Contextual Compositionality Detection with External Knowledge Bases and Word Embeddings

Dongsheng Wang
Department of Computer Science
University of Copenhagen
Copenhagen, Denmark
wang@di.ku.dk

Qiuchi Li
Department of Information Engineering
University of Padua
Padova, Italy
qiuchili@dei.unipd.it

Lucas Chaves Lima
Department of Computer Science
University of Copenhagen
Copenhagen, Denmark
lcl@di.ku.dk

Jakob Grue Simonsen
Department of Computer Science
University of Copenhagen
Copenhagen, Denmark
simonsen@di.ku.dk

Christina Lioma
Department of Computer Science
University of Copenhagen
Copenhagen, Denmark
c.lioma@di.ku.dk

ABSTRACT

When the meaning of a phrase cannot be inferred from the individual meanings of its words (e.g., *hot dog*), that phrase is said to be *non-compositional.* Automatic compositionality detection in multi-word phrases is critical in any application of semantic processing, such as search engines [9]; failing to detect non-compositional phrases can hurt system effectiveness notably. Existing research treats phrases as either compositional or non-compositional in a deterministic manner. In this paper, we operationalize the viewpoint that compositionality is contextual rather than deterministic, i.e., that whether a phrase is compositional or non-compositional depends on its context. For example, the phrase “green card” is compositional when referring to a green colored card, whereas it is non-compositional when meaning permanent residence authorization. We address the challenge of detecting this type of contextual compositionality as follows: given a multi-word phrase, we enrich the word embedding representing its semantics with evidence about its global context (terms it often collocates with) as well as its local context (narratives where that phrase is used, which we call *usage scenarios*). We further extend this representation with information extracted from external knowledge bases. The resulting representation incorporates both localized context and more general usage of the phrase and allows to detect its compositionality in a non-deterministic and contextual way. Empirical evaluation of our model on a dataset of phrase compositionality[1], manually collected by crowdsourcing contextual compositionality assessments, shows that our model outperforms state-of-the-art baselines notably on detecting phrase compositionality.

[1]https://github.com/dswang2011/ImprovedRankedList/tree/master/input

CCS CONCEPTS

• **Information systems** → *Data encoding and canonicalization.*

KEYWORDS

Compositionality detection; Knowledge base; Word embedding

ACM Reference Format:
Dongsheng Wang, Qiuchi Li, Lucas Chaves Lima, Jakob Grue Simonsen, and Christina Lioma. 2019. Contextual Compositionality Detection with External Knowledge Bases and Word Embeddings. In *Companion Proceedings of the 2019 World Wide Web Conference (WWW '19 Companion), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3308560.3316584

1 INTRODUCTION

Automatic compositionality detection refers to the automatic assessment of the extent to which the meaning of a multi-word phrase is decomposable into the meanings of its constituents words and their combination. For example, while *brown dog* is a fully compositional phrase meaning a dog of brown color, *hot dog* is a non-compositional phrase denoting a type of food. Compositionality plays a vital role in word embeddings because a non-decomposable phrase should, in principle, be treated as a single word instead of a bag of word (BOW) in word embedding approaches.

A typical line of research in automatic compositionality detection is to "perturb" the input phrase by replacing one of its constituent words at a time with its synonym, and then to measure the semantic distance between the original phrase and the perturbed phrase set [8]. The larger this distance, the less compositional the original phrase. For instance, *hot dog* would be perturbed to *warm dog* and *hot canine.* The semantic distance between the original phrase and its two perturbations is high, indicating that they denote different concepts; hence *hot dog* is non-compositional. However, the phrase *brown dog* would be perturbed to *hazel dog* and *brown canine*, which have a shorter semantic distance to *brown dog*, indicating that it is compositional.

In this paper, we posit that the compositionality of a phrase is not dichotomous or deterministic, but instead varies across scenarios. For instance, *heavy metal* could refer to a dense metal that is toxic, which is compositional, but it could also be non-compositional when it refers to a genre of music. Previous work acknowledges this property of compositionality theoretically [8], but no operational models implementing this have been presented to this day.

