Attributive adjective phrases behaving non-compositionally in distributional semantics

TERM PAPER: FOUNDATIONS OF DISTRIBUTIONAL SEMANTICS
LECTURER: ARIANNA BETTI

Logic and Language Group
Department of Philosophy
University of Amsterdam

Mick de Neeve
9305017/mick@live.nl

June 11, 2017

Abstract
In this study, the information gain of hypernym/hyponym patterns is compared to simple attributive adjective phrases consisting of an adjective and a noun, based on embedding vectors built from a large Wikipedia corpus. Some of the latter phrase types are associated with discriminatory bias while others are not. Instead of a difference between such phrase classes, it is found that the information gain associated with processing any of the phrases studied is on the same order as non-compositional single word hypernym/hyponym patterns. But when components are combined compositionally, this information gain is substantially larger. It is concluded that the compositionality which formal logics associate with such compound phrases, is not supported by the distributional data, at least not when large context windows are used.

1 Introduction

In 1954, Harris [Har54] conjectured that meaning is a function of distributional structure, a view that has since grown into the so-called Distributional Hypothesis (DH): words in similar contexts tend to have similar meanings. This provides an interesting alternative to the traditional way of doing semantics using formal logical models. In distributional models, words are represented as vectors in a multidimensional space of contextual co-occurences. While in logical models, relations between lexical items need to be specified explicitly, distributional approaches effectively leave this to the corpora on which distributional models are based.

But these relationships can be made explicit, for instance, Mikolov *et al.* [MCCD13] showed that a vector representation close to the one for *queen* can be retrieved

by taking the one for *king*, subtracting *man* and adding *woman*. This is based on the closeness of the representations in the vector space, e.g. the contexts of *king* and *queen*, and of *man* and *woman*, are similar. However, the (cosine) distance between *man* and *king* is the same as between *king* and *man*, i.e. it is not immediately obvious how to discern that *man* is more general than *king*.

To overcome this, Herbelot and Ganesalingam [HG13] have proposed to use the Kullback-Leibler divergence, also known as relative entropy or information gain, to distinguish between (e.g.) hypernyms and hyponyms. The idea is that given some degree of knowledge or expectation, it takes a particular amount of effort to process what is in the text, and what is subsequently picked up can be expressed in bits of information.[1] Probability distributions can be viewed as degrees of knowledge, with the uniform distribution a representation of the state of ignorance where everything is as likely as everything else. But the contextual word co-occurences in distributional semantics are also essentially probabilistic, and can be considered as measures of word informativeness.

Broadly speaking, a more general word, being less informative, will have a more predictable contextual distribution, while a more specific or informative one may be expected to be distributed closer to randomness (the opposite of ignorance). Herbelot and Ganesalingam measure how much the conditional probability distributions of word pair contexts given the words themselves, diverge from the prior distributions of just the respective contexts. The word with the smallest divergence is then conjectured to be the hypernym.

In this study, the focus is on the related category of attributive adjective phrase pairs such as *village* / *small village*, since these ought to behave in a similar way with respect to the informativeness of their contextual distributions. The original motivation was to study distributional behaviour of discriminatorally biased phrases like *black woman* (following Herbelot *et al.* [HvRM12]), but it was found that these have more in common with non-discriminatory ones than expected, while differing from hypernym/hyponym pairs in an unforseen way.

2 Background

This section gives some background for the present study, consisting of a description of Herbelot and Ganesalingam's method, as well as the relation to formal logic.

2.1 Generality and entropy

Herbelot and Ganesalingam advance the conjecture that semantic content is related to information theory. They formulate a version of the distributional hypothesis, namely that a word's generality varies with the generality of its context. Put differently, specific words are more informative about their contextual embedding than general words. The authors test this by taking a given set of hypernym/hyponym pairs, and computing Kullback-Leibler divergences

[1] One bit of information is defined as the amount needed to answer a yes/no question (cf. Shannon [Sha48]).

of probability distributions of these words given their contextual embedding, with the prior probability of these contexts for the whole corpus used.

For each target word pair, they calculate frequencies of context words given the targets, plus the frequencies of each target word and context word considered separately (priors). With P the probability distribution of context given word, and Q the prior contextual probability distribution, the Kullback-Leibler divergence is given as follows:

$$D_{KL}(P\|Q) = \sum_i ln(\frac{P(i)}{Q(i)}P(i))$$

A precision of almost 80% is achieved on the task of identifying hypernymy direction. This study uses the same formula, except that instead of the natural logarithm ln, the base 2 logarithm is used.[2] Moreover, instead of prior contextual probability, the uniform distribution is used, i.e. generality is measured against ignorance.

