UNIVERSITÀ DEGLI STUDI
DI TRENTO
CIMeC - Center for Mind/Brain Sciences

Capturing Discriminative Attributes in a Distributional Space

rijksuniversiteit
groningen

Author:
Alicia KREBS

Supervisors:
Denis PAPERNO
Malvina NISSIM

A thesis submitted in fulfilment of the requirements
for the Master's Degree in Cognitive Science (UNITN)
and Master's Degree in Linguistics (RUG)

2016

UNIVERSITY OF TRENTO

UNIVERSITY OF GRONINGEN

Abstract

Master's Degree in Cognitive Science (UNITN)

Master's degree in Linguistics (RUG)

Capturing Discriminative Attributes in a Distributional Space

by Alicia KREBS

Lexical similarity is not enough to assess the performance of word vector representations: new ways of evaluating semantic distributional models are needed.

We created a new task specifically tailored to evaluate semantic models on aspects that similarity or relatedness evaluation cannot assess. Contrary to similarity, the nature of semantic difference has yet to be extensively explored in the domain of distributional models. The goal of the task is to assess whether a distributional model can be used to capture the difference between the meaning of two words.

A dataset of discriminative attributes was created to try answering that question in a supervised machine learning setting, using both regression and classification based approaches. Experimental results show that more robust approaches are needed to capture semantic differences.

The discriminative attributes task is important not only to achieve a better understanding of the validity of our assessment of distributional models, but also to improve practical applications. Distributional models have been found to be useful in a variety of computational linguistics applications, such as sentiment analysis, information retrieval, or lexical ambiguity resolution. Solving the discriminative attributes task could largely improve the quality of automated lexicography approaches, machine translation software, and conversational agents.

Acknowledgements

First and foremost, I would like to thank my supervisors, Denis Paperno and Malvina Nissim, for their invaluable guidance and encouragement.

I would also like to thank Marco Baroni, Roberto Zamparelli, Aurélie Herbelot, Germán Kruszewski and Angeliki Lazaridou for providing insightful comments during the early stages of this work and helping steer it in the right direction.

My sincere thanks go to Gosse Bouma and Raffaella Bernardi, who helped me make the most of my time at the University of Groningen and the University of Trento.

Finally, I would like to thank Bobbye Pernice and Ivana Kruijff-Korbayova for the tremendous work they've put into the Erasmus Mundus Language and Communication Technologies program.

Contents

List of Figures

List of Tables

Glossary

Attribute vs Feature "Attribute" is used to refer to the semantic property of an object, while "feature" is used to refer to an independent variable in a machine learning model.

Difference vs Offset "Difference" is used to refer to semantic difference, while "offset" is used to refer to mathematical difference.

Item vs Example "Item" and "example" are both used interchangeably to refer to a data point in a machine learning model.

Chapter 1

Introduction

1.1 The distributional hypothesis

Any visitor walking into a library will find that every work of fiction has congregated into the same section, that dictionaries are sharing an entire rack with each other, and that history books are stacked back to back on the same shelf. Classification schemes have emerged because organizing books in a rational manner is the only way of retrieving them in a timely fashion. Similar books are stored next to each other not because they like the company of their own, but because it makes it easier for us to find them.

It also means that the contents of any book can be deduced by the contents of the books surrounding it.

Consider the following:

1. The haab can be thought of as another large wheel.
2. The haab cycle continues indefinitely, repeating every 365 days.
3. In texts, the haab date is represented by a number prefixed to a morphograph of the name of the month.

Rogers, 2004, p. 241-242.

Despite not knowing what the *haab* is, the contexts in which that word occurs are enough to give the reader a mental image of what it might be. Words are similar to the books that are found in libraries: the meaning of any word can be deduced from the meaning of the words surrounding it.

While that may not appear to be the most straightforward way a human could learn the meaning of a new word, it is an easy way of grant linguistic knowledge to a machine.

The distributional hypothesis states that words that occur in similar contexts carry similar meanings. The idea was popularized by Firth (1957), who phrased it as "You shall know a word by the company it keeps!". This view

had earlier been formalized in the works of Harris (1954): "If A and B have almost identical environments. . . we say that they are synonyms."

Semantic similarity being another way of expressing the idea of meaning differences, it can be said the differential view on meaning is itself rooted in earlier structuralist works (Sahlgren, 2008). The idea that meaning emerges from differences can be found in the work of Saussure:

> "In language there are only differences. [...] Whether we take the signified or the signifier, language has neither ideas nor sounds that existed before the linguistic system, but only conceptual and phonic differences that have issued from the system. [...] A linguistic system is a series of differences of sound combined with a series of differences of ideas; but the pairing of a certain number of acoustical signs with as many cuts made from the mass thought engenders a system of values."
>
> *Saussure, 1916 [ed. 1989], p. 121*

Variations in distributional contexts are thus essential to the construction of semantic meaning.

1.2 The discriminative attributes task

Similarity tasks have become the standard in the evaluation of distributional models. Similarity data sets can be produced in different forms: ratings of pairs, sorting of objects, or correlation between occurrences. For example, a popular data set used in similarity tasks is the MEN test collection: the data set contains 3000 word pairs rated by human similarity judgements (Bruni, Tran, and Baroni, 2014). The pairs were randomly selected from a set of frequent words. Each pair was rated by a human annotator as being more or less related than another pair, and obtained a final score on a 50-point scale. Using an approach that combines visual and textual features, Bruni, Tran, and Baroni (2014) report an accuracy of 78% on this data set.

These tasks rely on geometric models in which objects are represented as points in a coordinate space, so that similar objects are closer to each other than are to dissimilar objects. The distance between two points in a coordinate space is assumed to follow three axioms:

1. **Minimality:** The distance between two objects is equal to or greater than the distance between an object and itself, which is zero.

$$\delta(A, B) >= \delta(A, A) = 0$$

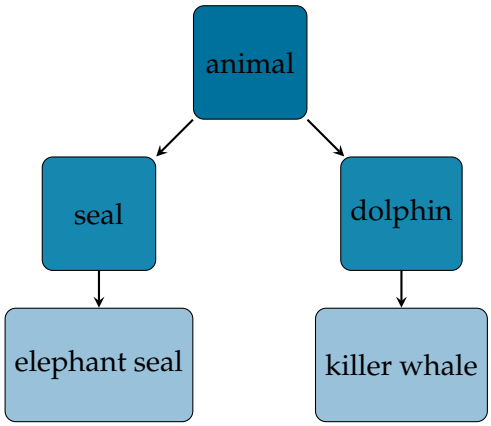
2. **Symmetry:** The distance of A to B is the same as the distance of B to A.

$$\delta(A, B) = \delta(B, A)$$

3. **Triangle inequality:** The sum of the distance between between A and B and the distance between B and C cannot be less than the distance between A and C.

$$\delta(A, B) + \delta(B, C) >= \delta(A, C)$$

Tversky (1977) has argued that these axioms do not hold for similarity measures: the minimality requirement is not satisfied since identical stimuli are not consistently judged as being the same or different, and empirical evidence suggests that similarity judgements are not symmetric. It was shown that the referent of a similarity statement tends to be more salient stimulus, which is why we say "an ellipse is like a circle" and not "a circle is like an ellipse" - an ellipse is more similar to a circle than a circle is to an ellipse. (Tversky, 1977, p. 8). Since the referent is always more salient than the subject, the subject often is a hypernym of the referent.



An elephant seal is similar to a seal. $\neq$ A seal is similar to an elephant seal.
A killer whale is similar to a dolphin. $\neq$ A dolphin is similar to a killer whale.
A seal is similar to an animal. $\neq$ An animal is similar to a seal.
A dolphin is similar to an animal. $\neq$ An animal is similar to a dolphin.

FIGURE 1.1: Similarity is an asymmetric relationship.

The triangle inequality has been put into question as well: while Jamaica is similar to Cuba (geographical proximity) and Cuba is similar to Russia (political proximity), Jamaica and Russia are not similar (Tversky, 1977, p. 9). In other words, the distance between Jamaica and Russia does not appear to be smaller than the sum of the distance between Jamaica and Cuba and

the distance between Cuba and Russia, which breaks the triangle inequality axiom.

While similarity tasks have become the standard in the evaluation of distributional models, the validity of those tasks has been put into question: inter-annotator agreement tends to be low, the small size of some of the most popular datasets is a concern, and subjective similarity scores have limitations when it comes to task-specific applications (Faruqui et al., 2016; Batchkarov et al., 2016).

The nature of semantic *difference* between two related words can vary greatly. Modelling difference can help capture individual aspects of meaning. While similarity may be too simple to assess the validity of semantic representations, modelling difference can help capture more subtle aspects of meaning and improve the overall performance of computational models.

Tversky (1977) proposes that similarity should be described as a comparison of features, rather than as a comparison of metric distances. For the domain of objects $\Delta = \{a, b, c, ..., n\}$, each object a,b and c is associated with a set of features A, B and C. Features may be binary (e.g. body part), nominal (e.g. color) or ordinal (e.g. weight). Similarity can then be evaluated by counting how many features two objects have in common.

The goal of the discriminative attributes task is to capture differences between concepts using the set of their features. Given two words a and b, a model should be able to predict which attributes characterize their difference in meaning the best.

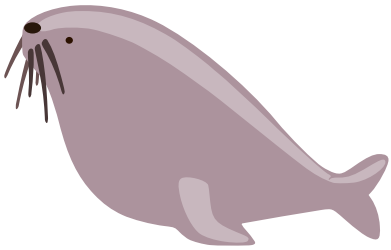


FIGURE 1.2:
a = seal;
{A} = features of a

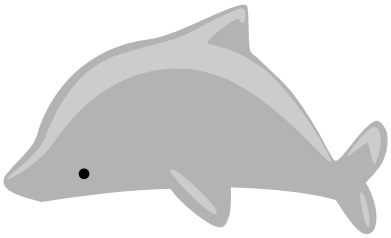


FIGURE 1.3:
b = dolphin;
{B} = features of b

FIGURE 1.4: If $whiskers$ is a discriminative attribute of the word pair $seal$ and $dolphin$, then $whiskers \in A \backslash B$.

Since this task does not rely on subjective scores, it can easily be solved by humans. While a pair is rarely given the same similarity score by two separate annotators, inter-annotator agreement is expected to be high when asked to judge if a given word is a discriminative attributes between two concepts. When annotating a test set for the discriminative task (see Section 3.3), annotators agreed on 72.4% of items on average.