Given a multi-word phrase as input, we reason that the phrase is used in some narrative, e.g., a query, sentence, snippet, document, etc. We refer to this narrative as *usage scenario* of the phrase. We combine evidence extracted from this usage scenario of the phrase with the global context (frequently co-occurring terms) of the phrase and use this to enrich the word embedding representation of the phrase. We linearly combine the weights of the tokens that are obtained from the usage scenario and the global context. We further extend this representation with information extracted from external knowledge bases.

We evaluate our model on a large dataset of phrases which are labeled as per five degrees of compositionality under various usage scenarios. We find that our model outperforms state-of-the-art baselines notably on identifying phrase compositionality. Our contributions are as follows:

- A novel model that detects phrase compositionality under different contexts and that outperforms the state of the art performance in the area.
- A benchmarking dataset of contextualized compositionality detection, that we make publicly available to the community.

2 RELATED WORK ON AUTOMATIC COMPOSITIONALITY DETECTION

Compositionality detection mainly focuses on the semantic distance or similarity calculation between a given phrase and its component words or its perturbations under a corpus or dictionary. Earlier approaches mostly estimate the similarity between the original phrase and its component words. For example, Baldwin et al. [1], and Katz and Giesbrecht [6] employ Latent Semantic Analysis (LSA) to calculate the semantic similarity (and hence to measure compositionality). Venkatapathy and Joshi [16] extended this by adding collocation features, e.g., phrase frequency, point-wise mutual information, extracted from the British National Corpus.

More recent work estimates the similarity between a phrase and perturbed versions of that phrase where the words are replaced, one at a time, by their synonyms. For instance, Kiela and Clark [7] compute the semantic distance between a phrase and its perturbation, using cosine similarity, which measures a phrase weight by pointwise-multiplication vectors of its terms. Lioma et al. [10] calculate the semantic distance with Kullback-Leibler divergence based on a language model; and, in subsequent work, Lioma et al. [8] represent the original phrase and its perturbations as ranked lists, and measure their correlation or distance.

A promising line of work uses word embeddings and deep artificial neural networks for compositionality detection. Salehi [15] employs the word-based skip-gram model for learning non-compositional phrases, treating phrases as individual tokens with vectorial composition functions. Hashimoto and Tsuruoka [5] adopt syntactic features including word index, frequency and PMI of a phrase and its components words to learn the embeddings. Yazdani et al. [17] utilize a polynomial projection function and deep artificial neural networks to learn the semantic composition and detect non-compositional phrases like those that stand out as outliers, assuming that the majority are compositional.

Closer to our work, Salehi el. [14] use Wiktionary and utilize the definition, synonyms, and translations of Wiktionary to detect non-compositional components. Specifically, they analyze the lexical overlap between the definition of a phrase and its component words to measure compositionality. They assume that multi-word phrases are included in Wiktionary, while there is no guarantee for perfect coverage of the dictionary. Unlike this approach, we use Wiktionary together with DBPedia as a structured knowledge base to represent the contextual semantics of phrases.

To our knowledge, no prior work has operationalized the compositionality of a phrase as contextual.

3 OUR CONTEXTUAL REPRESENTATION MODEL FOR COMPOSITIONALITY DETECTION

3.1 Problem Formulation

Given an input phrase p and its accompanying usage scenario s, the aim is to compute the compositionality score $Score(p)$ of phrase p with respect to usage scenario s. We follow the substitution-based line of work [7], which (a) generates perturbations of the input phrase p by substituting one word at a time with its synonym, (b) builds a semantic representation (a vector of its co-occurring terms) separately for the input phrase p and each perturbed phrase, and (c) uses the distance between the vectors of the input phrase and its perturbations to approximate the compositionality of the input phrase: the higher the distance, the less compositional the input phrase. This substitution-based line of work does not accommodate the usage scenario of the input phrase or its perturbations. The vectors of co-occurring terms are computed on one corpus, and hence these vectors represent the *global* distributional semantics of the input phrase and its perturbations. We extend this line of work by incorporating the *local* usage scenario of the phrase and its perturbations. We furthermore enrich these representations using external knowledge bases KBs. We describe this next.