2.2 Semantics and logic

While a key strength of distributional models lies in the ability to model gradations in meaning by exploiting cosine similarities in vector representations, logic tasks are a weak point, as indicated in Section 1: if it is not straightforward to decide the hypernymy direction of *man* and *king*, it will be hard to do inference.

One of the core strengths of formal logics is compositional semantics, i.e. the ability to construct complex meanings out of simpler ones.[3] For instance, *small village* may be represented as the intersection of small things and villages. Erk ([Erk12], p.642) describes basic operations on vectors to combine them compositionally: addition, pairwise multiplication and the tensor product. She suggests that which method works best depends on how much of word context is considered, and points out that for small context windows, multiplication is better than addition. In this paper a large window is used, with vector addition producing the most relevant word similarities.

2.3 Divergence in predict models

Herbelot and Ganesalingam rely on so-called count models to obtain their results. In this study, a predict model is used, after the aforementioned Mikolov *et al.* [MCCD13]. In such models, preceding and successive contexts are used to train a model to maximise the probability of correctly predicting a word (or vice versa, i.e. predict context). Baroni *et al.* [BDK14] compared count and predict models and found that the latter type is to be prefered. But its use has the consequence that the computation of Kullback-Leibler divergence as given in Section 2.1 needs to be adapted, as there is no direct way to retrieve conditional word frequencies.

A predict model's architecture is based on a neural network with a single hidden

[2] This is conventional in computer science as it expresses the result in bits.
[3] A key effort in the application to natural language is due to Montague [Mon73].

layer, with weights initialised randomly and trained to maximise the probability of an output word given an input context. This has the effect of projecting vector representations of size V, the vocabulary, onto representations of size K, the subset consisting of words found within the context window in the corpus. In turn this has the effect of representing words in their contextual space.

A single vector consists of the values assigned by training, and has no immediate apparent meaning in isolation. But word probabilities for the maximization are effectively computed during training by softmax (cf. [Ron14], p.2), a logisitic function which can normalise the positive and negative reals in the vectors to lie in the $[0, 1]$ interval. However, it is not these word vectors themselves that (latently) encode contextual occurence probabilities, but the input context vectors that are used to predict the word vectors.

Given a contextual input vector C of size K for a word w, the contextual probability distribution $\rho(w)$ for w can be estimated component-wise as follows:

$$\rho(c_i) = \frac{\exp(c_i)}{\sum_{j=1}^{K} \exp(c_j)}$$

However, it is less clear how the prior contextual distributions as used by Herbelot and Ganesalingam (cf. Section 2.1) might be obtained, hence the following conjecture is made. Given two words w_1 and w_2, their prior contextual distributions C_1 and C_2, and the uniform distribution U: if it holds that $D_{KL}(\rho(w_1)\|U) < D_{KL}(\rho(w_2)\|U)$, then it is likely that also $D_{KL}(\rho(w_1)\|C_1) < D_{KL}(\rho(w_2)\|C_2)$; i.e. w_1 is more general than w_2 in Herbelot and Ganesalingam's sense. That is, divergence from the ignorant distribution is used as an approximation for divergence from the contextual priors.

To compensate for the lack of using contextual priors, the additional computation of $D_{KL}(\rho(w_2)\|\rho(w_1))$ is done. In other words, information gain is considered not only for the move from ignorance to general ($KL_{I\to G}$) and from ignorance to specific ($KL_{I\to S}$), but also from general to specific ($KL_{G\to S}$). It will turn out that this last level of information gain can justify the classification of collated phrases along with non-compositional single-word hypernym/hyponym pairs.

3 Experiments

This section describes experiments performed to measure Kullback-Leibler divergence between simple two-word phrases and the composition of their components.

3.1 Data

The data used to build vectors was a Wikipedia 'latest pages' articles dump, downloaded from `https://download.wikimedia.org/enwiki/` on May 30, 2017 and representing pages up to May 20. It was preprocessed using Attardi's WikiExtractor code (`https://github.com/attardi/wikiextractor`) to remove

markup and output plain text. The text was duplicated and a copy was processed further to collate the set of terms in Table 1.

black woman	white woman	black man	white man
old woman	old man	small village	short story

Table 1: Collated two-word phrases

The initial idea of these phrases was to have four race-related ones and four without race, with the former split between black and white, and female and male, and the latter between female/male on the one hand, and non-racial and non-gender control categories on the other. However, these differences were ultimately ignored, and comparisons were made with the non-phrasal hypernym/hyponym pairs in Table 2.

fruit/banana	building/house	furniture/table	vehicle/car
color/green	liquid/water	composer/mozart	country/france

Table 2: Hypernym/hyponym pairs

To obtain vector representations, the Word2Vec (cf. Mikolov *et al.* [MCCD13] module included in the Gensim Python library[4] was used. Since [HvRM12] use a context window set to whole sentences, a window of 10 words was used here, in order to captures more of a word's topicality as opposed to information about its function (cf. Levy and Goldberg [LG14], p.3.). It must be noted that this decision was originally made with an eye to capturing discriminatory context. Two models were produced, an intersective and a phrasal one, following Herbelot *et al.* For the latter, the words in Table 1 were glued together in the corpus before training the model.