This task is related to previous work that attempts to predict the discriminative features of referents, using natural images to represent the input objects (Lazaridou, Pham, and Baroni, 2016). Attributes have also been used to simulate similarity judgements and concept categorization (Silberer and Lapata, 2014). On a more abstract level, this task is related to previous attempts at using offset vectors to capture lexical relations without explicit supervision (Mikolov, Yih, and Zweig, 2013), which have been shown to be able to generalise well to a range of relations (Vylomova et al., 2015).

Solving the discriminative attributes task could help in a number of applications: attributes to include in dictionary definitions could be generated automatically, taking into account semantic differences could greatly improve the quality of the machine translation and sentiment analysis software, and choosing lexical items with contextually relevant differential features could help conversational agents have more pragmatically appropriate dialogues.

Chapter 2

Distributional Space Models

2.1 Counts

Computing the meaning of a word from the distribution of words around it is a data-driven method that allows to collect word meanings on a large scale (Baroni, Bernardi, and Zamparelli, 2014). Representations of word meanings use distributional vectors to represent words. A word can be represented as a high-dimensional sparse vector through the occurrence counts of every context c of a word w. Thus, the number of dimensions of such a vector corresponds to the number of contexts c of the word w.

	apple	melon
red	2	3
green	3	6

TABLE 2.1: A Distributional Space Model.

Two distributional models will be used: a count-based model, and a predictive model. The count-based model uses the cooccurrence counts of the best count-based model from Baroni et al. (Baroni, Dinu, and Kruszewski, 2014). These counts were collected using the concatenation of Wikipedia, the web-crawled ukWack corpus (Baroni et al., 2009), and the BNC (British National Corpus, 2001). This corpus contains a total of 2.8 billion tokens.

2.2 Weighting

2.2.1 Association metrics

Raw co-occurrence is not very informative by itself, as it is skewed towards very frequent words. For example, frequent words such as *the*, *it*, or *they* cannot help us differentiate *pineapple* from *apricot* (Martin and Jurafsky, 2016).

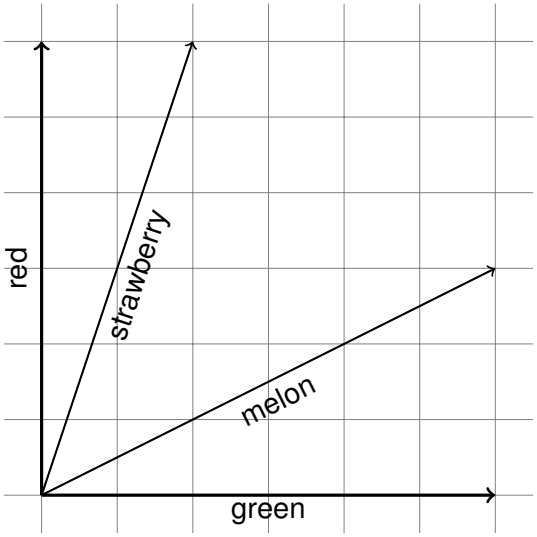


FIGURE 2.1: A two-dimensional distributional space model.

Association metrics are used to determine the degree of statistical association between words.

Church and Hanks (1990) used the concept of mutual information, which is used to compare the probabilities of two events, to compute an association ratio. This association ratio became known as pointwise mutual information (PMI). The PMI value of a word-context pair (w, c) is the log of the probability of w occurring in the context of c divided by its expected value.

$$PMI(w, c) = \log \frac{P(w, c)}{P(w)P(c)}$$

The PMI value can be either negative or positive. A high PMI value means that the word and its context are highly correlated. When computing positive pointwise mutual information (PPMI), every negative value of PMI is set to zero. This allows us to only look at positive associations, which are less prone to noise and more immediately interpretable.

$$PPMI(w, c) = \max(\log \frac{P(w, c)}{P(w)P(c)}, 0)$$

2.2.2 Hyperparameters

Smoothing

Large data structures tend to contain both meaningful patterns and noise: smoothing techniques can be implemented to remove such noise. Mikolov et al. (2013a) have shown that smoothing the original contexts distribution

by raising unigram frequencies to the power of alpha improves the performance of predictive models. This same smoothing technique was used by Levy, Goldberg, and Dagan (2015) in conjunction with PMI to improve the performance of count models. Previous research has shown that .95 is an optimal smoothing parameter (Krebs and Paperno, 2016).

$$PMI(w,c) = \log \frac{\hat{P}(w,c)}{\hat{P}(w) \cdot \hat{P}_\alpha(c)}$$

$$\hat{P}_\alpha(c) = \frac{\#(c)^\alpha}{\sum_c \#(c)^\alpha}$$

CDS was implemented in the count-based model by raising every count to the power of .95.

Shifting

Levy and Goldberg (2014) introduced Shifted Positive Pointwise Mutual Information (SPPMI) as an association measure more efficient than PPMI. For every word w and every context c, the SPPMI of w is the higher value between 0 and its PMI value minus the log of a constant k. SPPMI helps remove low PPMI values, the majority of which are statistical noise. A value of 5 for k is recommended (Krebs and Paperno, 2016).

$$PPMI(w,c) = \max(\log \frac{P(w,c)}{P(w)P(c)}, 0)$$

$$SPPMI_k(w,c) = \max(PMI(w,c) - \log k, 0)$$

The count-based model was weighted using SPPMI ($k = 5$).

Subsampling

Subsampling consists of taking a subsample (a number of words) from a larger sample (a corpus of words). It was used by Mikolov et al. (2013b) to remove frequent words from a corpus, as they carry less meaningful information than rare words. Each word in the corpus with frequency above threshold t can be ignored with probability p, computed for each word using its frequency f:

$$p = 1 - \sqrt{\frac{t}{f}}$$

Following Mikolov et al., I used $t = 10^{-5}$. In word2vec, subsampling is applied before the corpus is processed. Levy, Goldberg, and Dagan (2015)

explored the possibility of applying subsampling afterwards, which does not affect the context window's size, but found no significant difference between the two methods.

2.3 Sparse and dense vectors

The vectors in the weighted count-based model are sparse: most of the elements of the matrix are equal to zero. Dense vectors are preferable when building machine learning systems, as they contain fewer parameters, which leads to better generalization and helps avoid overfitting (Martin and Jurafsky, 2016, Chapter 16).

2.3.1 SVD

Several methods can be used to generate dense vectors from sparse vectors. Singular Value Decomposition (SVD) finds the dimensions along which the data varies the most. A matrix A can be broken into the product of three matrices: an orthogonal matrix U, a diagonal matrix S, and the transpose of an orthogonal matrix V.

$$A_{mn} = U_{mm} \cdot S_{mn} \cdot V^T_{nn}$$

Under this configuration, contexts are represented as row vectors in V and words are represented as row vectors in U. Using words as context dimensions instead of documents was first proposed by Schütze (1992). Similarity between words can be measured by computing the strength of association between two rows of U (see section 2.4.1).

2.3.2 NMF

Non-negative Matrix Factorization (NMF) was introduced by Lee and Seung (1999). Given a non-negative input matrix A, NMF will find two matrices P and Q such that their product approximates A:

$$A \approx P \cdot Q^T = \hat{A}$$

Each row of P and Q represent the strength of association between a word and its context. P and Q are computed by minimizing the squared error:

$$e_{ij}^2 = (a_{ij} - \hat{a}_{ij})^2$$

2.3.3 CBOW

The values of the vectors can also be learned as weights using neural networks. Using this approach, weights are set directly to predict the contexts in which words tend to appear. Mikolov et al. have introduced the Continuous Bag-of-Word (CBOW) architecture for this purpose. With CBOW, the vectors of context words are given as input, and used to predict the current word. The error is then backpropagated and the weights of the context vectors are updated accordingly.

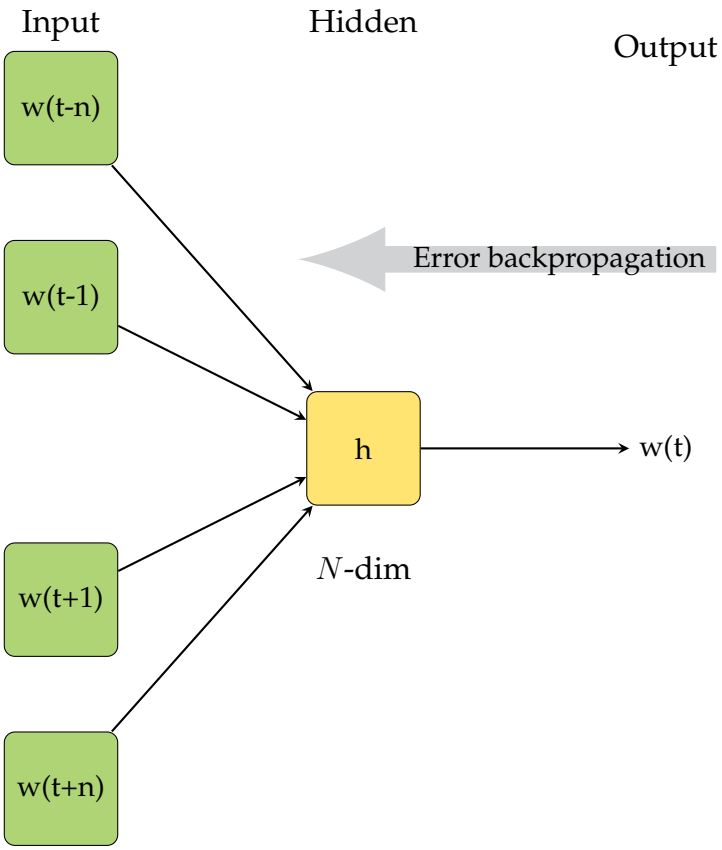


FIGURE 2.2: The CBOW architecture: the context words of the output t are given as input.

The predict model that will be used in this work is the best predict model from Baroni et al. (Baroni, Dinu, and Kruszewski, 2014), which uses the same corpus as the count model. It was trained at 400 dimensions using the CBOW approach, a 5 words window, 10 negative samples, and subsampling.

2.4 Vector operations

Vector operations that will be used in this work are introduced here.