Figure 1 shows how the phrase and scenario are fed into the external corpus and knowledge base in a sequential manner in the architecture, which we refer to as contextual representation model (CRM).

3.2 Building Global and Local Phrase Context

Global Phrase Context. In Natural Language Processing (NLP), the distributional semantics of an input word are computed by fixing a natural number n and, for each occurrence of a word in some corpus, finding the n words occurring immediately before, and n words occurring immediately after each occurrence of the input word (called *context window*). If there is a total of N context windows for a word, its distributional semantics in vector form can be calculated by using all these N windows. Because this is a global representation of the word's distributional semantics across the whole corpus, the vector is called a *globalized* vector. A general word embedding (e.g., word2vec) is comparable to such global context. Concretized representations of this globalized vector can be

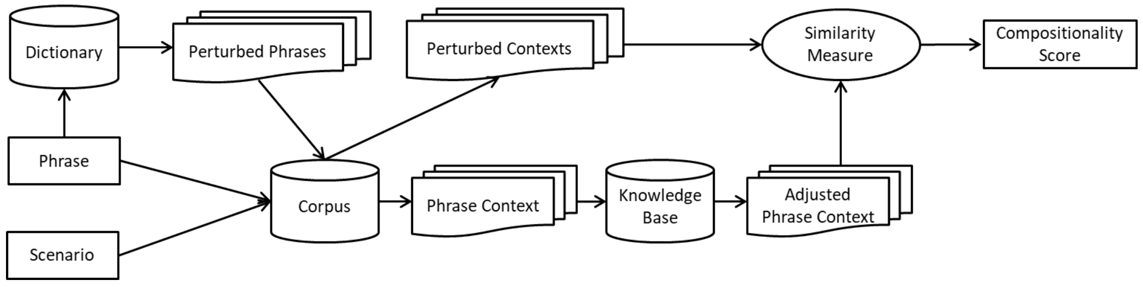

Figure 1: The diagram for the CRM sequential framework.

calculated with, e.g. ranked lists or word embeddings, as described in section 4.1.

Local Phrase Context. We aim to incorporate a representation of the local usage scenario of the input phrase (*local phrase context*) into the above described global phrase context. This representation will not be in the vector directly, because usage scenarios are typically extremely short (in terms of words), which may strongly bias the contextual representation of the phrase. Therefore, we rank all the global context windows of the phrase according to their similarity to the usage scenario of the phrase, and we select the top K most similar context windows to the usage scenario. These top K context windows are used to build the local usage scenario context representation of the phrase. Then, we linearly combine the global representation of the phrase (i.e., by taking all N context windows) and the local usage scenario representation of the phrase (i.e., the top K context windows) to acquire the *localized phrase context*. The ranking score is the similarity between the usage scenario s and a window W_i, i.e. $\text{sim}_i = \text{similarity}(W_i, s) \in [0, 1]$. The details of how the similarity score is computed are introduced in Section 4.

In the above, the value of K is determined by the length of the usage scenario as follows:

$$K = \max\left(\frac{N}{2^{\text{length}(s)}}, M\right) \tag{1}$$

where N is the total number of context windows that contain phrase p in the corpus; length(s) is the number of words in usage scenario s excluding the original phrase; and M is a threshold. We explain these next.