3.2 Combining vectors

With the intersective model, the methods from [Erk12] refered to in Section 2.2 were tried in order to combine the vectors for the phrases in Table 1, i.e. combinations were produced by addition, multiplication, and tensor product. That is, vectors for (e.g.) *woman* and *black* were combined to produce a potential vector to represent the combination *black woman*.

For each vector combination method, the top 20 most similar words were retrieved, and these were intersected with the 20 most similar words for the collated version of the phrase from the phrasal model. The combination method that yielded the largest intersections was then selected to compute Kullback-Leibler divergences to be compared to the hypernym/hyponym pairs of Table 2.

A full description of how the results were obtained and can be reproduced, including Python source code, is included in Appendix A at the end of this paper.

[4] https://radimrehurek.com/gensim/models/word2vec.html

3.3 Results

Addition turned out to be the best vector combination method. This yielded an average of $5\frac{3}{8}$ shared similar words out of 20 between the intersective and phrasal models. Tensor product produced 2, and oddly enough multiplication yielded 0, but this is likely due to the large context window, making this method unsuitable. All sorts of rare and bizarre similarities were produced by this method; some examples are listed in Table 6 in Appendix B.

For the hypernym/hyponym pairs of Table 2, Kullback-Leibler divergences were obtained that are listed in Table 3. $KL_{I\to G}$ refers to the divergence from ignorance to general, $KL_{I\to S}$ to the one from ignorance to specific, and $KL_{G\to S}$ to the divergence from general to specific.

	$KL_{I\to G}$	$KL_{I\to S}$	$KL_{G\to S}$
fruit/banana	0.37271	0.31223	0.01173
building/house	0.14437	0.09830	0.01201
furniture/table	0.36672	0.23636	0.03163
vehicle/car	0.27297	0.24536	0.00550
color/green	0.29603	0.09969	0.05924
liquid/water	0.47874	0.21326	0.06340
composer/mozart	0.35527	0.51267	0.02653
country/france	0.09352	0.16960	0.02205
average	*0.29754*	*0.23593*	*0.02901*

Table 3: Kullback-Leibler divergences for hypernym/hyponym pairs

For the collated phrases of Table 1, the divergences from the phrasal model are in Table 4.

	$KL_{I\to G}$	$KL_{I\to S}$	$KL_{G\to S}$
woman/blackwoman	0.22485	0.38547	0.02642
woman/whitewoman	0.22485	0.39067	0.02902
woman/oldwoman	0.22485	0.45977	0.04915
man/blackman	0.13267	0.25215	0.02506
man/whiteman	0.13267	0.29613	0.04191
man/oldman	0.13267	0.32368	0.04730
village/smallvillage	0.15915	0.41140	0.06195
story/shortstory	0.16606	0.40989	0.06331
average	*0.17472*	*0.36615*	*0.04302*

Table 4: Kullback-Leibler divergences for phrasal model phrases

Finally for the same phrases, but now with their vectors combined by addition in the intersective model, the divergences are in Table 5.

The divergences for phrasal model in Table 4 turn out to be significantly closer to the hypernym/hyponym pairs of Table 3 than those for the intersective model

	$KL_{I \to G}$	$KL_{I \to S}$	$KL_{G \to S}$
woman/black	0.20846	0.62556	0.11413
woman/white	0.20846	0.64886	0.12404
woman/old	0.20846	0.62618	0.11262
man/black	0.12842	0.49329	0.12188
man/white	0.12842	0.47513	0.11297
man/old	0.12842	0.47390	0.11045
village/small	0.15980	0.63150	0.15183
story/short	0.16583	0.63016	0.15242
average	*0.16703*	*0.57557*	*0.12504*

Table 5: Kullback-Leibler divergences for intersective model phrases

in Table 5. The important columns are the far right ones; the divergences from general to specific. In Tables 3 and 4 these are on the same order, while in Table 5 these divergences are significantly higher, and on the same order as the first column. So it appears the collated phrasal model captures something about generality and specificity which the intersective model misses.

4 Discussion

This section discusses the results from the previous section, and relates this to non-compositionality in constructionism, and to the connectionism debate in cognitive science.