2.4.1 Cosine Similarity

The most common way of computing the similarity between two vectors is to measure their cosine. The cosine is calculated using the inner product of two word vectors $\vec{a}$ and $\vec{b}$:

$$\vec{a} \cdot \vec{b} = \sum_{i=1}^{N} a_i b_i$$

The inner product acts as a similarity metric between word vectors because it tends to be high when two vectors have large values in the same dimensions, while two vectors that have low values in different dimensions will have a low inner product. However, simply using the inner product will systematically give higher similarity scores to words that have longer vectors. To account for word frequency, the inner product can be normalized by dividing it by the length of each vector, which is the cosine of the angle of the two vectors:

$$cosine(\theta) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}$$

The cosine similarity between $\vec{a}$ and $\vec{b}$ can be computed as:

$$CosSim(\vec{a}, \vec{b}) = \frac{\sum_{i=1}^{N} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \sqrt{\sum_{i=1}^{n} b_i^2}}$$

If the semantic space only contains positive values (raw frequencies or PPMI weighted space), the cosine values will range from 1 (the two vectors are identical) to 0 (the two vectors are orthogonal, as they share no positive dimensions).

2.4.2 Vector offsets

Mikolov, Yih, and Zweig (2013) have introduced vector offsets to solve analogy tasks. Word embeddings capture relationships which can be learned as vector offsets, so that $king - man + woman$ results in a vector very similar to $queen$. The vector $queen$ can then be found with cosine similarity. On a

semantic generalization task, vector offsets have been found to outperform previous state-of-the-art methods.

FIGURE 2.3: The vector offset method.

Chapter 3

Dataset

A model can only learn how to solve the discriminative attributes task if it has data to learn from. A dataset is a set of example inputs and their desired outputs: the goal of the algorithm is to find which function can map the inputs to the outputs. This chapter outlines the steps that were taken to create such a dataset for the discriminative attributes task.

3.1 Small data set

A random sample of seed words from the BLESS dataset (Baroni and Lenci, 2011) along with their semantic neighbours was used to create 131 word pairs that were similar and denoted concrete objects. For each word pair, one or more pair(s) of discriminating attributes were assigned manually. For example, the word pair `[scooter, moped]` had two pairs of attributes: `[big, small]` and `[fast, slow]`. The attributes were also annotated by type (e.g., *speed*, *color*...). This dataset could be used to predict which pair of attributes differentiates which pair of words. A simpler task would be to predict the direction of the effect: does $attribute_1$ belong to $word_1$ or $word_2$? To get a simple baseline on direction detection, a similarity score was computed for each item using the following formula:

$$Score = (CosSim(word_1, att_1) \cdot CosSim(word_2, att_2)) \\ -(CosSim(word_1, att_2) \cdot CosSim(word_2, att_1))$$

Out of 86 items, 58 had positive scores. In other words, cosine similarity can detect which attribute describes which word in 67.4% of cases. Looking at the type of attributes that were positively scored indicated that some types of differences could be easier to predict than others. For example, attributes of the type $color$ had positive scores 19 times out 26, while attributes of the type $size$ had positive scores 11 times out of 21.

The difference between color and size is that color is an absolute attribute, while size is a relative one: while a truck can be red independently of the contribution of any other element in the discourse, it can only be called big by comparison to something else.

Relative and absolute adjectives also differ in terms of set intersection: while the set of red trucks is equal to the intersection of the set of trucks and the set of red objects, we cannot describe the set of big trucks as the intersection of the set of trucks and the set of big objects. All members of the set of red objects share the property of being red, but the members of a set of big objects do not share a common property: a red apple is as red as a red truck, but a big apple is not as big as a big truck. It is thus unsurprising that relative attributes cannot easily be used to extract meaning from a distributional model, as the meaning of those attributes can only emerge from discourse.

In a second version of the small dataset, only absolute attributes were included. Some word pairs were then manually added to further exemplify specific differences, such as `[horse, foal]` for the age type. This time, 67 out of 91 items had positive scores. The most successful types of attributes were $color$ (41 out of 51), age (7 out of 9) and $diet$ (4 out of 5).

$type$	w_1	w_2	a_1	a_2
color	tomato	spinach	red	green
color	banana	carrot	yellow	orange
color	tiger	panther	orange	black
age	cat	kitten	old	young
age	dog	pup	old	young
age	horse	foal	old	young
diet	deer	fox	herbivorous	carnivorous
diet	cow	lion	herbivorous	carnivorous
sex	pig	sow	male	female
sex	tiger	tigress	male	female

TABLE 3.1: Examples of positive absolute attributes.

This dataset was too small for training supervised models. Using logistic regression on pairwise cosines with cross-validation showed negligibly low results. This small experiment suggests that it is possible to use absolute attributes to model differences between words. In the future, the small dataset could be used to test models that weren't trained on it in an analogy task setting.

3.2 Feature norms

Only some differences can be expressed as the opposition of two attributes. Other differences are better expressed as the presence or absence of an attribute. As shown in Figure 1.3, one of the differences between a seal and a dolphin is the presence of whiskers: there is no attribute that can express that a dolphin is "whiskerless". This second dataset was thus created to address this issue.

The set of feature norms collected by McRae et al.(McRae et al., 2005) was used to extract discriminative attributes. This set includes attributes for 541 concepts (living and non-living entities). They were collected by asking 725 participants to produce attributes they found important for each concept. Production frequencies of these attributes indicate how salient they are. Attributes are able to encode semantic knowledge because they tap into the representations that the participants have acquired through repeated exposure to those concepts. McRae et al. divided disjunctive attributes, so that if a participant produced the attribute `is_green_or_red` the concept will be associated with both the attribute `is_green` and the attribute `is_red`. Concepts that have different meanings had been disambiguated before being shown to participants. For example, there are two entries for *bow*, `bow_(weapon)` and `bow_(ribbon)`. Because the word vector for *bow* encodes the properties of both senses, words that have multiple senses did not get multiple entries in the discriminative attribute dataset, so that the concept `bow` has the attributes of both the weapon and the ribbon.

The McRae dataset uses the brain region taxonomy (Cree and McRae, 2003) to classify attributes into different types, such as *function*, *sound* or *taxonomic*. I decided to only work with visual attributes, which exist for all concrete concepts, while attributes such as *sound* or *taste* are only relevant for some concepts. Indeed, while it is easy to judge if a concrete object is red, it would be much harder to tell whether an apple is loud.

The brain region taxonomy classification distinguishes between three types of visual attributes: *motion*, *color* and *form and surface*. All three types of attributes were used. Words had to have at least one visual attribute of any type to be selected for inclusion. Then, word pairs were created by selecting the closest neighbours of each of these words. While creating word pairs randomly would have created more items, creating word pairs from words that are similar makes the task less trivial because meaning differences between similar words are more subtle.

Several parameters were explored to select those neighbours: either selecting the first 50, 75 or 100 closest neighbour in the McRae dataset, or selecting

the first 50 or 75 closest neighbours in the WUB distributional space. If the 50 closest neighbours of a word w are selected in the McRae dataset, there will be exactly 50 pairs that start with w in the discriminative attributes dataset. On the other hand, if the 50 closest neighbours of a word w are selected in the distributional space, only the neighbours that are also present in the McRae dataset are kept. If neighbours that aren't present in the McRae dataset were kept, there would be no attributes to describe them, and thus no discriminative attributes could be added. The method that selects neighbours directly from the McRae dataset generates more items, but the method that selects neighbours directly from the distributional space generates word pairs which have higher similarity scores.

For each word pair, if there was an attribute that one word had and the other didn't, that word pair and attribute item was added to the dataset. The set was built in such a way that the attribute of each item always refers to an attribute of the first word. For example, in Table 3.2, *wings* is an attribute of *airplane*. The word pair `[airplane,helicopter]` will only be included in the order `[helicopter, airplane]` if *helicopter* has a attribute that *airplane* doesn't have. The relations are thus asymmetric and have fixed directionality. For simplicity, multi-word attributes were processed so that only the final word is taken into account (e.g. `has_wings` becomes `wings`). Some attributes were excluded from the dataset because they did not contain any meaningful information when processed as a single word. The excluded attributes are the following:

anything	outside
elements	shapes
ends	sides
end	surface
inside	top
it	up
level	

The dataset was then supplemented with negative examples. Two types of negative examples were added: examples where the attribute is a similarity between $word_1$ and $word_2$, and examples where the attribute is neither an attribute of $word_1$ nor $word_2$.

$word_1$	$word_2$	$attribute$
airplane	helicopter	wings
bagpipe	accordion	pipes
canoe	sailboat	fibreglass
dolphin	seal	fins
gorilla	crocodile	bananas
oak	pine	leaves
octopus	lobster	tentacles
pajamas	necklace	silk
skirt	jacket	pleats
subway	train	dirty

TABLE 3.2: Examples of word pairs and their discriminative attribute.

	50	75	100
WUB	3694	5097	6325
RAE	25074	36673	-

TABLE 3.3: Number of items under different parameters.

3.3 Training and testing

The dataset was split into a training and a test set. In order to minimize lexical overlap, whenever a word pair was used to populate the test set, all of the samples containing that pair were added to the test set. The order of the items was first randomized, and the procedure stopped when 2000 items were added to the test set.

In other words, none of the items in the training set start with a word with which an item of the testing set starts. This ensures that the models do not learn which attributes belong to which word, but instead learn what is and what isn't a discriminative attributes given a pair of words.

If lexical overlap isn't minimized, good results could have be achieved if the model simply learned the attributes of each word: if $whiskers$ is always predicted when the first word is $seal$, the model will be correct most of the time because there are more animals that do not have whiskers than animals that do. However, if items that start with $seal$ are only present in the test set, the model cannot learn that narwhals have horns.

Early experiments showed that attribute memorization led to misleading results. Table 3.4 shows results for the linear regression task (see section 4.2.1) when lexical overlap is not minimized: the high accuracy of the model when it comes to closest neighbour accuracy is not representative of its inability to generalize.