We posit that $2^{\text{length}(s)}$ indicates the degree of shrinking: the longer the usage scenario is, the smaller number of windows will be shrunk. The reason behind this is that the longer the usage scenario is, the more semantics it contains and subsequently fewer specific windows we are supposed to be capable of locating on. For instance, if the usage scenario is empty with $\text{length}(s) = 0$, then it returns the entire N windows of p with no shrinking performed; if the usage scenario has three words with $2^{length(s)} = 8$, it only collects the top $\frac{1}{8}$ of all the windows ($K = \frac{N}{8}$). Note that K depends on the usage scenario length, and is not fixed as a threshold of the similarity values. Since the similarity values may vary drastically in between $[0, 1]$ for different usage scenarios, we argue that our method is more robust to such variations. Furthermore, an empirical threshold of M is introduced to guarantee at least M windows will be selected anyhow. To this end, the localized usage scenario context of a phrase is given by:

$$C(p, s) = \alpha \frac{\sum_{i=1}^{N} \mathcal{R}(W_i)}{N} + (1 - \alpha) \frac{\sum_{j=1}^{K} \mathcal{R}(W_j)}{K} \tag{2}$$

where a is a weight parameter between 0-1 indicating the weight of global vector, and the remaining of $1 - a$ corresponds to the contribution of the localized vector; $\mathcal{R}$ denotes the semantic representation for W. In this paper, we represent a phrase as a ranked list of words and word embedding, so the same symbol $\mathcal{R}$ is adopted to denote both. The approach to calculate $\mathcal{R}$ is described in section 4.1.

3.3 Enriching Global and Local Phrase Context with Knowledge Bases

We enrich the global and local context representations of the input phrase with information extracted from external knowledge bases. We describe this next.

We reason that the corpus used to extract the global and local contexts has good coverage of various but not all possible usage scenarios of the phrase. A knowledge base is expected to contain more comprehensive, declarative information about the phrase, e.g., entities and phrase senses with categorized information. We, therefore, enrich the global and local phrase contexts extracted from the corpus with phrase information extracted from external knowledge bases.

Given an input phrase, we collect all candidate senses and entities (uniformly referred to as *candidates* in this paper) by searching the following properties (and associated values) from the knowledge bases: the properties *dbpedia:redirects*, *dbpedia:disambiguation* and their propagation relation with *dbpedia:name* and *rdfs:label*. The associated resources in these retrieved triples result in a set of candidates. Then, for each candidate, the values of *rdfs:label*, *dbpedia:abstract* and *rdf:type* are concatenated as the context for that candidate, excluding the title (which is mostly the phrase name). We also use the interface[2] to retrieve senses from Wiktionary and merge them into the same candidate set for that given input.

Most phrases only contain a limited number of candidates, and different candidates of the same phrase can have entirely different meanings or be distinct entities. We hence investigate a sequential way to incorporate the knowledge base into the phrase contextual representation, as follows. First, the phrase is fed into the knowledge base to find all candidate articles. Then, the candidates are ranked

[2]https://dkpro.github.io/dkpro-jwktl/

according to how similar they are to the localized phrase context. Those with similarity values above a certain threshold are identified as the matched candidates, denoted as $\{D_i, sim_i\}_{i=1}^{n}$, where D_i and sim_i refer to the i^{th} matched candidate with similarity value $\text{sim}_i \in [0, 1]$. A linear combination of the localized phrase context and the candidate articles is then conducted to compute the adjusted phrase context as follows:

$$C(p, D) = \lambda C(p, s) + (1 - \lambda) \sum_{i=1}^{n} w_i \mathcal{R}(D_i), \tag{3}$$

where $w_i = \frac{\text{sim}_i}{\sum_{i=1}^{n} \text{sim}_i}$ is the normalized similarity score for the i^{th} candidate article D_i while $\mathcal{R}(D_i)$ denotes the semantic representation for D_i. Since KB contains well-defined knowledge of words, we use a weighted sum of the matched candidates, instead of a simple average of matched contexts in the text corpus.