4.1 Compositionality

Although the the number of instances tested was small, some observations can be made. While close to hyponyms, attributive adjective phrases are not quite the same: there is a significant difference between the second columns of Tables 3 and 4. The higher value in the latter arguably reflects a semantic effort on processing an unfamiliar combined term, and in this case compositionality arguably plays a role. In Table 3 it cannot, and there consequently appears to be no significant difference between the divergences in the first two columns, i.e. from ignorance to the general and to the specific term.

But the rightmost column of Table 4 suggests that once a term is familiar, it behaves as a non-compositional unit, since the information gain from generality to specificity is frequently as small as it is for a hyponym. Table 5 on the the other hand suggests that little is to be gained from familiarity with a general term upon processing a specific one, since the values in the left and right column are roughly the same.

The high values in the middle column of Table 5 seem due to the inclusion of related adjectives such as *blond* for *black woman* and *large* for *small village*, since those appear high in the list of similar words found for these phrases in the intersective model. This is may be because of the large context window used, so the question how the Kullback-Leibler divergence patterns found here stand up in smaller contexts is a topic for further research.

But despite the apparent non-compositional behaviour, there may be a link between reduced information gain of attributive adjective phrases and the dynamic semantics of Groenendijk and Stokhof (who have been at pains to keep semantics compositional). For instance, p.13 of [GS00] describes the process of an agent successively adding information about a man while eliminating possible worlds. This might be compared to adding information about some entity from ignorance, eliminating n worlds on finding out it is a story, and m worlds on learning it is a short story, with $m < n$.

4.2 Constructionism

Linguists in the constructionist tradition would be less bothered than formal semanticists by the non-compositional behaviour of a collation like *old woman*. From their perspective, it is an example of a form-meaning pair[5] for which the meaning is not strictly predictable from its parts (cf. Goldberg [Gol03], p219), making it a so-called construction by definition.

There is evidence from Tomasello that children imitate phrases as wholes ([Tom00], p.156) rather than learn its constituents.[6] According to him, children pick up so-called items or holophrases (e.g.*"lemme-see"*), rather than words.

Learning then, is not so much about putting words together as pulling constructions apart at later stages of language acquisition. But constructions are not always eventually pulled apart; certain phrases apparently retain particular meanings which transcend the sum of their parts.

4.3 Connectionism

The question what the nature of distributional meaning is, has parallels with questions concerning the nature of connectionist cognition, a subject of debate in the 1990s. The correspondence is between DS and parallel distributed processing. The latter paradigm was juxtaposed with classical cognitive architecture by Smolensky [Smo95b], who advocated a distinction between classical symbols and connectionist sub-symbols. While symbols are manipulated by rules, sub-symbols participate in numerical computations. Moreover, there is a difference in the notion of context. Whereas context is manifest around symbols in the form of other symbols at the symbolic level, at the sub-symbolic level the context is manifested within symbols, consisting of other sub-symbols (*ibid*, p.34).

This is reminiscent of the situation in DS, where it is hard to separate the notion of meaning from the context, because meaning is all but defined as context itself. In other words, in distributional semantics as in connectionism, it is hard to satisfy what Borensztajn has more recently termed the context invariance criterion ([Bor11], p.88).

Smolensky considered neural representations as activity patterns among possibly very many neurons. Erk ([Erk12], p.646) refers to Smolensky in the context

[5] The pair here is the form-meaning pair, not the pair of constituents.
[6] As Chomsky [Cho57] would have it.

of tensor products, which he describes in [Smo95a], p.236, in an attempt to characterise how neurons can participate in symbolic representations. However, Borensztajn (*ibid*) argues that neural representations are not just activation patterns, but that information can be encapsulated in the brain and 'named', and refered to in higher-level processing not by its activation but by identity. Indeed, for a phrase like *black woman* or *short story* the brain may not bother to carry out neural pattern combinations, and process these terms as units.

5 Concluding remarks

This paper has demonstrated that when the meaning of attributive adjective phrases is considered as a function of a predictive distributive Wikipedia embedding model trained with a ten word context window, the bevaviour is close to that of hyponyms. This is considered as evidence that the semantics of such phrases should to a significant degree be considered non-compositionally.

Links have been suggested with dynamic semantics, with constructionism - where non-compositionality is considered quite normal; and the broader meaning debate of distributional semantics has been compared to the connectionist/-classicist debate of cognitive science. Further research is needed to determine to what extent context window size influences non-compositional or hyponym-like behaviour of attributive adjective phrases.

References

[BDK14] M Baroni, G Dinu, and G Kruszewski. 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*, Baltimore, Maryland, 2014.

[Bor11] G Borensztajn. *The neural basis of structure in language: Bridging the gap between symbolic and connectionist models of language processing*. PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam, 2011.