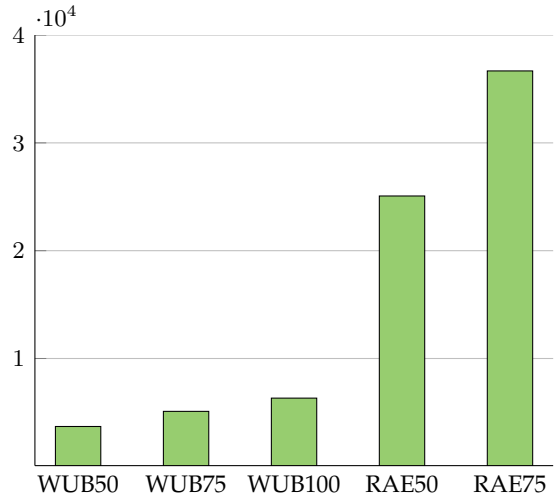


FIGURE 3.1: Number of items in the dataset under different parameters.

TABLE 3.4: Linear regression with lexical overlap (SVD).

	WUB(50)		**RAE(50)**	
	R^2	*CN Accuracy*	R^2	*CN Accuracy*
concatenate	0.1675	0.88	0.7726	0.95
subtract	0.2222	0.57	0.4471	0.73

The test set created under the WUB(100) parameter was then checked manually by two annotators to ensure its quality. Items were only kept when an agreement was reached, resulting in a test set of 904 positive examples and 1272 negative examples.

For the positive examples, 54.8% of items were discarded. Annotators agreed to keep 45.2% of items, agreed to discard 33% of items, and disagreed on 21.8% of items. For the negative examples, 12.5% of items were discarded. Annotators agreed to keep 87.5% of items, agreed to discard 0.8% of items, and disagreed on 11.6% of items. The items that both annotators agreed to discard from the positive examples were added to the negative examples.

Table 3.5 shows the number of positive and negative examples in the test set, table 4.18 shows the number of positive and negative examples in the training set.

positive	904
negative	1272

TABLE 3.5: Number of positive and negative items in the manually checked test set.

Chapter 4

Results

Several experiments were performed to try solving the discriminative attributes task. This chapter covers the different ways in which the discriminative attributes dataset can be used. The task can have three different goals:

1. **Predicting discrete variables**: Given the vector representation for $word_1$ and $word_2$, predict the discriminative attribute x, or the set of discriminative attributes $[x_1, x_2 \dots x_n]$.
2. **Predicting continuous variables**: Given the vector representation for $word_1$ and $word_2$, predict the vector representation of a discriminative attribute x.
3. **Predicting dichotomous variables**: Given the vector representation for $word_1$, $word_2$ and $attribute_x$, predict whether or not $attribute_x$ is a discriminative attribute.

The performance of the models described in this chapter depends on four variables:

- The quantity and quality of the items being predicted, as well as their distribution in the training and testing set.
- The quality of the word embeddings being used to make those predictions.
- The suitability of the learning algorithms that are used to solve the task.
- The difficulty of the task itself.

4.1 Predicting discrete variables

When predicting discrete variables, the goal of the task is to predict a nominal class which is represented as a written word. As an example, for the triple `[seal,dolphin,whiskers]`, the input will be the dimensions of the vector representations of *seal* and *dolphin* (which may be variables of any type), and the output will be the word "whiskers".

4.1.1 Logistic regression

Logistic regression is a regression model which was first developed by Cox (1958). Logistic regression measures the relationship between a categorical dependent variable and one or more independent variables. The outcome of the model is a probability P which ranges from 0 to 1 and indicates the likelihood that an input point belongs to a certain class.

Logistic regression is based on the logistic function, which is defined as:

$$\sigma(y) = \frac{1}{1 + e^{-y}}$$

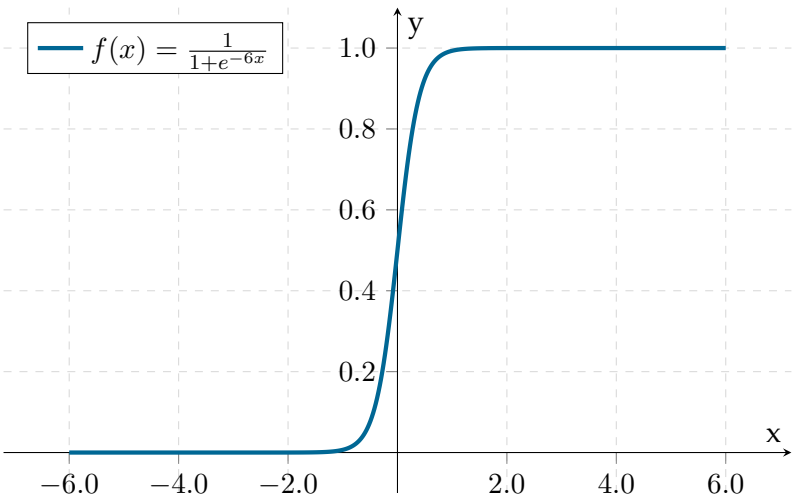


FIGURE 4.1: Logistic function on the interval (-6,6).

Given two independent variables X_1 and X_2, the input point Y can be expressed in the following terms:

$$Y = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2$$

Where β_0 is the coefficient of the intercept, β_1 is the coefficient of the independent variable X_1, and β_2 is the coefficient of the independent variable X_2.

In a binary setup, the probability that the input point Y belongs to a class k would then be the following:

$$P(Y = k|X_1, X_2) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2)}}$$

In a multi-class scenario, the One-versus-Rest strategy can be used. For every class k, a binary classifier is trained using all samples of that class as positive examples and all other samples as negative examples. When all classifiers have been run, a class label is applied to each input point depending on which classifier reports the highest confidence score.

Logistic regression was implemented using scikit-learn (Pedregosa et al., 2011).

4.1.2 Multi-class, Single-label

The multi-class, single-label task is a classification task where there can only be one class (e.g., discriminative attribute) per word-neighbour pair. The number of classes corresponds to the total number of attributes. The task is multi-class because the classifier must choose between more than two classes when assigning class labels. But it is a single-label task because a sample can only belong to one class.

In the full version of the discriminative attributes dataset, each word pair has more than one difference - and each one of those differences could be chosen as the class to be predicted. In this set-up, one discriminative attribute has to be selected, and all other discriminative attributes must be discarded. To select the most salient difference, the attribute that had the highest production frequency was chosen. The assumption is that if $attribute_k$ is strongly associated with $word_a$ but never produced with $word_b$, it is the feature that speakers would produce if asked what the most striking difference between $word_a$ and $word_b$ is.

For example, the word pair `[seal, dolphin]` might have two discriminative attributes, *whiskers* and *flippers*. For this task, if *whiskers* has a higher production frequency than *flippers*, only the triple `[seal, dolphin, whiskers]` would be included in the dataset.

The features of the model can be the concatenation of the vectors for $word_1$ and $word_2$, or the offset of the vector for $word_1$ and the vector for $word_2$.

Table 4.1 shows the size of the datasets used in the multi-class task under different parameters. For this task, the datasets were split randomly so that the test set size was 33% of the whole dataset size.

TABLE 4.1: Size of datasets for the Multi-class, Single-label task.

	train	**test**
WUB(50)	2474	1220
WUB(75)	3414	1683
WUB(100)	4237	2088
RAE(50)	16799	8275
RAE(75)	24570	12103

Performance

TABLE 4.2: Multi-class, Single-label - WUB(50)

	WUB(50)					
	concatenate			*subtract*		
	precision	recall	f1	precision	recall	f1
svd	0.94	0.94	0.93	0.89	0.89	0.89
nmf	0.32	0.33	0.28	0.23	0.18	0.15
w2v	0.95	0.96	0.95	0.89	0.89	0.89

TABLE 4.3: Multi-class, Single-label - WUB(75)

	WUB(75)					
	concatenate			*subtract*		
	precision	recall	f1	precision	recall	f1
svd	0.95	0.95	0.95	0.91	0.91	0.91
nmf	0.37	0.38	0.34	0.25	0.23	0.20
w2v	0.96	0.97	0.96	0.92	0.91	0.91

In the multi-class, single-label task, the model achieves a high performance. The word2vec and SVD embeddings achieve similarly high results, but the NMF embeddings achieves significantly lower results.

Figure 4.2 shows how the NMF embeddings underperform compared to other word embeddings. It also shows that for this task adding more data has little effect: for the SVD embeddings, going from a dataset size of 6325 items (WUB(100)) to a dataset size of 25074 items (RAE(50)) only improves the performance of the model by 2%. Using vector offsets is less efficient than concatenating the vectors.

4.1.3 Multi-class, Multi-label

In the multi-class, multi-label scenario, each word-neighbour pair can belong to several classes (e.g., have several differences). For example, if the

TABLE 4.4: Multi-class, Single-label - WUB(100)

	WUB(100)					
	concatenate			*subtract*		
	precision	recall	f1	precision	recall	f1
svd	0.97	0.97	0.97	0.93	0.93	0.93
nmf	0.40	0.40	0.35	0.32	0.26	0.23
w2v	0.97	0.98	0.97	0.93	0.93	0.92

TABLE 4.5: Multi-class, Single-label - RAE(50)

	RAE(50)					
	concatenate			*subtract*		
	precision	recall	f1	precision	recall	f1
svd	0.98	0.99	0.99	0.97	0.97	0.97
nmf	0.71	0.68	0.66	0.59	0.50	0.49
w2v	0.99	0.99	0.99	0.97	0.97	0.97

word pair `[seal, dolphin]` has two discriminative attributes *whiskers* and *flippers*, the model will be expected to predict both *whiskers* and *flippers*.

In Table 4.7, the number of pairs is the number of word-neighbour pairs, and the number of triples is the number of word-neighbour-difference triples. On average, a word pair has 5.8 discriminative attributes in the WUB(50) dataset and 5.1 discriminative attributes in the RAE(50) dataset.

Table 4.8 shows the size of the training and testing sets for the multi-class, multi-label task. For this task, the datasets were split randomly so that the test set size was 33% of the whole dataset size.

Two methods are explored to predict discriminative attributes: concatenating the vectors for $word_1$ and $word_2$, or subtracting the vector for $word_2$ from the vector for $word_1$.

Performance

Tables 4.9 and 4.10 show that performance in the multi-class, multi-label task is high, but slightly lower than in the multi-class, single-label. Once again, using vector offsets is less efficient than concatenating vectors: only when using the RAE(50) dataset, which is much larger, do the vector offsets perform well.

