have lower maximum recall and F-score, with
22.9% and 29.5% respectively.

6.5 Discussion
The results of this paper show great potential for
employing word and phrase embeddings in sum-
marization. We believe that by using embeddings
we move towards more semantically aware sum-
marization systems. In the future, we anticipate

1$^http://metaoptimize.com/projects/wordreprs/$
2$^https://code.google.com/p/word2vec/$
4$^http://nlp.stanford.edu/software/lex-parser.shtml$
improvements for the field of automatic summarization as the quality of the word vectors improve and we find enhanced ways of composing and comparing the vectors.

It is interesting to compare the results of different composition techniques on the CW vectors, where vector addition surprisingly outperforms the considerably more sophisticated unfolding RAE. However, since the unfolding RAE uses syntactic information, this may be a result of using a dataset consisting of low quality text.

In the interest of comparing word embeddings, results using vector addition and cosine similarity were computed based on both CW and Word2Vec vectors. Supported by the achieved results CW vectors seems better suited for sentence similarities in this setting.

An issue we encountered with using precomputed word embeddings was their limited vocabulary, in particular missing uncommon (or common incorrect) spellings. This problem is particularly pronounced on the evaluated Opinosis dataset, since the text is of low quality. Future work is to train word embeddings on a dataset used for summarization to better capture the specific semantics and vocabulary.

The optimal R-1 scores are higher than R-2 and SU4 (see Table 1) most likely because the score ignores word order and considers each sentence as a set of words. We come closest to the optimal score for R-1, where we achieve 60% of maximal recall and 49% of F-score. Future work is to investigate why we achieve a much lower recall and F-score for the other ROUGE scores.

Our results suggest that the phrase embeddings capture the kind of information that is needed for the summarization task. The embeddings are the underpinnings of the decisions on which sentences that are representative of the whole input text, and which sentences that would be redundant when combined in a summary. However, the fact that we at most achieve 60% of maximal recall suggests that the phrase embeddings are not complete w.r.t summarization and might benefit from being combined with other similarity measures that can capture complementary information, for example using multiple kernel learning.

7 Related Work

To the best of our knowledge, continuous vector space models have not previously been used in summarization tasks. Therefore, we split this section in two, handling summarization and continuous vector space models separately.

7.1 Continuous Vector Space Models

Continuous distributed vector representation of words was first introduced by Bengio et al. (2003).
They employ a FFNN, using a window of words as input, and train the model to predict the next word. This is computed using a big softmax layer that calculate the probabilities for each word in the vocabulary. This type of exhaustive estimation is necessary in some NLP applications, but makes the model heavy to train.

If the sole purpose of the model is to derive word embeddings this can be exploited by using a much lighter output layer. This was suggested by Collobert and Weston (2008), which swapped the heavy softmax against a hinge loss function. The model works by scoring a set of consecutive words, distorting one of the words, scoring the distorted set, and finally training the network to give the correct set a higher score.

Taking the lighter concept even further, Mikolov et al. (2013a) introduced a model called Continuous Skip-gram. This model is trained to predict the context surrounding a given word using a shallow neural network. The model is less aware of the order of words, than the previously mentioned models, but can be trained efficiently on considerably larger datasets.

An early attempt at merging word representations into representations for phrases and sentences is introduced by Socher et al. (2010). The authors present a recursive neural network architecture (RvNN) that is able to jointly learn parsing and phrase/sentence representation. Though not able to achieve state-of-the-art results, the method provides an interesting path forward. The model uses one neural network to derive all merged representations, applied recursively in a binary parse tree. This makes the model fast and easy to train but requires labeled data for training.

### 7.2 Summarization Techniques

Radev et al. (2004) pioneered the use of cluster centroids in their work with the idea to group, in the same cluster, those sentences which are highly similar to each other, thus generating a number of clusters. To measure the similarity between a pair of sentences, the authors use the cosine similarity measure where sentences are represented as weighted vectors of tf-idf terms. Once sentences are clustered, sentence selection is performed by selecting a subset of sentences from each cluster.

In TextRank (2004), a document is represented as a graph where each sentence is denoted by a vertex and pairwise similarities between sentences are represented by edges with a weight corresponding to the similarity between the sentences. The Google PageRank ranking algorithm is used to estimate the importance of different sentences and the most important sentences are chosen for inclusion in the summary.