[Cho57] N Chomsky. *Syntactic Structures*. Mouton, 1957.

[Erk12] K Erk. Vector Space Models of Word Meaning and Phrase Meaning: A Survey. *Language and Linguistics Compass*, 6/10, 2012.

[Gol03] A Goldberg. Constructions: a new theoretical approach to language. *Trends in Cognitive Sciences*, 7(5), 2003.

[GS00] J Groenendijk and M Stokhof. Meaning in Motion. In *Reference and Anaphorical Relations*. Kluwer, 2000.

[Har54] Z Harris. Distributional Structure. *Word*, 10(2-3), 1954.

[HG13] A Herbelot and M Ganesalingam. Measuring semantic content in distributional vectors. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, volume 2, 2013.

[HvRM12] A Herbelot, E von Redecker, and J Müller. Distributional techniques for philosophical enquiry. In K Zervanou and A van den Bosch, editors, *Proceedings of the 6th EACL Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, 2012.

[LG14] O Levy and Y Goldberg. Dependency-based Word Embeddings. In K Toutanova and H Wu, editors, *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, Baltimore, Maryland, 2014.

[MCCD13] T Mikolov, K Chen, G Corrado, and J Dean. Efficient estimation of word representations in vector space. Technical report, Google Research, 2013. https://research.google.com/pubs/archive/41224.pdf.

[MM95] C Macdonald and G Macdonald, editors. *Connectionism: Debates on Psychological Explanation*. Blackwell, 1995.

[Mon73] R Montague. The Proper Treatment of Quantification in Ordinary English. In *Approaches to Natural Language*. Reidel, 1973.

[Ron14] X Rong. Word2Vec Parameter Learning Explained. arXiv.org, 2014. https://arxiv.org/abs/1411.2738.

[Sha48] C Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27, 1948.

[Smo95a] P Smolensky. Constituent Structure and Explanation in an Integrated Connectionist/Symbolic Cognitive Architecture. In Macdonald and Macdonald [MM95].

[Smo95b] P Smolensky. On the Proper Treatment of Connectionism. In Macdonald and Macdonald [MM95].

[Tom00] M Tomasello. First steps toward a usage-based theory of language acquisition. *Cognitive Linguistics*, 11(1/2), 2000.

A Data and code

This appendix contains the Python source code plus references to external code, as well as to the data used, and indicates how the results in this paper can be reproduced. The use of a Unix or Linux operating system with the availability of the Python programming language is assumed.

The compressed 13GB Wikipedia articles dump can be downloaded at `https://dumps.wikimedia.org/enwiki/20170520/enwiki-20170520-pages-articles.xml.bz2`. Using WikiExtractor,[7] the dump file can then be extracted and preprocessed to a directory `wikint/` by issuing `python WikiExtractor.py -o wikint enwiki-20170520-pages-articles.xml` in a command shell. This creates a range of subdirectories `AA/` to `EN/`, each with files named `wiki_00` to `wiki_99` of 1MB each.

After processing the data is to be copied to another folder `wikphr`, after which the script `sedphrases.sh` in Listing 1, collates the phrases in Table 1. This can take up to five hours to run, but the use of `grep` to search for the phrases and make a file list, before `sed` does the actual replacements, does speed things up as opposed to letting `sed` do all the work.

```
#!/bin/bash

grep -rl 'black woman' wikphr/ | xargs sed -i 's/\bblack woman\b/blackwoman/gI'
grep -rl 'black women' wikphr/ | xargs sed -i 's/\bblack women\b/blackwomen/gI'
grep -rl 'white woman' wikphr/ | xargs sed -i 's/\bwhite woman\b/whitewoman/gI'
grep -rl 'white women' wikphr/ | xargs sed -i 's/\bwhite women\b/whitewomen/gI'
grep -rl 'black man' wikphr/ | xargs sed -i 's/\bblack man\b/blackman/gI'
grep -rl 'black men' wikphr/ | xargs sed -i 's/\bblack men\b/blackmen/gI'
grep -rl 'white man' wikphr/ | xargs sed -i 's/\bwhite man\b/whiteman/gI'
grep -rl 'white men' wikphr/ | xargs sed -i 's/\bwhite men\b/whitemen/gI'
grep -rl 'old woman' wikphr/ | xargs sed -i 's/\bold woman\b/oldwoman/gI'
grep -rl 'old women' wikphr/ | xargs sed -i 's/\bold women\b/oldwomen/gI'
grep -rl 'old man' wikphr/ | xargs sed -i 's/\bold man\b/oldman/gI'
grep -rl 'old men' wikphr/ | xargs sed -i 's/\bold men\b/oldmen/gI'
grep -rl 'small village' wikphr/ | xargs sed -i 's/\bsmall village\b/smallvillage/gI'
grep -rl 'small villages' wikphr/ | xargs sed -i 's/\bsmall villages\b/smallvillages/gI'
grep -rl 'short story' wikphr/ | xargs sed -i 's/\bshort story\b/shortstory/gI'
grep -rl 'short stories' wikphr/ | xargs sed -i 's/\bshort stories\b/shortstories/gI'
```