The knowledge base we employ consists of DBpedia, Wiktionary, and Wordnet. DBpedia is constructed by extracting structured information from Wikipedia. The English version of the DBpedia contains 4.58 million entries, of which 4.22 million are classified and managed under one consistent ontology. Wiktionary is a multilingual, web-based, freely available dictionary, thesaurus and phrase book, designed as the lexical companion to Wikipedia. Volunteers collaboratively construct Wiktionary, so there are no specialized qualifications necessary.

3.4 Non-linear combination

This section represents an approach of non-linear combination as a companion to the linear combination approach introduced in section 3.2 and 3.3. In addition to the weight parameter oriented design of the linear combination, we also employ a non-linear sigmoid function in RNNs (recurrent neural networks), which resolves the arrangement of the combining order for context inputs. In other words, RNNs take into consideration the feedback from the previous context vector back and forth, leading to numerous applications [3, 4]. Specifically, we train a neural network model using the Keras library to identify the compositionality label of each phrase. We encode the semantics adopting pre-trained word embeddings - word2vec [11] as word representations, a recurrent neural network with LSTM cells as the model, and cross-entropy as the loss function. As an optimizer, we utilize Adam optimizer for training the model.

In a realistic scenario (also represented in our dataset) there are fewer non-compositional than compositional phrases. This situation resembles the class imbalance issue which happens when one class (or label) is represented by most of the examples while the other one is represented just by a few. Therefore, we adopt re-sampling strategies to tackle this problem.

3.5 Compositionality Detection

Here we introduce our proposed method for compositionality detection with Algorithm 1. Given a phrase p of length l and its usage scenario s in a large corpus $Corp$, we compute its compositionality score through the following steps:

```
Input: Phrase $p$ with length $l$
Input: Usage scenario $s$ for $p$
Input: Corpus $Corp$
Input: Knowledge Base - DBPedia
Input: Similarity threshold - $thred$
Output: Compositionality score $comp(p)$
 1: Set of perturbed phrase $S(\hat{p}) \leftarrow \varnothing$
 2: Find synonym $\hat{t}$ of each term $t \in p$
 3: for each $\hat{t}$ do
 4:     Perturbed phrase $\hat{p} \leftarrow \{ \hat{t}$ , $l$-1 original terms $t_o \}$
 5:     Update perturbed phrase set $S(\hat{p}) \leftarrow S(\hat{p}) \cup \hat{p}$
 6: end for
 7: for phrase $p' \in \{p \cup S(\hat{p})\}$ do
 8:     $C(p') \leftarrow$ get context terms from localized phrase context
        from $Corp$, smooth with Eq. 1 if it has scenario
 9: end for
10: Find $n$ candidate articles $D_i$ from KB where
    $sim_i = similarity(s, D_i) > thred$
11: $\mathcal{R}(D_i) \leftarrow$ semantic representation of $D_i$
12: $C(p, D) \leftarrow$ linear combined context
    $\lambda C(p) + (1-\lambda) \sum_{i=1}^{n} \frac{sim_i}{\sum_{i=1}^{n} sim_i} \mathcal{R}(D_i)$
13: $Q(L_{\hat{p}}) \leftarrow \varnothing$
14: for each perturbed phrase $\hat{p} \in S(\hat{p})$ do
15:     $Q(L_{\hat{p}}) \leftarrow Q(L_{\hat{p}}) \cup C(\hat{p})$
16: end for
17: return $\frac{1}{|Q(L_{\hat{p}})|} \sum_{C(\hat{p}) \in Q(L_{\hat{p}})} Similarity(C(\hat{p}), C(p, D))$
```