These experiments showed that logistic regression is able to correctly predict discrete variables in the discriminative attributes task. However, in this task

TABLE 4.6: Multi-class, Single-label - RAE(75)

	RAE(75)					
	concatenate			*subtract*		
	precision	recall	f1	precision	recall	f1
svd	0.99	0.99	0.99	0.98	0.98	0.98
nmf	0.82	0.80	0.79	0.68	0.62	0.61
w2v	0.99	0.99	0.99	0.98	0.98	0.98

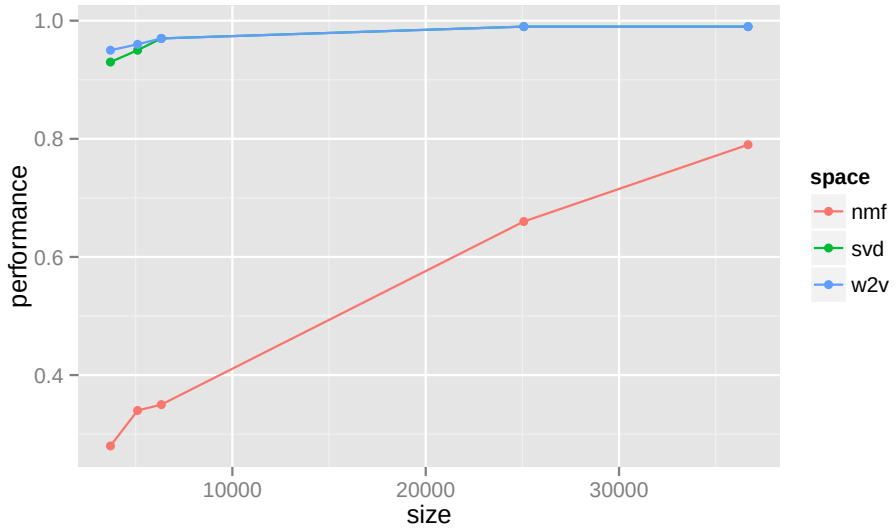


FIGURE 4.2: Performance of different word embeddings as a function of dataset size in the Multi-class, Single-label task.

lexical overlap is not minimized: we suspect that attribute memorization is the reason the models are doing so well.

4.2 Predicting continuous variables

When predicting continuous variables, the goal of the task is to predict a series of numbers. For the triple `[seal,dolphin,whiskers]`, the input may be the concatenation of the vectors for *seal* and *dolphin*, while the output will be a completely new vector representation. To assess the performance of the model, the closest neighbour of that new vector is retrieved from the distributional space using cosine similarity: if the model performs well, we expect that neighbour to be "whiskers".

TABLE 4.7: Number of pairs and triples in the different builds for the Multi-class, Multi-label task.

	WUB(50)	**RAE(50)**
pairs	21666	24967
triples	3717	128515
average	5.8	5.1

TABLE 4.8: Size of datasets for the Multi-class, Single-label task.

	train	**test**
WUB(50)	2490	1227
RAE(50)	16728	8239

4.2.1 Linear regression

The regression equation is used to compute the estimated value $\hat{y}$ of the dependent variable y for every value of the independent variable x:

$$\hat{y} = \beta_0 + \beta_1 \cdot x$$

The Ordinary Least Squares procedure is used to minimize the sum of the squared residuals between the observed responses and the responses predicted by the regression equation.

Given $\bar{x}$, the mean of x, the OLS estimator for the slope is:

$$\beta_1 = \frac{\sum_{i=1}^{n} x_i - \bar{x}}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

Given $\bar{y}$, the mean of y, the OLS estimator for the intercept is:

$$\beta_0 = \bar{y} - \beta_1 \cdot \bar{x}$$

In this task, the model tries to predict a vector V, where every element of V is a dependent variable. Linear regression will be performed in a setting where the columns of y are tested independently, as multiple univariate tests.

The linear regression task can only be performed as a single-label task. The datasets used were the same as those used for the multi-class, single-label logistic regression task, but were split differently. As seen in table 3.4, a high performance can be achieved through attribute memorization. To minimize

TABLE 4.9: Multi-class, Multi-label - WUB(50)

	WUB(50)					
	concatenate			*subtract*		
	precision	recall	f1	precision	recall	f1
svd	0.93	0.92	0.93	0.73	0.72	0.72
nmf	0.82	0.51	0.60	0.73	0.11	0.19
w2v	0.96	0.95	0.96	0.86	0.72	0.77

TABLE 4.10: Multi-class, Multi-label - RAE(50)

	RAE(50)					
	concatenate			*subtract*		
	precision	recall	f1	precision	recall	f1
svd	0.96	0.97	0.96	0.97	0.96	0.97
nmf	0.98	0.83	0.89	0.97	0.53	0.65
w2v	0.96	0.96	0.95	0.95	0.90	0.92

lexical overlap, the datasets were split so that the discriminative attributes that were randomly added to the test set were not included in the training set. For example, if the test set contains `[seal, dolphin, whiskers]`, none of the items in the training set contain the attribute $whiskers$. This was not possible when predicting discrete variables, since the class to be predicted has to already be present in the training test in that setting.

Linear regression was implemented using scikit-learn (Pedregosa et al., 2011).

TABLE 4.11: Size of datasets for the linear regression task.

	train	**test**
WUB(50)	2447	1247
WUB(100)	4155	2170

4.2.2 Performance

The coefficient score R^2 (between -Inf and 1, where 1 is the best and -Inf is the worst) shows how close the predicted vectors are to the expected vectors. It is defined as:

$$R^2 = (1 - U/V)$$

Where U is the regression sum of squares for the predicted vector $\hat{y}$ given the expected vector y:

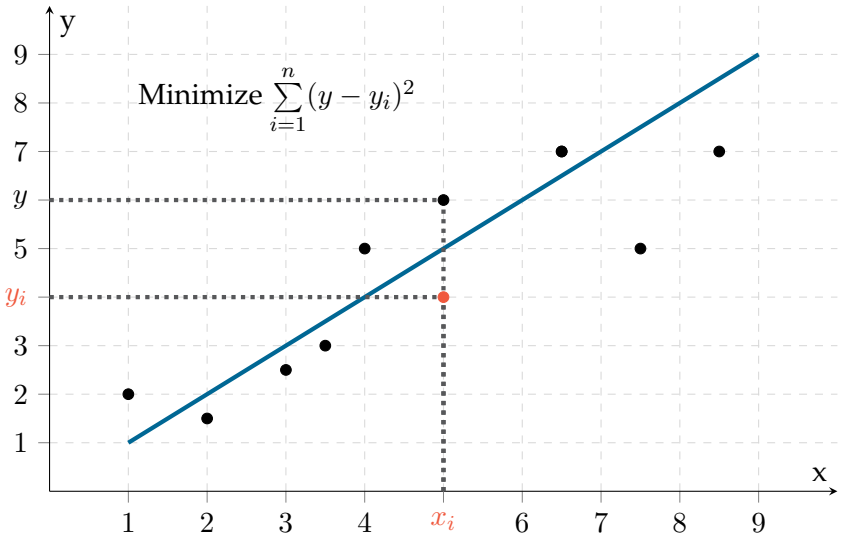


FIGURE 4.3: Linear regression with Ordinary Least Squares

$$U = \sum_{i=1}^{n}(y - \hat{y})^2$$

And V is the residual sum of squares for the expected vector y and its mean $\bar{y}$:

$$V = \sum_{i=1}^{n}(y - \bar{y})^2$$

Additionally, for each predicted vector, its closest neighbour was retrieved from the distributional space. The closest neighbour accuracy is the number of times the extracted closest neighbour was the same as the expected discriminative attribute: in table 4.12, the best model achieves 4.1% accuracy.

TABLE 4.12: Linear Regression (WUB(50), SVD)

	R^2	*CN Accuracy*
concatenate	-1.46E+23	0.0
subtract	-1.0930	0.023
multiply	-0.7626	0.024
add	-0.7859	0.0
multiply ⌒ subtract	-1.2302	0.041
multiply ⌒ direction	-0.4262	0.032

Several methods were explored to combine the vector for $word_1$ and the vector for $word_2$:

1. Concatenating $word_1$ and $word_2$ ($|V| = 800$).
2. Subtracting $word_1$ from $word_1$ ($|V| = 400$).

TABLE 4.13: Linear Regression (WUB(100), SVD)

	R^2	*CN Accuracy*
concatenate	-1.98E+23	0.0
subtract	-1.002	0.0
multiply	-0.46	0.028

3. Multiplying $word_1$ and $word_2$ ($|V| = 400$).
4. Adding $word_1$ and $word_2$ ($|V| = 400$).
5. Concatenating the vector resulting from the multiplication of $word_1$ and $word_2$ with the vector resulting from the subtraction of $word_2$ from $word_1$ ($|V| = 800$).
6. Concatenating the vector resulting from the multiplication of $word_1$ and $word_2$ with an additional dimension indicating the direction of the effect (1 if the attribute is an attribute of $word_1$, 2 if the attribute is an attribute of $word_2$) ($|V| = 401$).

While element wise multiplication shows a higher performance than subtraction or addition, it results in a model where the features for `[seal, dolphin]` and `[dolphin, seal]` are the same, which means that the direction of the effect is lost. Concatenating the multiplied vector with the subtracted vector means the features for `[seal, dolphin]` are different from the features for `[dolphin, seal]`, which results in a higher accuracy, as the direction of the effect is preserved. Concatenating the multiplied vector with a dimension that encodes direction is not as effective.

4.2.3 Analysis of errors

Examining the predicted closest neighbours shows that the model often predicts a discriminative attribute that belongs to $word_1$, but not the one that had the highest frequency in the McRae dataset (for example, predicting that *flippers* is a discriminative attribute of `[seal, dolphin]`, which is correct, but not what we expected). The model also often predicts a discriminative attribute of $word_2$ (for example, predicting that *whiskers* is a discriminative attribute of the word pair `[dolphin, seal]`, when it actually is a discriminative attribute of `[seal, dolphin]`.

The model also sometimes predicts an attribute that $word_1$ and $word_2$ have in common instead of predicting a difference (for example, predicting *swims* for the word pair `[seal, dolphin]`). Table 4.14 shows the distribution of these types of errors. In this table, the reported accuracy counts every predicted attribute that belongs to $word_1$ as a correct prediction. For example,

for the word pair `[seal, dolphin]`, if the expected attribute was *whiskers* but the model predicted *flippers*, it would count as a correct answer.

TABLE 4.14: Linear Regression (SVD), detailed accuracy. Predicting $attr(w_1)$ counts as a correct answer.