Bonzanini, Martinez, Roelleke (2013) presented an algorithm that starts with the set of all sentences in the summary and then iteratively chooses sentences that are unimportant and removes them. The sentence removal algorithm obtained good results on the Opinosis dataset, in particular w.r.t F-scores.

We have chosen to compare our work with that of Lin and Bilmes (2011), described in Sec. 2.1. Future work is to make an exhaustive comparison using a larger set similarity measures and summarization frameworks.

### 8 Conclusions

We investigated the effects of using phrase embeddings for summarization, and showed that these can significantly improve the performance of the state-of-the-art summarization method introduced by Lin and Bilmes in (2011). Two implementations of word vectors and two different approaches for composition where evaluated. All investigated combinations improved the original Lin-Bilmes approach (using tf-idf representations of sentences) for at least two ROUGE scores, and top results where found using vector addition on CW vectors.

In order to further investigate the applicability of continuous vector representations for summarization, in future work we plan to try other summarization methods. In particular we will use a method based on multiple kernel learning were phrase embeddings can be combined with other similarity measures. Furthermore, we aim to use a novel method for sentence representation similar to the RAE using multiplicative connections controlled by the local context in the sentence.

### Acknowledgments

The authors would like to acknowledge the project Toward a knowledge-based culturomics supported by a framework grant from the Swedish Research Council (2012–2016; dnr 2012-5738), and the project Data-driven secure business intelligence grant IIS11-0089 from the Swedish Foundation for Strategic Research (SSF).
References


Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In IEEE International Conference on Neural Networks, volume 1, pages 347–352. IEEE.


Extractive Summarization by Aggregating Multiple Similarities

O. Mogren, M. Kågebäck, and D. Dubhashi

Reprinted from Proceedings of Recent Advances in Natural Language Processing, 2015
Abstract

News reports, social media streams, blogs, digitized archives and books are part of a plethora of reading sources that people face every day. This raises the question of how to best generate automatic summaries. Many existing methods for extracting summaries rely on comparing the similarity of two sentences in some way. We present new ways of measuring this similarity, based on sentiment analysis and continuous vector space representations, and show that combining these together with similarity measures from existing methods, helps to create better summaries. The finding is demonstrated with MULTSUM, a novel summarization method that uses ideas from kernel methods to combine sentence similarity measures. Submodular optimization is then used to produce summaries that take several different similarity measures into account. Our method improves over the state-of-the-art on standard benchmark datasets; it is also fast and scale to large document collections, and the results are statistically significant.

1 Introduction

Extractive summarization, the process of selecting a subset of sentences from a set of documents, is an important component of modern information retrieval systems (Baeza-Yates et al., 1999). A good summarization system needs to balance two complementary aspects: finding a summary that captures all the important topics of the documents (coverage), yet does not contain too many similar sentences (non-redundancy). It follows that it is essential to have a good way of measuring the similarity of sentences, in no way a trivial task. Consequently, several measures for sentence similarity have been explored for extractive summarization.

In this work, two sets of novel similarity measures capturing deeper semantic features are presented, and evaluated in combination with existing methods of measuring sentence similarity. The new methods are based on sentiment analysis, and continuous vector space representations of phrases, respectively.

We show that summary quality is improved by combining multiple similarities at the same time using kernel techniques. This is demonstrated using MULTSUM, an ensemble-approach to generic extractive multi-document summarization based on the existing, state-of-the-art method of Lin and Bilmes (2011). Our method obtains state-of-the-art results that are statistically significant on the de-facto standard benchmark dataset DUC 04. The experimental evaluation also confirm that the method generalizes well to other datasets.

2 MULTSUM

MULTSUM, our approach for extractive summarization, finds representative summaries taking multiple sentence similarity measures into account. As Lin and Bilmes (2011), we formulate the problem as the optimization of monotone non-decreasing submodular set functions. This results in a fast, greedy optimization step that provides a \( (1 - \frac{1}{e}) \) factor approximation. In the original version, the optimization objective is a function scoring a candidate summary by coverage and diversity, expressed using cosine similarity between sentences represented as bag-of-terms vectors. We extend this method by using several sentence similarity measures \( M^l \) (as described in Section 3) at the same time, combined by multiplying them together element-wise:

\[
M_{s_i, s_j} = \prod M^l_{s_i, s_j}.
\]
In the literature of kernel methods, this is the standard way of combining kernels as a conjunction (Duvenaud, 2014; Schölkopf et al., 2004, Ch 1).