Algorithm 1: Algorithm of contextual compositionality detection

(1) Obtain localized phrase context through Eq. 1 and 2. The usage scenario of a phrase is the critical information, and the idea behind this step is to smooth the scenario context representation with the original phrase representation, as shown in line 8 in Algorithm 1.
(2) Adjust phrase context with a knowledge base. The knowledge base is fed to adjust the localized phrase context where we adopt Eq. 3 to encode the information (from line 10 to line 12).
(3) Obtain a perturbed phrase set. For each term in the phrase, we find its synonyms in WordNet. We then generate the set of perturbed phrases $S(p)$ as: S(p) = {$\hat{p}$ where $\hat{p}$ = l-1 terms of p plus a synonym of the remaining term of p}, from line 3 to line 6.
(4) Construct a perturbation representation set. For each perturbed phrase $\hat{p}$ in S(p), the corresponding representation $C(\hat{p})$ is composed of all windows of $\hat{p}$ from the corpus, and is added to the perturbation list $Q(L_{\hat{p}})$ from line 14 to line 16. Note that we do not combine context from KB for perturbations.
(5) Compute the compositionality score for the input phrase, shown in line 17, using the following equation:

$$score(p) = \frac{\sum_{\hat{p} \in S(p)} sim(C(p, D), C(\hat{p}))}{|S(p)|} \tag{4}$$

4 IMPLEMENTATION

4.1 Semantic Representation

The semantic representation of a context (a phrase, a context window or a candidate content), i.e., $\mathcal{R}(\cdot)$ is concretized as either a ranked list or a word embedding. For the ranked list model, we calculate the TF-IDF as weight for all the tokens, rank them according to the weight, resulting in a ranked list of those tokens as the localized contextual representation. For the word embedding model, we use existing pre-trained word vectors - Glove [12], and represent the vector with the average of all tokens. The corpus we employ in our experiment is ClueWeb12-B13, a subset of some 50 million pages of ClueWeb12-Full dataset[3].

These two contextual representations lead to two different compositionality scores for the same model. We apply suffixes "-word embedding" or "-ranked list" in order to distinguish the way the contextual representation is computed, resulting in two distinct models, namely *CRM word embedding* and *CRM ranked list*.

4.2 Similarity Measure

In this study, we are faced with the problem of computing the similarity value between two context vectors. Here, we consider two types of similarity measures to achieve this purpose: *cosine similarity* and *Pearson correlation coefficient*.

One of the most commonly used similarity measures, *cosine similarity*, computes the cosine value of the angle between the two vectors of the same length. For two vectors $\vec{a} = [a_1, a_2, .., a_n]$ and $\vec{b} = [b_1, b_2, .., b_n]$, their cosine similarity cossim(a,b) is given below:

$$cossim(a, b) = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}} \tag{5}$$

The *Pearson correlation coefficient* computes the degree of correlation between two variables, each having a set of observed values. Suppose two variables X and Y are associated with two set of values $\{X_1, X_2, ..., X_n\}$ and $\{Y_1, Y_2, ..., Y_n\}$ respectively. The Pearson correlation coefficient r can, therefore, be computed as follows:

$$r = \frac{\sum_{i=1}^{n} (X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n} (X_i - \overline{X})^2}\sqrt{\sum_{i=1}^{n} (Y_i - \overline{Y})^2}} \tag{6}$$

where $\overline{X}$ and $\overline{Y}$ denote the average of X and Y respectively.

4.3 Perturbation

Here, we introduce the process to obtain the perturbations of a phrase p with length l. First, we get the synonyms for each word in the phrase. Then, we construct the whole perturbation set, which contains all phrases composed of $l - 1$ words in p and a synonym of the remaining word. Suppose the i^{th} word has n_i synonyms, then the perturbation set contains $\sum_{i=1}^{l} n_i$ perturbed phrases. We then prune the perturbation set by filtering out the rare perturbed phrases in the text corpus. Basically, we compute the occurrence frequency of all perturbed phrases and pick the perturbed phrases with top K frequency values. In our study, we set K to be 7, which is derived from empirical observation of the data. Then, the final perturbation set contains 7 perturbed phrases in our study.

[3]https://lemurproject.org/clueweb12/

4.4 Parameter Settings

Contextual Windows setting: We set $window = 20$, which means it scans the previous 20 and subsequent 20 words of that phrase, with a sum of 40 words for each window. In Equation 1, we set $M = 10$, and the base 2 can also be parameter-free which can be changed into 2,3,4,etc., to increase the localization level.