	WUB(50)				
	conc.	sub.	multi.	multi. ⌒ sub.	multi. ⌒ dir.
expected	0	3	3	5	4
$attr(w_1)$	163	59	178	194	108
$attr(w_2)$	21	9	285	117	367
$attr(w_1) \wedge attr(w_2)$	23	8	153	6	10
$\neg(attr(w_1) \vee attr(w_2))$	1040	1224	604	878	708
accuracy	0.13	0.04	0.14	0.16	0.09

Further analysis of the incorrect responses ($\neg(attr(w_1) \vee attr(w_2))$) was conducted for the model which concatenates the multiplied vector with the the subtracted vector. Since the McRae dataset does not include every possible attribute, the errors were manually annotated to evaluate whether the model correctly predicts attributes that were not in the original dataset. Six types of predictions were annotated:

1. $attr(w_1)$: The model correctly predicted a discriminative attribute of $word_1$ and $word_2$.
2. $attr(w_2)$: The model incorrectly predicted a discriminative attribute of $word_2$ and $word_1$.
3. $attr(w_1) \wedge attr(w_2)$: The model incorrectly predicted a similarity between $word_1$ and $word_2$.
4. *same type*: The model incorrectly predicted an attribute which is of the same type as the expected answer.
5. *same domain*: The model incorrectly predicted an attribute which is semantically related to $word_1$, $word_2$, or both.
6. *unrelated*: The model incorrectly predicted an attribute which is not semantically related to either $word_1$ or $word_2$.

Table 4.15 shows examples of these predictions, while table 4.16 shows their distribution. Overall, this analysis showed that the model fails to generalize to new attributes: for the vast majority of predictions, the predicted attribute is an attribute that was present in the training set (for example, *tail*). The rest of the time, the model fails to predict an attribute, and the closest neighbour of the resulting vector ends up being an unrelated word (for example, *protruding*).

TABLE 4.15: Examples of errors.

	w_1	w_2	*expected*	*predicted*
1. $attr(w_1)$	tiger	spider	teeth	tail
2. $attr(w_2)$	beans	parsley	brown	leaves
3. $attr(w_1) \wedge attr(w_2)$	beaver	rabbit	wood	fur
4. *same type*	pillow	bag	white	pink
5. *same domain*	caterpillar	spider	long	wings
6. *unrelated*	ambulance	taxi	wheels	protruding

TABLE 4.16: Number of errors.

	count	**proportion**
$attr(w_1)$	46	0.06
$attr(w_2)$	21	0.02
$attr(w_1) \wedge attr(w_2)$	89	0.12
same type	33	0.04
same domain	58	0.08
unrelated	458	0.64

Figure 4.4 shows the distribution of predictions by combining the results from Table 4.14 and Table 4.16: the model correctly predicts a discriminative attribute 23% of the time.

While the results have shown that the model fails to generalize, 23% accuracy is promising and different methods could give better results. It would also be possible to retrieve the top 10 neighbours of the predicted vector (instead of the top 1) to see if the expected attribute is amongst them.

4.3 Predicting dichotomous variables

The discriminative attributes task can also be framed in a different way: instead of trying to predict which attributes differentiate two words, we can ask the question "Is word c a discriminative attribute between word a and word b?" There can only be two answers, yes or no. For the triple `[seal, dolphin, whiskers]`, the model should predict *yes*, while for the triple `[dolphin, seal, whiskers]` the model should predict *no*, since dolphins do not have whiskers. For this purpose, the dataset had to be supplemented with negative examples (see Section 3.2). The test set used is the test set that was manually checked, as described in Section 3.3.

All positive examples are differences between $word_1$ and $word_2$. The negative examples can be similarities between $word_1$ and $word_2$ or attributes that belong to neither $word_1$ nor $word_2$. For that last type of attributes, the examples were selected randomly so that the number of negative examples

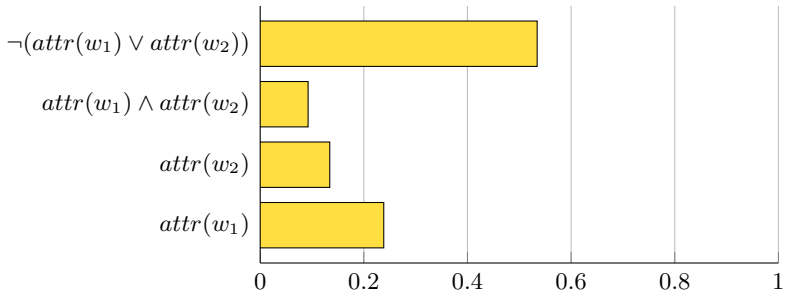


FIGURE 4.4: Distribution of predictions (multiply⌒subtract, WUB(50), SVD), combining results from Table 4.14 and Table 4.16.

matched the number of positive examples. Lexical overlap was minimized: no attributes present in the test set are also present in the training set. Table 4.17 shows the distribution of these three types of attributes under the WUB(100) parameter. Table 4.18 shows the final size of the training and testing sets for this task.

TABLE 4.17: Three types of negative examples.

	WUB(100)
differences	28694
similarities	9524
neither	19170

TABLE 4.18: Size of the dataset for the binary task.

	WUB(100)
training, positive	18057
training, negative	23224
testing, positive	904
testing, negative	1272

Figure 4.5 shows a visualisation of the concatenated triples of the test set using the word2vec word embeddings. T-distributed stochastic neighbour embedding (t-SNE) was used to reduce the embeddings to two dimensions. The t-SNE algorithm uses similarity between data points to construct a probability distribution while minimizing Kullback-Leibler divergence between the probabilities of the low-dimensional representation and the original high-dimensional dataset (Maaten and Hinton, 2008). The goal of the probability distribution is to cluster data points together, so that objects that are very similar have a high chance of being picked, while objects that are dissimilar have a very low chance of being picked.

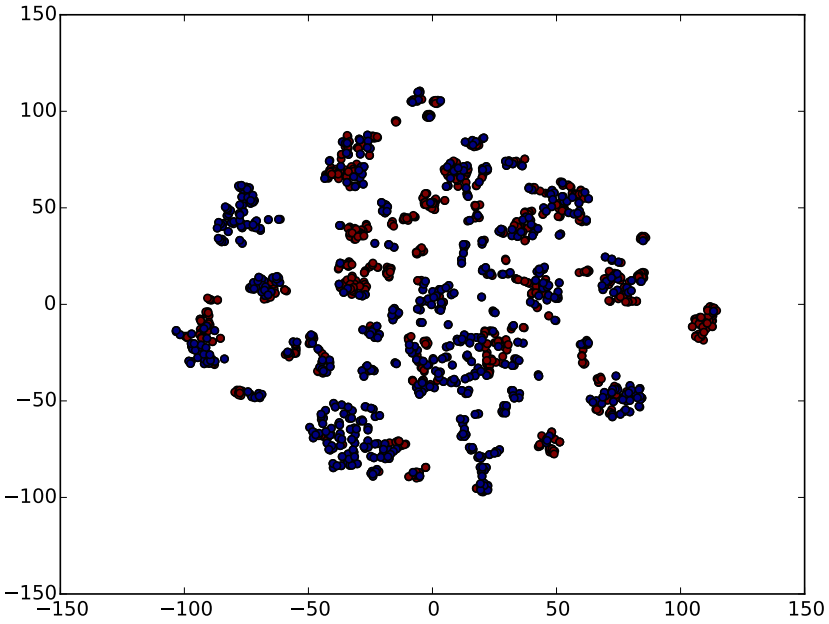


FIGURE 4.5: Visualization of the two classes of the test set using t-SNE.

4.3.1 Binary logistic regression

In the binary setup, the probability that the input point Y belongs to a class k is computed using the logistic function:

$$P(Y = k|X_1, X_2) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2)}}$$

In this task, the independent variable is the concatenation of the vectors for $word_1$, $word_2$ and $attribute$ ($|V| = 1200$). The dependent variable is 1 (the triple includes a discriminative attribute) or 0 (the triple does not include a discriminative attribute).

Binary logistic regression was implemented using scikit-learn (Pedregosa et al., 2011).

Performance

For each measure, the weighted average is computed by adding the products of the score of each class and their weight. As an example, the average precision score in Table 4.19 is computed with:

$$w_1 = \frac{1271}{2176} = 0.58$$

TABLE 4.19: Binary logistic regression (SVD).

	WUB(100)			
	precision	recall	f1	support
0	0.56	0.36	0.44	1272
1	0.40	0.60	0.48	904
avg	0.49	0.46	0.46	2176

TABLE 4.20: Binary logistic regression (word2vec).

	WUB(100)			
	precision	recall	f1	support
0	0.57	0.39	0.46	1272
1	0.41	0.58	0.48	904
avg	0.50	0.47	0.47	2176

$$w_2 = \frac{904}{2176} = 0.41$$

$$AP = 0.56 \cdot 0.58 + 0.40 \cdot 0.41 = 0.49$$

Overall, the performance of the model is poor, as it does not exceed a random baseline. We then turned to a different learning model that uses gradient descent.

4.3.2 Neural network

A feedforward neural network is a classification algorithm which consists of a number of layers - each of which apply a different function to the data. Each node in a given layer is connected to all nodes in the previous layer, and is assigned a given weight. For this task, a single layer perceptron network was used: the inputs are fed directly to the outputs via a series of weights. The perceptron architecture was first proposed by (Rosenblatt, 1958).

The dataset set used for this model is the same as the one used for the binary logistic regression task, which means the input layer has 1200 nodes (the number of dimensions of the three concatenated vectors for $word_1$, $word_2$ and $attribute$).