3 Sentence Similarity Measures

Many existing systems rely on measuring the similarity of sentences to balance the coverage with the amount of redundancy of the summary. This is also true for MULTSUM which is based on the existing submodular optimization method. Similarity measures that capture general aspects let the summarization system pick sentences that are representative and diverse in general. Similarity measures capturing more specific aspects allow the summarization system to take these aspects into account.

We list some existing measures in Table 3 (that mainly relies on counting word overlaps) and in Sections 3.1 and 3.2, we present sentence similarity measures that capture more specific aspects of the text. MULTSUM is designed to work with all measures mentioned below; this will be evaluated in Section 4. Interested readers are referred to a survey of existing similarity measures from the literature in (Bengtsson and Skeppstedt, 2012). All these similarity measures require sentence splitting, tokenization, part-of-speech tagging and stemming of words. The Filtered Word, and TextRank comparers are set similarity measures where each sentence is represented by the set of all their terms. The KeyWord comparer and LinTFIDF represent each sentence as a word vector and uses the vectors for measuring similarity.

DepGraph first computes the dependency parse trees of the two sentences using MaltParser (Nivre, 2003). The length of their longest common path is then used to derive the similarity score.

The similarity measure used in TextRank (Michalcea and Tarau, 2004) will be referred to as TR-Comparer. The measure used in submodular optimization (Lin and Bilmes, 2011) will be referred to as LinTFIDF. All measures used in this work are normalized, $M_{s_i,s_j} \in [0,1]$.

3.1 Sentiment Similarity

Sentiment analysis has previously been used for document summarization, with the aim of capturing an average sentiment of the input corpus (Lerman et al., 2009), or to score emotionally charged sentences (Nishikawa et al., 2010). Other research

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filtered</td>
<td>$M_{s_i,s_j} = \frac{</td>
</tr>
<tr>
<td>TRCmp.</td>
<td>$M_{s_i,s_j} = \frac{</td>
</tr>
<tr>
<td>LinTFIDF</td>
<td>$M_{s_i,s_j} = \frac{\sum_{w \in s_i} tvf_{w,i} tvf_{w,j} \cdot df_w}{\sqrt{\sum_{w \in s_i} tvf_{w,i} tvf_{w,i} \cdot df_w \cdot \sum_{w \in s_j} tvf_{w,j} tvf_{w,j} \cdot df_w}}$</td>
</tr>
<tr>
<td>KeyWord</td>
<td>$M_{s_i,s_j} = \frac{\sum_{w \in (s_i \cap s_j) \cap K} tvf_{w,i} tvf_{w,j} \cdot df_w}{</td>
</tr>
<tr>
<td>DepGraph</td>
<td>See text description.</td>
</tr>
</tbody>
</table>

Table 1: Similarity measures from previous works.

has shown that negative emotion words appear at a relative higher rate in summaries written by humans (Hong and Nenkova, 2014). We propose a different way of making summaries sentiment aware by comparing the level of sentiment in sentences. This allows for summaries that are both representative and diverse in sentiment.

Two lists, of positive and of negative sentiment words respectively, were manually created\(^1\) and used. Firstly, each sentence $s_i$ is given two sentiment scores, $positive(s_i)$ and $negative(s_i)$, defined as the fraction of words in $s_i$ that is found in the positive and the negative list, respectively. The similarity score for positive sentiment are computed as follows:

$$M_{s_i,s_j} = 1 - |positive(s_i) - positive(s_j)|$$

The similarity score for negative sentiment are computed as follows:

$$M_{s_i,s_j} = 1 - |negative(s_i) - negative(s_j)|$$

3.2 Continuous Vector Space Representations

Continuous vector space representations of words has a long history. Recently, the use of deep learning methods has given rise to a new class of continuous vector space models. Bengio et al. (2006) presented vector space representations for words that capture semantic and syntactic properties. These vectors can be employed not only to find similar words, but also to relate words using multiple dimensions of similarity. This means that words sharing some sense can be related using

\(^1\)To download the sentiment word lists used, please see http://www.mogren.one/
translations in vector space, e.g. $v_{\text{king}} - v_{\text{man}} + v_{\text{woman}} \approx v_{\text{queen}}$.