Listing 1: Prase collation shell script `sedphrases.sh`

So there are now two data directories, `wikint/` to train an 'intersective', and `wikphr/` to train a 'phrasal' model, in the terminology of Herbelot *et al.* It is assumed that these in turn reside in directories named `intersective/` and `phrasal/`, i.e. there is `intersective/wikphr/` and `phrasal/wikphr/`.

The training is done using the Python script `train_word2vec_with_gensim.py` by Pan Yang, which can be found at `https://github.com/panyang/Wikipedia_Word2vec/blob/master/v2/train_word2vec_with_gensim.py`. This script is to be copied into the `intersective/` and `phrasal/` directories and left as-is, except for lines 59 and 60. For the intersective version, these become as in Listing 2:

```
model.save("modint/word2vec_gensim")
model.wv.save_word2vec_format("modint/word2vec_org",
                              "modint/vocabulary",
```

Listing 2: Edit for lines 59-60 of `train_word2vec_with_gensim.py`

[7] `https://github.com/attardi/wikiextractor`

For the phrasal version, change `modint` to `modphr`.

The models are then trained[8] by issuing `python train_word2vec_with_gensim.py wikint` in the `intersective/` directory, and by subsequently entering `python train_word2vec_with_gensim.py wikphr` in the `phrasal/` directory. Each training session takes up to fourteen hours, depending on computer processing power.[9]

To compute Kullback-Leibler divergence as well as word/phrase similarities, the code in Listing 3 is used:

```
from __future__ import division
from gensim.models import Word2Vec
import numpy as np
import math


# Path is to Word2Vec model file. Returns the trained model.
#
def loadmodel(Path):

    Mod = Word2Vec.load(Path)

    return Mod


# Wx and Wy are words, Mod is a Word2Vec model, T is the
# number of similarities to retrieve, and J is a boolean int
# whether or not to produce Wy's vector by multiplying by
# Wx's. Returns the top T similar words for Wx and Wy.
#
def termsim((Wx,Wy), Mod, T, J):

    Vx = Mod.wv[Wx]
    Vy = Mod.wv[Wy]

    (Jx, Jy) = joinvec(Vx, Vy, J)
    (Tx, Ty) = incrementtop(T, J, 1)

    Sx = Mod.wv.similar_by_vector(Jx, topn=Tx)
    Sy = Mod.wv.similar_by_vector(Jy, topn=Ty)

    (Ox, Oy) = offsetlist(Sx, Sy, J, 1)

    return (getwords(Ox), getwords(Oy))
```

[8] Using the CBOW architecture that predicts words from contexts rather than the converse Skip-gram - see Section 2.3.