Knowledge base threshold: As for the threshold of KB candidates, we set the threshold of similarity value between localized context and KB candidates as 0.5 to filter out those candidates with similarity less than 0.5.

Ranked list length: In line with the work [8], we set a maximum length of the ranked list as 1000, which means that we rank the tokens according to their TFIDF weight, and the tokens after the position of 1000 would be pruned.

Training and testing: As we are working with imbalanced data, we use a random oversampling strategy. We split our data in a stratified fashion into 65% for training, 15% for validation, and 20% for testing. The re-sampling is be done after splitting the data into training and test, and only on the training data, i.e., none of the information in the test data is being used to create synthetic observations.

5 EXPERIMENT AND VALIDATION

In this section, we evaluate the effectiveness of the model presented in Section 3. Section 5.1 introduces the dataset and Section 5.2 presents the results achieved by our model.

5.1 Crowdsourcing data

We employ a dataset that consists of 1042 phrases that are noun-noun 2-term phrases [2]. In this dataset, each phrase was assessed four times using a binary scale (compositional or non-compositional). However, these phrases are assessed with a deterministic label, meaning that no scenario or context was given, and the degree of compositionality may not always be binary [13]. Therefore, we extend the dataset into a new version where each phrase is enriched with one or two scenarios if possible, by taking advantage of a crowdsourcing website - Figure Eight [4], and we use a graded level of compositionality. In Table 2 we summarize the dataset statistics.

We divided the assessment into two stages: for the first stage, the trustful assessors, with level 3 (highest in Figure Eight), are required to understand the various meanings of a phrase, and, if possible, create two scenarios for the same phrase. From these two scenarios, one should be compositional or as compositional as possible, and the other non-compositional or as non-compositional as possible. If the phrase can only be compositional or only be non-compositional, then they create one scenario for it. For the second stage, the assessors are required to assess the compositionality of phrases within different scenarios with one of the five graded labels: compositional, mostly-compositional, ambiguous to judge, mostly non-compositional, and non-compositional. Note that, for the first

[4]https://www.figure-eight.com/

Unsupervised Methods	ρ	α, λ
Baseline: Ranked list [8]	0.131	na
Baseline: Word Embedding	0.147	na
CRM ranked list	0.209	0.1,0.5
CRM Word Embedding	0.375	0.9, 0.1
Supervised Methods (20% Testing)	ρ	
RNN (LSTM cells)	0.176	na
RNN (LSTM cells) CRM	0.324	na

Table 1: Results of different compositionality detection methods; *na* denotes *not applicable.*

No. Non-Compositional	43 (3.6%)
No. Mostly Non-Compositional	145 (12.1%)
No. Ambiguous Phrases	126 (10.5%)
No. Mostly Compositional	141 (12.0%)
No. Compositional	739 (61.8%)
Unique number of Phrases	1042
No. of scenarios	1194
Average number of context by Phrase	1.146

Table 2: Summary of dataset statistics.

stage, the two scenarios of a phrase are not necessarily of two extreme polarities.

5.2 Performance and Validation

Two linear combination parameters influence the performance of our model: the combination weight α (in Eq. 2) between the vectors of a phrase and its scenario, resulting in a localized phrase context, and λ (in Eq. 3) between the localized context and knowledge base. The impacts of these two parameters on the final performance are visualized in Figure 2 and 3, corresponding to the word embedding-based and ranked list-based contextual representation respectively. α and λ denote the x and y coordinates. The colors indicate the performance, which is the correlation between the ground truth labels and the predicted labels of our models ranging from -1 to 1. The performance values are colored ranging from red to blue, representing the lowest performance to the highest.