Early work on extractive summarization using vector space models was presented in (Kågebäck et al., 2014). In this work we use a similar approach, with two different methods of deriving word embeddings. The first model ($CW$) was introduced by Collobert and Weston (2008). The second ($W^2V$) is the skip-gram model by Mikolov et al. (2013).

The Collobert and Weston vectors were trained on the RCV1 corpus, containing one year of Reuters news wire; the skip-gram vectors were trained on 300 billion words from Google News.

The word embeddings are subsequently used as building blocks for sentence level phrase embeddings by summing the word vectors of each sentence. Finally, the sentence similarity is defined as the cosine similarity between the sentence vectors.

With MULTSUM, these similarity measures can be combined with the traditional sentence similarity measures.

4 Experiments

Our version of the submodular optimization code follows the description by Lin and Bilmes (2011), with the exception that we use multiplicative combinations of the sentence similarity scores described in Section 3. The source code of our system can be downloaded from \url{http://www.mogren.one/}. Where nothing else is stated, MULTSUM was evaluated with a multiplicative combination of TRComparer and FilteredWordComparer.

4.1 Datasets

In the evaluation, three different datasets were used. DUC 02 and DUC 04 are from the Document Understanding Conferences, both with the settings of task 2 (short multi-document summarization), and each consisting of around 50 document sets. Each document set is comprised of around ten news articles (between 111 and 660 sentences) and accompanied with four gold-standard summaries created by manual summarizers. The summaries are at most 665 characters long. DUC 04 is the de-facto standard benchmark dataset for generic multi-document summarization.

Experiments were also carried out on Opinosis (Ganesan et al., 2010), a collection of short user reviews in 51 different topics. Each topic consists of between 50 and 575 one-sentence user reviews by different authors about a certain characteristic of a hotel, a car, or a product. The dataset includes 4 to 5 gold-standard summaries created by human authors for each topic. The gold-standard summaries is around 2 sentences.

4.2 Baseline Methods

Our baseline methods are Submodular optimization (Lin and Bilmes, 2011), DPP (Kulesza and Taskar, 2012), and ICSI (Gillick et al., 2008). The baseline scores are calculated on precomputed summary outputs (Hong et al., 2014).

4.3 Evaluation Method

Following standard procedure, we use ROUGE (version 1.5.5) for evaluation (Lin, 2004). ROUGE counts n-gram overlaps between generated summaries and the gold standard. We have concentrated on recall as this is the measure with highest correlation to human judgement (Lin and Hovy, 2003), on ROUGE-1, ROUGE-2, and ROUGE-SU4, representing matches in unigrams, bigrams, and skip-bigrams, respectively.

The Opinosis experiments were aligned with those of Bonzanini et al. (2013) and Ganesan et al. (2010)\textsuperscript{2}. Summary length was 2 sentences. In the DUC experiments, summary length is 100 words\textsuperscript{3}.

5 Results

Our experimental results show significant improvements by aggregating several sentence similarity measures, and our results for ROUGE-2 and ROUGE-SU4 recall beats state-of-the-art.

5.1 Integrating Different Similarity Measures

Table 2 shows ROUGE recall on DUC 04. MULTSUM\textsuperscript{4} obtains ROUGE scores beating state-of-the-art systems, in particular on ROUGE-2 and ROUGE-SU4, suggesting that MULTSUM produce summaries with excellent fluency. We also note, that using combined similarities, we beat original submodular optimization.

Figure 5.1 shows, for each $n \in [1..9]$, the highest ROUGE-1 recall score obtained by MULTSUM, determined by exhaustive search.

\textsuperscript{2}ROUGE options on Opinosis: -a -m -s -x -n 2 -2 4 -u.

\textsuperscript{3}ROUGE options on DUC: -a -n 2 -m -l 100 -x -c 95 -r 1000 -f A -p 0.5 -t 0 -2 4 -u.

\textsuperscript{4}Here, MULTSUM is using TRComparer and FilteredWordComparer in multiplicative conjunction.
Table 2: ROUGE recall scores on DUC 04. Our system MULTSUM obtains the best result yet for ROUGE-2 and ROUGE-SU4. DPP has a higher ROUGE-1 score, but the difference is not statistically significant (Hong et al., 2014).