The weights and bias terms are initialized randomly. The weights w are applied to the input vector x through element-wise multiplication. The bias term b is added to the weighted vector W through element-wise addition. The activation function ϕ then computes the sigmoid of the input B element-wise.

$$W = x \cdot w$$

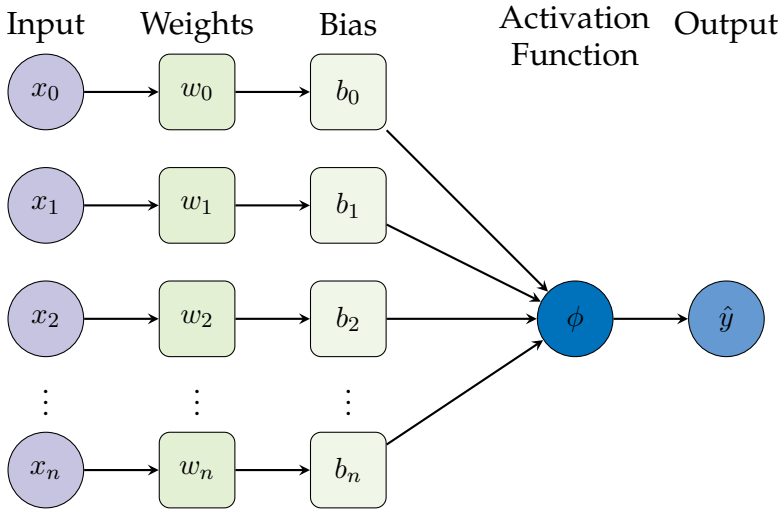


FIGURE 4.6: Architecture of a single-layer neural network.

$$B = W + b$$
$$\phi = \frac{1}{1 + e^{-B}}$$
$$\hat{y} = argmax(\phi, 1)$$

The output of the activation function is used to determine the value of the $\hat{y}$ label (argmax returns the index of the maximum value, e.g. 0 or 1). If the predicted label $\hat{y}$ matches the expected label y, no changes to the weights are made. Otherwise, the weights need to be updated. The cost function indicates how large the error is. In this model, the cost function calculates the mean squared error:

$$Cost = \frac{(\phi - y)^2}{2}$$

If the weights need to be updated, gradient descent is used to minimize the error. The gradient descent algorithm iteratively moves towards a set of weights that minimize the cost function, using a learning parameter η:

$$w^{new} = w^{old} - \eta \cdot (\phi - y) \cdot \phi(1 - \phi) \cdot x$$

The learning rate η (a value between 0 and 1) indicates how much of a step is taken during each iteration: a large learning rate may overstep the minimum of the function, while a small learning rate will require more iterations to reach it. The learning rate can also be adjusted during training. In this model, an exponential decay function was used to update the learning rate during training. The function was computed with the following parameters:

$$decayed\_learning\_rate = learning\_rate \cdot 0.95^{\frac{1}{X_m}}$$

Where X_m is the number of training examples. In the subsequent section, *learning rate* indicates the original `learning_rate` input value.

These steps are repeated until a stopping condition is satisfied: either when the change in cost is below a certain threshold, or after a set number of iterations. In our experiments, all models were stopped after 1000 epochs.

The TensorFlow library (Abadi et al., 2015) was used to implement this model.

Performance

TABLE 4.21: Neural network (SVD, 1000 epochs).

learning rate	train accuracy	test accuracy
0.01	0.7859	0.4283
0.001	0.7231	0.4319
0.0001	0.6012	0.4701

Table 4.21 suggests that the model could be overfitting to the training set, as the training accuracy is much higher than test accuracy. To help mitigate that effect, L2 regularization was added to the cost function:

$$Cost = \frac{(\phi - y)^2}{2} + \frac{w^2}{2}$$

TABLE 4.22: Neural network with l2 regularization term (SVD, 1000 epochs).

learning rate	train accuracy	test accuracy
0.01	0.7294	0.3653
0.001	0.7670	0.4260
0.0001	0.6036	0.5340

TABLE 4.23: Neural network with L2 regularization term (W2V, 1000 epochs).

learning rate	train accuracy	test accuracy
0.01	0.7028	0.4136
0.001	0.7784	0.5863
0.0001	0.6535	0.5776

Table 4.22 seems to indicate that L2 regularization does help minimize overfitting for the SVD embeddings, but only when a small learning rate

is used. Using the word2vec embeddings and L2 regularization, the best accuracy is reached with a learning rate of 0.001.

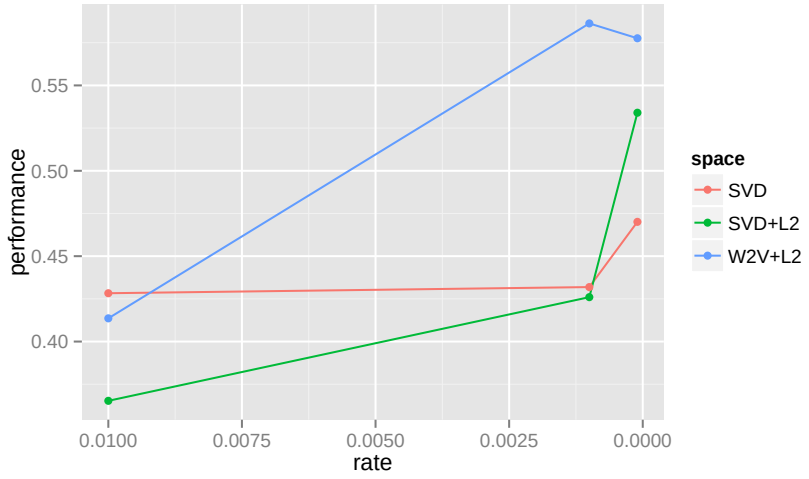


FIGURE 4.7: Performance of different word embeddings as a function of learning rate in the binary task.

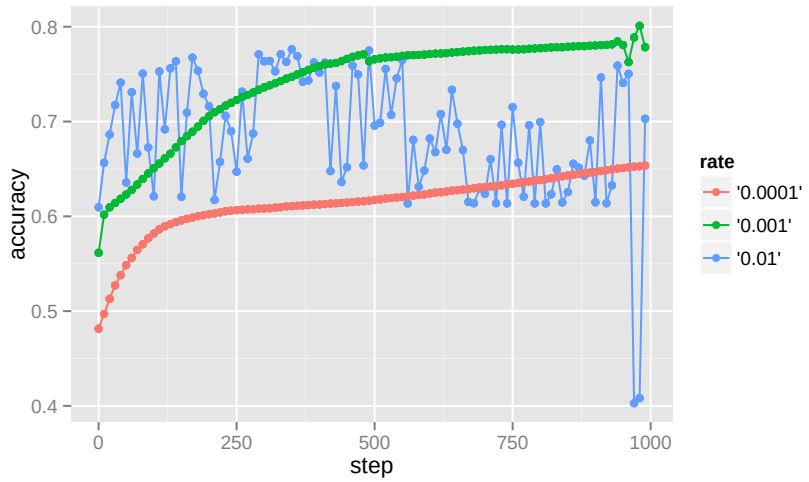


FIGURE 4.8: Training accuracy over 1000 epochs for different learning rates (W2V, L2 regularization).

Figure 4.7 compares the performance of different word embeddings with and without L2 regularization, while Figure 4.8 shows the evolution of training accuracy over 1000 epochs. We can see from figure 4.8 that 0.01 is too big a learning rate and does not lead to consistent performance gains over time: this means that any increase or drop in accuracy reported for this learning rate is not significant.

Chapter 5

Discussion

5.1 Performance analysis

5.1.1 Difficulty of the task

Three types of tasks were used in the aforementioned experiments. The first task was to predict discrete variables, the second task was to predict continuous variables, and the third task was to predict dichotomous variables.

By design, the first task is much easier than the other two, as it requires the discrete variable that will be predicted to be included in the training set. Statistics for $word_1$ overlap show that the training set contains 509 unique first words and the test set contains 414 unique first words. There are 413 first words in common between the training set and the test set: the random split only included one first word in the test set that did not appear in the training set. This confirms that the high performance of the logistic regression models for this task is due to attribute memorization: the model learns that *whiskers* is an attribute which belongs to *seal*, instead of learning that *whiskers* is an attribute that differentiates *seal* from *dolphin*.

The second task is thus much more difficult because its design allows for the dataset to be split in a way that minimizes lexical overlap. While the performance of the linear regression model on this task was poor, it was also very informative: it showed that the model was unable to generalize to new attributes, and instead only predicted attributes it had learned from the training set.

The design of the third task also allows for the dataset to be split so that the test set only contains attributes that have never been seen before. Since there are only two classes in this task, its random baseline is higher than it is for other tasks. Figure 5.1 shows that in the third task the best model exceeds a random baseline by 7%, while in the first task the best model exceeds a random baseline by over 96%.

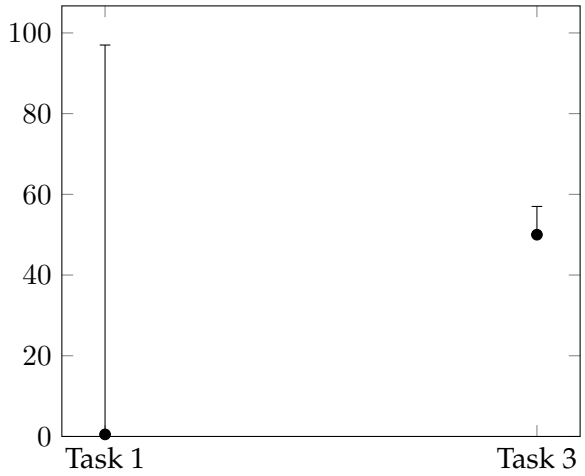


FIGURE 5.1: Improvement of the best models over a random baseline for Task 1 (multi-class logistic regression) and Task 3 (neural network). The line is the accuracy of the model, the dot is the accuracy if the answer was chosen randomly.

In terms of practical applications, the third task is also the most useful: while the first and second task only allow for a definite number of attributes to be classified, a model created for the third task could be used to traverse a distributional space and retrieve all discriminative attributes of any given word as defined by a learned threshold.

5.1.2 Quality of the word vector representations

Across the board, the different word embeddings being used had a significant impact on performance: the first task quickly showed that the NMF reduced embeddings could not compete with the SVD reduced embeddings. Overall, the the word2vec word embeddings beat the performance of the count-based vector representations.

As shown in Figure 5.2, when predicting discrete variables using multi-label logistic regression, as well as when predicting dichotomous variables using binary logistic regression, using the word2vec embeddings resulted in a slightly improved performance (+3% ; +1%). When predicting dichotomous variables using a neural network approach, using the word2vec embeddings improved the performance by 4%.