[9] For this paper, a system with four Intel Core i5 6300HQ processors at 2.3GHz with 8GB memory was used.

```
# Wx and Wy are words, Mod is a Word2Vec model and J is a
# boolean int whether or not to produce Wy's vector by
# multiplying by Wx's. Returns Kullback-Leibler divergence
# based on Wx and Wy's prediction vectors.
#
def vecmeasure((Wx,Wy), Mod, J):

    Nx = Mod.syn1neg[Mod.wv.vocab[Wx].index]
    Ny = Mod.syn1neg[Mod.wv.vocab[Wy].index]
    U = uniform(len(Nx))

    (Jx,Jy) = joinvec(Nx, Ny, J)

    Mx = softmax(Jx)
    My = softmax(Jy)

    K = measure(Mx, My, U)

    return K


# A is a real-valued array. Returns normalised array
# for use as a probability distribution.
#
def softmax(A):

    E = np.exp(A - np.max(A))
    M = E/E.sum()

    return M


# N is length. Returns uniform distribution.
#
def uniform(N):

    U = range(0,N)

    for i in U:
        U[i] = 1/N

    return U
```

```
# P and Q are a general and a specific probability
# distribution, U is the uniform distribution. Returns
# Kullback-Leibler divergence of P and Q against U
# (ignorance to general and to specific), and Q against P
# (general to specific).
#
def measure(P, Q, U):

    Kpu = kldiv(P, U)
    Kqu = kldiv(Q, U)
    Kqp = kldiv(Q, P)

    return (Kpu, Kqu, Kqp)


# P and Q are probability distributions. Returns Kullback-
# Leibler divergence.
#
def kldiv(P, Q):

    div = 0

    for i in range(0,len(P)):
        div = div + P[i] * math.log(P[i]/Q[i], 2)

    return div


# Sims is a list of tuples: (word, similarity_value).
# Returns only the words.
#
def getwords(Sims):

    Words = range(0,len(Sims))

    for i in Words:
        Words[i] = Sims[i][0]

    return Words


# Sims is a list of tuples: (word, similarity_value).
# Returns only the values.
#
def getvalues(Sims):

    Values = range(0,len(Sims))

    for i in Values:
        Values[i] = Sims[i][1]

    return Values
```

```
# Vx is a noun vector, Vy is either a compound noun vector or
# a property vector. If J is nonzero, latter is assumed:
# Returns compound vector by addition (1), multiplication
# (2), or tensor product (3).
#
def joinvec(Vx, Vy, J):

    Jx = Vx

    if J == 0:
        Jy = Vy

    elif J == 1:
        Jy = np.add(Vx, Vy)

    elif J == 2:
        Jy = np.multiply(Vx, Vy)

    else:
        Jy = np.sum(np.outer(Vx,Vy), axis=1)

    return (Jx, Jy)


# T is the number of similar words to retrieve, J is boolean
# int for whether or not to produce second word vector by
# multiplying with the first, O is offset for shifting the
# similarities list. If J is false, both lists must be offset
# by 1 to remove self-similar word, else at most the first.
# Returns incremented retrieval numbers to compensate for
# word deletion.
#
def incrementtop(T, J, O):

    Tx = T
    Ty = T

    if O == 1:
        Tx = T + 1

        if J == 0:
            Ty = T + 1

    return (Tx, Ty)
```

```
# Sx and Sy are word similarity lists; other arguments: see
# incrementtop function. Returns similarity lists shifted by
# offset.
#
def offsetlist(Sx, Sy, J, O):

    Ox = 0
    Oy = 0

    if O == 1:
        Ox = Ox + 1

        if J == 0:
            Oy = Oy + 1

    return (Sx[Ox:], Sy[Oy:])
```

Listing 3: File `genmeasure.py`

The program `genmeasure.py` is assumed to reside in or linked to one of the directories `intersective/` or `phrasal/`.

It is a called using a script; the one called `genshr.py` in Listing 4 was used to compute the similar words that were shared by the intersective and phrasal models, as described in Section 3.2.

```
from genmeasure import loadmodel, termsim


TermsP = [('woman','blackwoman'), ('woman','whitewoman'),
          ('woman','oldwoman'), ('man','blackman'), \
          ('man','whiteman'), ('man','oldman'), \
          ('village','smallvillage'), ('story','shortstory')]


TermsI = [('woman','black'), ('woman','white'), \
          ('woman','old'), ('man','white'), \
          ('man','black'), ('man','old'),
          ('village','small'), ('story','short')]


PathP = 'phrasal/modphr/word2vec_gensim'


PathI = 'intersective/modint/word2vec_gensim'
```

```
def init():

    global ModelP
    ModelP = loadmodel(PathP)

    global ModelI
    ModelI = loadmodel(PathI)


def sharesims(J):

    Len = len(TermsP)

    ResP = range(0,Len)
    ResI = range(0,Len)
    Res = range(0,Len)

    for p in ResP:
        ResP[p] = termsim(TermsP[p], ModelP, 20, 0)

    for i in ResI:
        ResI[i] = termsim(TermsI[i], ModelI, 20, J)
        SetP = set(ResP[i][1])
        SetI = set(ResI[i][1])
        Res[i] = SetP.intersection(SetI)

    return Res
```

Listing 4: File: `genshr.py`, calls `genmeasure.py` to produce shared term similarities

This script is called from the directory above `phrasal/` and `intersective/`, and loads both models (i.e. those in `phrasal/modphr/` and `intersective/modint/`.

It should be loaded into a Python shell, after which the functions are imported by entering `from genshr import init, sharesims`. After that, `init()` must be issued to load the models. Then the top 20 similar words that are shared by the phrasal and intersective models, when the intersective model combinations are produced using addition, can be retrieved by typing `S = sharesims(1)`. Shared words for combinations made by multiplication and tensor product are retrieved by changing the passed variable to 2 and 3, respectively. Shared similar words for, e.g., the collation *oldman* and the composition *old man* can then be inspected by entering `S[5]`.[10]

The following script, `genint.py` in Listing 5, resides in `intersective/` and was used to compute Kullback-Leibler divergences for the intersective model, but can also return word similarities.