<table>
<thead>
<tr>
<th></th>
<th>ROUGE-1</th>
<th>ROUGE-2</th>
<th>ROUGE-SU4</th>
</tr>
</thead>
<tbody>
<tr>
<td>MULTSUM</td>
<td>39.35</td>
<td>9.94</td>
<td>14.01</td>
</tr>
<tr>
<td>ICSISumm</td>
<td>38.41</td>
<td>9.77</td>
<td>13.62</td>
</tr>
<tr>
<td>DPP</td>
<td>39.83</td>
<td>9.62</td>
<td>13.86</td>
</tr>
<tr>
<td>SUBMOD</td>
<td>39.18</td>
<td>9.35</td>
<td>13.75</td>
</tr>
</tbody>
</table>

Table 3: $p$-values from the Mann-Whitney U-test for combinations of similarity measures of size $n \in [1..4]$, compared to using just one similarity measure. Using 2, 3, or 4 similarity measures at the same time with MULTSUM, gives a statistically significant improvement of the ROUGE-1 scores. Dataset: DUC 04.

Table 3 shows $p$-values obtained using the Mann-Whitney U-test (Mann et al., 1947) on the ROUGE-1 scores when using a combination of $n$ similarities with MULTSUM, compared to using only one measure. The Mann-Whitney U-test compares two ranked lists $A$ and $B$, and decides whether they are from the same population. Here, $A$ is the list of scores from using only one measure, and $B$ is the top-10 ranked combinations of $n$ combined similarity measures, $n \in [1..4]$. One can see that for each $n \in [1..4]$, using $n$ sentence similarity measures at the same time, is significantly better than using only one.

On DUC 02, the best combination of similarity measures is using CW, LinTFIDF, NegativeSentiment, and TRComparer. Each point in Figure 5.1 represents a combination of some of these four similarity measures. Let $n$ be the number of measures in such a combination. When $n = 1$, the “combinations” are just single similarity measures. When $n = 2$, there are 6 different ways to choose, and when $n = 3$, there are four. A line goes from each measure point through all combinations the measure is part of. One can clearly see the benefits of each of the combination steps, as $n$ increases.

5.2 Evaluation with Single Similarity Measures

In order to understand the effect of different similarity measures, MULTSUM was first evaluated using only one similarity measure at a time. Table 4 shows the ROUGE recall scores of these experiments, using the similarity measures presented in Section 3, on DUC 04.

We note that MULTSUM provides summaries of high quality already with one similarity measure (e.g. with TRComparer), with a ROUGE-1 recall of 37.95. Using only sentiment analysis as the single similarity measure does not capture enough information to produce state-of-the-art summaries.
Figure 2: ROUGE-1 recall for the top-performing four-combination on DUC 2002 (CW, LinTFIDF, NegativeSentiment, and TRComparer), and all possible subsets of these four similarity measures. (When the number of similarity measures is one, only a single measure is used).

6 Discussion

Empirical evaluation of the method proposed in this paper shows that using several sentence similarity measures at the same time produces significantly better summaries.

When using one single similarity at a time, using sentiment similarity and vector space models does not give the best summaries. However, we found that when combining several similarity measures, our proposed sentiment and continuous vector space measures often rank among the top ones, together with the TRComparer.

MULTSUM, our novel summarization method, based on submodular optimization, multiplies several sentence similarity measures, to be able to make summaries that are good with regards to several aspects at the same time. Our experimental results show significant improvements when using multiplicative combinations of several sentence similarity measures. In particular, the results of MULTSUM surpasses that of the original submodular optimization method.

In our experiments we found that using between two and four similarity measures lead to significant improvements compared to using a single measure. This verifies the validity of commonly used measures like TextRank and LinTFIDF as well as new directions like phrase embeddings and sentiment analysis.

There are several ideas worth pursuing that could further improve our methods. We will explore methods of incorporating more semantic information in our sentence similarity measures. This could come from systems for Information Extraction (Ji et al., 2013), or incorporating external sources such as WordNet, Freebase and DBpedia (Nenkova and McKeown, 2012).

7 Related Work

Ever since (Luhn, 1958), the field of automatic document summarization has attracted a lot of attention, and has been the focus of a steady flow of research. Luhn was concerned with the importance of words and their representativeness for the input text, an idea that’s still central to many current approaches. The development of new techniques for document summarization has since taken many different paths. Some approaches concentrate on what words should appear in summaries, some focus on sentences in part or in whole, and some consider more abstract concepts.