5.1.3 Quality of the training and testing items

While the test set using in the binary task was manually checked, the training set was not. When the test set was annotated, 54.8% of positive items and 12.5% of negative items were discarded. The reason why the dataset contains

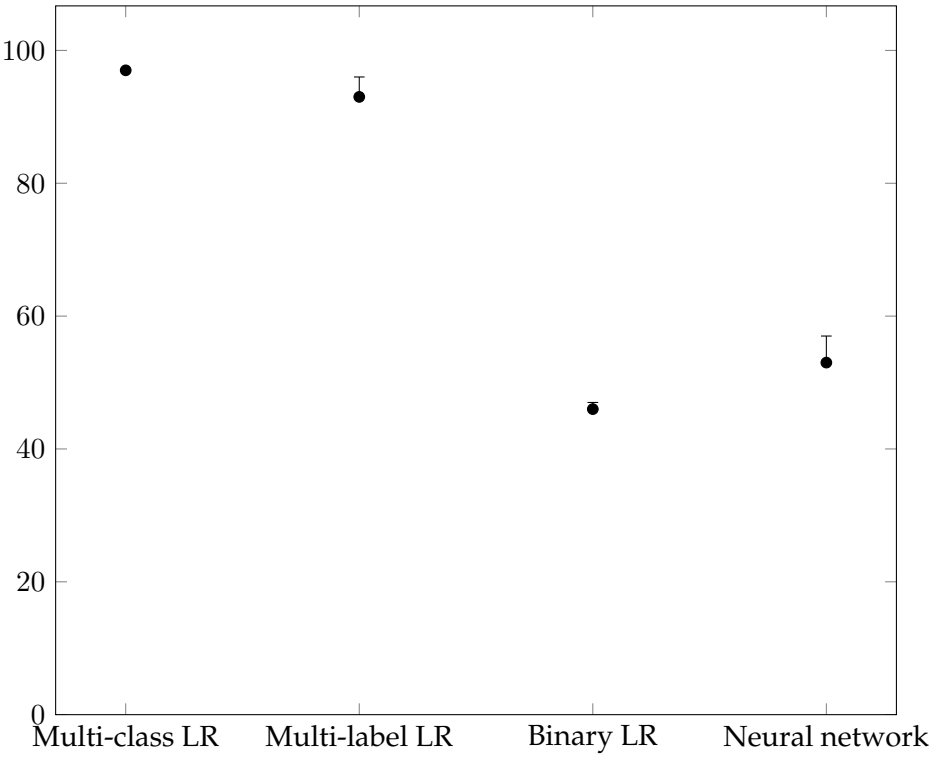


FIGURE 5.2: Improvements which can be attributed to the word2vec embeddings for the best models. The line is the accuracy of the model using word2vec embeddings, the dot is the accuracy of the same model using SVD embeddings.

a high proportion of false positives is mostly due to the fact the McRae feature norms dataset is not extensive enough. When it comes to colors for example, most concepts are not annotated with every color that can be attributed to them, which means that positive triples are falsely generated. If the training set was manually checked as well, we can expect that the same proportion of items would be discarded, which would in turn greatly improve the performance of the models.

5.1.4 Suitability of the learning algorithms

The datasets that have been used so far may not be linearly separable. There are alternatives to linear models which could give better results: for the continuous variables task, Epsilon-Support Vector Regression with a non-linear function could produce a model that can better generalize to new examples.

For the dichotomous variables task, using more epochs and strengthening the regularization term could improve the performance of the current model. Adding more layers and trying different activation functions could also take us in the right direction.

5.2 Future work

Future work should start by having the training set validated by human annotators. Efforts should be made to solve the binary classification task, as it has the most practical applications. The performance of the single-layer neural network was promising: exploring more parameters of the network (e.g. adding more layers, using vector offsets) could lead to a performance breakthrough.

5.3 Conclusion

Being able to detect that concepts are similar is not enough for a full-fledged language understanding system: detecting in which ways concepts differ from each other is a basic requirement.

This work has shown that the discriminative attributes task is not trivial, and that a simple approach to the problem is not enough to solve it. While the models that have been implemented so far work well when all discriminative attributes are present in the training set, generalizing to new features remains a challenge. This work has also shown how a dataset can be built to evaluate a system's abilities at detecting semantic differences, and how a task should be designed to produce meaningful results.

Natural language understanding requires many stepping stones: in the future, solving the discriminative attributes will take us further than similarity tasks currently can.

Bibliography

Abadi, Martín et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: http://tensorflow.org/.

Agirre, Eneko et al. (2009). "A study on similarity and relatedness using distributional and wordnet-based approaches". In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 19–27.

Baroni, Marco, Raffaela Bernardi, and Roberto Zamparelli (2014). "Frege in space: A program of compositional distributional semantics". In: *Linguistic Issues in Language Technology* 9.

Baroni, Marco, Georgiana Dinu, and Germán Kruszewski (2014). "Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Vol. 1, pp. 238–247.

Baroni, Marco and Alessandro Lenci (2011). "How we BLESSed distributional semantic evaluation". In: *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*. Association for Computational Linguistics, pp. 1–10.

Baroni, Marco et al. (2009). "The WaCky wide web: a collection of very large linguistically processed web-crawled corpora". In: *Language resources and evaluation* 43.3, pp. 209–226.

Batchkarov, Miroslav et al. (2016). "A critique of word similarity as a method for evaluating distributional semantic models". In: *First Workshop on Evaluating Vector Space Representations for NLP (RepEval 2016)*.

British National Corpus (2001). In: URL: http://www.natcorp.ox.ac.uk.

Bruni, Elia, Nam-Khanh Tran, and Marco Baroni (2014). "Multimodal Distributional Semantics." In: *J. Artif. Intell. Res.(JAIR)* 49.1-47.

Brysbaert, Marc, Amy Beth Warriner, and Victor Kuperman (2014). "Concreteness ratings for 40 thousand generally known English word lemmas". In: *Behavior research methods* 46.3, pp. 904–911.

Cann, Ronnie, Ruth Kempson, and Eleni Gregoromichelaki (2009). *Semantics*. Cambridge University Press.

Church, Kenneth Ward and Patrick Hanks (1990). "Word association norms, mutual information, and lexicography". In: *Computational linguistics* 16.1, pp. 22–29.

Cox, David R (1958). "The regression analysis of binary sequences". In: *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 215–242.

Cree, George S and Ken McRae (2003). "Analyzing the factors underlying the structure and computation of the meaning of chipmunk, cherry, chisel, cheese, and cello (and many other such concrete nouns)." In: *Journal of Experimental Psychology: General* 132.2, p. 163.

Faruqui, Manaal et al. (2016). "Problems With Evaluation of Word Embeddings Using Word Similarity Tasks". In: *First Workshop on Evaluating Vector Space Representations for NLP (RepEval 2016)*.

Firth, John R (1957). "A synopsis of linguistic theory, 1930-1955". In: *Studies in Linguistic Analysis*.

Harris, Zellig S (1954). "Distributional structure". In: *Word* 10.2-3, pp. 146–162.

Krebs, Alicia and Denis Paperno (2016). "When Hyperparameters Help: Beneficial Parameter Combinations in Distributional Semantic Models". In: *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics (*SEM 2016)*, pp. 97–101.

Landauer, Thomas K and Susan T Dumais (1997). "A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge." In: *Psychological review* 104.2, p. 211.

Lazaridou, Angeliki, Nghia The Pham, and Marco Baroni (2016). "The red one!: On learning to refer to things based on their discriminative properties". In: *arXiv preprint arXiv:1603.02618*.

Lee, Daniel D and H Sebastian Seung (1999). "Learning the parts of objects by non-negative matrix factorization". In: *Nature* 401.6755, pp. 788–791.

Levy, Omer and Yoav Goldberg (2014). "Neural word embedding as implicit matrix factorization". In: *Advances in Neural Information Processing Systems*, pp. 2177–2185.

Levy, Omer, Yoav Goldberg, and Ido Dagan (2015). "Improving distributional similarity with lessons learned from word embeddings". In: *Transactions of the Association for Computational Linguistics* 3, pp. 211–225.

Li, Yuhua, Zuhair A Bandar, and David McLean (2003). "An approach for measuring semantic similarity between words using multiple information sources". In: *IEEE Transactions on knowledge and data engineering* 15.4, pp. 871–882.

Maaten, Laurens van der and Geoffrey Hinton (2008). "Visualizing data using t-SNE". In: *Journal of Machine Learning Research* 9.Nov, pp. 2579–2605.

Martin, James H and Daniel Jurafsky (2009). *Speech and language processing*. Second edition. Prentice Hall.

– (2016). *Speech and language processing*. Third edition. Unpublished. URL: https://web.stanford.edu/~jurafsky/slp3/.

McRae, Ken et al. (2005). "Semantic feature production norms for a large set of living and nonliving things". In: *Behavior research methods* 37.4, pp. 547–559.

Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig (2013). "Linguistic Regularities in Continuous Space Word Representations." In: *HLT-NAACL*. Vol. 13, pp. 746–751.

Mikolov, Tomas et al. (2013a). "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*, pp. 3111–3119.

Mikolov, Tomas et al. (2013b). "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781*.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Rapp, Reinhard (2003). "Word sense discovery based on sense descriptor dissimilarity". In: *Proceedings of the ninth machine translation summit*, pp. 315–322.

Rogers, Henry (2004). "Writing Systems: A Linguistic Approach (Blackwell Textbooks In Linguistics)". In:

Rosenblatt, Frank (1958). "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6, p. 386.

Sahlgren, Magnus (2008). "The distributional hypothesis". In: *Italian Journal of Linguistics* 20.1, pp. 33–54.

Saint-Dizier, Patrick and Viegas Evelyne, eds. (1995). *Computational lexical semantics*. Cambridge University Press.

Saussure, Ferdinand de (1916 [ed. 1989]). *Course in General Linguistics*. New York Philosophical Library.

Schütze, Hinrich (1992). "Dimensions of meaning". In: *Supercomputing'92., Proceedings*. IEEE, pp. 787–796.

Silberer, Carina and Mirella Lapata (2014). "Learning Grounded Meaning Representations with Autoencoders." In: *ACL (1)*, pp. 721–732.

Swart, Henriette de (1998). *Introduction to natural language semantics*. CSLI publications.

Turney, Peter D, Patrick Pantel, et al. (2010). "From frequency to meaning: Vector space models of semantics". In: *Journal of artificial intelligence research* 37.1, pp. 141–188.

Tversky, Amos (1977). "Features of similarity." In: *Psychological review* 84.4, p. 327.

Vylomova, Ekaterina et al. (2015). "Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning". In: *arXiv preprint arXiv:1509.01692*.