[10]The output array is indexed starting from 0.

```
from genmeasure import loadmodel, termsim, vecmeasure


Terms = [('woman','black'), ('woman','white'), \
         ('woman','old'),   ('man','white'), \
         ('man','black'), ('man','old'), \
         ('village','small'), ('story','short')]


Path = 'modint/word2vec_gensim'


def init():

    global Model
    Model = loadmodel(Path)


def termsims(J):

    Res = range(0,len(Terms))

    for i in Res:
        Res[i] = termsim(Terms[i], Model, 20, J)

    return Res


def vecmeasures(J):

    Res = range(0,len(Terms))

    for i in Res:
        Res[i] = vecmeasure(Terms[i], Model, J)

    return Res
```

Listing 5: File: `genint.py`, calls `genmeasure.py` to produce Kullback-Leibler divergences and similarities

To run this script, the functions `init`[11] and `vecmeasures` are loaded by issuing `from genmeasures import init, vecmeasures`. Subsequently issuing `init()` loads the model whose path is given in the `Path` variable.

It is loaded by issuing `from genint import init, termsims, vecmeasures`, and initialised by typing `init()`. After that, `M = vecmeasures(1)` computes Kullback-Leibler divergences, with the vectors for the tuples in the `Terms` list combined using addition. Setting the variable `J` to 2 makes the combined vectors by multiplication, and using `3` produces them via the tensor product.

The results will then be in `M` as tuples of the form $(KL_{I\to G}, KL_{I\to S}, KL_{G\to S})$, that is, ignorance to general, ignorance to specific, and general to specific. For

[11] `init` must always be loaded and `init()` always run.

instance, the result for *'old man'* can be inspected by entering `M[5]` into the shell. To see the top 20 similar words for a term using vector addition, import the function `termsims`, and issue `S = termsims(1)`. For the previous example, these will then be in `S[5]`.

To run phrasal data, the `Path` variable in the script must point to the phrasal model, i.e. `Path = 'modphr/word2vec_gensim'`. `Terms` should then be set as: `Terms = [('woman','blackwoman'), ('woman','whitewoman'), ('woman', 'oldwoman'), ('man','blackman'), ('man','whiteman'), ('man', 'oldman'), ('village','smallvillage'), ('story','shortstory')]`. That is, the second component of the input tuple is now the collated phrase, as in Listing 4. Using any of the functions should be with the variable `J` set to 0, since there will be nothing to combine.

Finally, the hypernym/hyponym pairs were run using the intersective model, and in this case `Terms = [('fruit','banana'), ('building','house'), ('furniture','table'), ('vehicle','car'), ('color','green'), ('liquid','water'), ('composer','mozart'), ('country','france')]`. As above, this is run with `J` set to 0.

B Similarity listings

This appendix lists some top 10 similarities found with the intersective and phrasal models.

Table 6 gives similarities of the combinations for the intersective model for the phrases *white woman*, *old man*, and *short story* using addition, multiplication, and tensor product.

term	Addition	Multiplication	Tensor product
white woman	woman, white, girl, man, black, prostitute, mammy, brunette, female, blonde	aniu, shizumasa, tarzan, tressy, yangzom, khenpos, yugas, hadai, mythologicals, svetambaras	woman, girl, man, prostitute, person, maid, lover, seductress, herself, divorcee
old man	man, old, boy, drunkard, girl, lad, woman, beggar, handsome, woodcutter	xerces, aniu, kenders, teewinot, maharidge, enkei, chemay, baghini, enameled, tarzoo	man, woman, boy, girl, thief, drunkard, stranger, lover, person, beggar
short story	story, novella, tale, short, stories, novel, narrative, retelling, narration, novelette	dawangzhangzi, abpa, ypresian, namurian, speb, strettweg, thanetian, cryptococcal, uec, priabonian	restricted, administrate, permitted, reconfirmed, applied, nordauto, restricting, saponify, permitting, reallocated

Table 6: Similarities for intersective vector combinations

Table 7 lists similarities for the base noun, i.e. *woman*, *man*, and *story* in the intersective model, and for their collations in the phrasal model.

term	Base noun	Collation
white woman	girl, man, prostitute, person, maid, lover, seductress, herself, divorcee, maidservant	blackwoman, blackman, woman, whitewomen, whitemen, whiteman, biracial, mulatto, prostitute, divorcee
old man	woman, boy, girl, thief, drunkard, stranger, lover, person, beggar, policeman	oldwoman, man, beggar, innkeeper, imposter, drunkard, boy, impostor, ogress, woodcutter
short story	tale, stories, novel, novella, narrative, plot, plotline, tales, retelling, book	novella, novel, shortstories, novelette, novels, book, tale, anthology, memoir, poem

Table 7: Base and phrasal similarities