In the 1990’s we witnessed the dawn of the data explosion known as the world wide web, and research on multi document summarization took off. Some ten years later, the Document Understanding Conferences (DUC) started providing researchers with datasets and spurred interest with a venue for competition.

Luhn’s idea of a frequency threshold measure for selecting topic words in a document has lived on. It was later superseded by $tf \times idf$, which measures the specificity of a word to a document,
The two bombers who carried out Friday’s attack, which led the Israeli Cabinet to suspend deliberations on the land-for-security accord signed with the Palestinians last month, were identified as members of Islamic Holy War from West Bank villages under Israeli security control. The radical group Islamic Jihad claimed responsibility Saturday for the market bombing and vowed more attacks to try to block the new peace accord. Israel radio said the 18-member Cabinet debate on the Wye River accord would resume only after Yasser Arafat’s Palestinian Authority fulfilled all of its commitments under the agreement, including arresting Islamic militants.

Table 5: Example output from MULTSUM. Input document: d30010t from DUC 04. Similarity Measures: W2V, TRComparer, and FilteredWordComparer.

something that has been used extensively in document summarization efforts. RegSum (Hong and Nenkova, 2014) trained a classifier on what kinds of words that human experts include in summaries. (Lin and Bilmes, 2011) represented sentences as a \( tf \times idf \) weighted bag-of-words vector, defined a sentence graph with weights according to cosine similarity, and used submodular optimization to decide on sentences for a summary that is both representative and diverse.

Several other methods use similar sentence-based formulations but with different sentence similarities and summarization objectives (Radev et al., 2004; Mihalcea and Tarau, 2004).

(Bonzanini et al., 2013) introduced an iterative sentence removal procedure that proved good in summarizing short online user reviews. CLASSY04 (Conroy et al., 2004) was the best system in the official DUC 04 evaluation. After some linguistic preprocessing, it uses a Hidden Markov Model for sentence selection where the decision on inclusion of a sentence depends on its number of signature tokens. The following systems have also showed state-of-the-art results on the same data set. ICSI (Gillick et al., 2008) posed the summarization problem as a global integer linear program (ILP) maximizing the summary’s coverage of key n-grams. OCCAMS_V (Davis et al., 2012) uses latent semantic analysis to determine the importance of words before the sentence selection. (Kulesza and Taskar, 2012) presents the use of Determinantal point processes (DPPs) for summarization, a probabilistic formulation that allows for a balance between diversity and coverage. An extensive description and comparison of these state-of-the-art systems can be found in (Hong et al., 2014), along with a repository of summary outputs on DUC 04.

Besides the aforementioned work, interested readers are referred to an extensive survey (Nenkova and McKeown, 2012). In particular, they discuss different approaches to sentence representation, scoring and summary selection and their effects on the performance of a summarization system.

8 Conclusions

We have demonstrated that extractive summarization benefits from using several sentence similarity measures at the same time. The proposed system, MULTSUM works by using standard kernel techniques to combine the similarities. Our experimental evaluation shows that the summaries produced by MULTSUM outperforms state-of-the-art systems on standard benchmark datasets. In particular, it beats the original submodular optimization approach on all three variants of ROUGE scores. It attains state-of-the-art results on both ROUGE-2 and ROUGE-SU4, showing that the resulting summaries have high fluency. The results are statistically significant and consistent over all three tested datasets: DUC 02, DUC 04, and Opinion.

We have also seen that sentence similarity measures based on sentiment analysis and continuous vector space representations can improve the results of multi-document summarization. In our experiments, these sentence similarity measures used separately are not enough to create a good summary, but when combining them with traditional sentence similarity measures, we improve on previous methods.

Acknowledgments

This work has been done within "Data-driven secure business intelligence”, grant IIS11-0089 from the Swedish Foundation for Strategic Research (SSF). We would like to thank Gabriele Capannini for discussion and help and Jonatan Bengtsson for his work on sentence similarity measures.
References


Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In Proceedings of COLING, pages 340–348. ACL.


Kai Hong, John M Conroy, Benoit Favre, Alex Kulesza, Hui Lin, and Ani Nenkova. 2014. A repository of state of the art and competitive baseline summaries for generic news summarization. LREC.