As shown in Figure 2, the performance is negatively correlated with α while positively correlated with λ. This indicates that reducing the relative importance of localized context (right direction on x-coordinate) while enhancing the influence of knowledge base (bottom direction on y-coordinate) can improve the performance for word embedding based contextual representation. In contrast, as shown in Figure 3, if we ignore the *0* column, α is negatively correlated with the performance while λ does not have an apparent influence on the performance. This indicates that attaching higher importance to the localized context (left direction on x-coordinate) can improve the performance for the ranked list based contextual representation, while the adoption of knowledge base does not have an apparent influence to the overall performance. The first *0* column, which is shown more like an outlier, indicates that the existence of a vector of the original phrase is necessary. In other words, localized context would have relatively poor performance.

As summarized in Table 1, the performance improved from 0.147 to 0.375 for CRMs based on word embeddings (the best); from 0.131 to 0.209 for CRMs based on ranked list; and 0.176 to 0.324 for CRM based on RNN. For the word embedding based contextual representation model, relying more on the knowledge base while keeping the scenario to limited importance will lead to a high-performed model; for the ranked list based contextual representation model, on the other hand, adequately high adoption of localized context can lead to improved performance. The reason behind this can be that the knowledge base contains relatively trimmed but well-categorized information, therefore, the word embedding model can take full use of this text as informative vectors. In contrast, ranked lists, depending on tokens, work better on a large-scale corpus where they induce a large number of context windows. However, the knowledge base contains a limited number of tokens that may have little contribution to the final representation. Even though we can tune the weight of tokens from a knowledge base, it still can have limited influence in comparison to the long ranked list, which can be as long as 1000 tokens in our experiment.

For the non-linear combination where we employed the sigmoid function in RNNs, the CRM based on RNN still beats the original RNN. However, the performance is still lower than the unsupervised approaches.

6 CONCLUSION

We developed a novel method for compositionality detection where the compositionality of a phrase is contextual rather than static. Instead of considering an isolated phrase as input, we assume a phrase and its usage scenario (e.g., a query, snippet, sentence, etc.) as input, and we model a joint semantic representation of these by combining distributional semantics extracted from a corpus and additional evidence extracted from an external structured knowledge base.

Our resulting model uses word embeddings to detect compositionality, more accurately than the related state of the art. Our experiments show that for word embeddings, the usage of knowledge bases can lead to notable performance improvements.

In the future, we plan to evaluate our model on further datasets and compositionality detection scenario, e.g., Verbal Phraseological Units (VPUs).

ACKNOWLEDGEMENT

This work is supported by the Quantum Access and Retrieval Theory (QUARTZ) project, which has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 721321.

REFERENCES

[1] Timothy Baldwin, Colin Bannard, Takaaki Tanaka, and Dominic Widdows. 2003. An empirical model of multiword expression decomposability. In *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment-Volume 18*. Association for Computational Linguistics, 89–96.

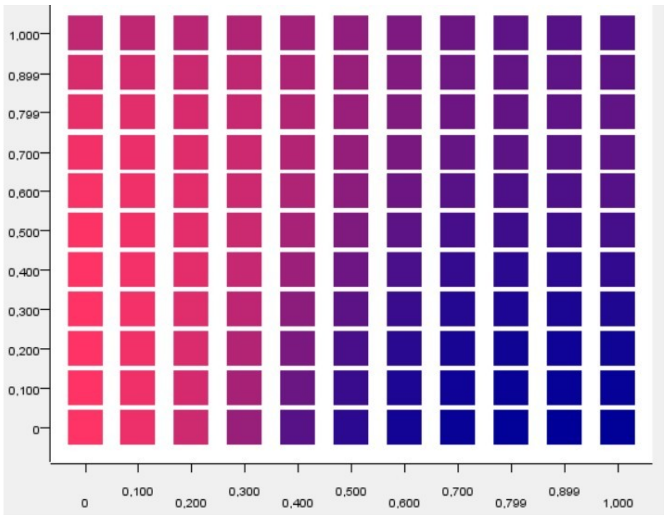

Figure 2: The grid search for Word embedding based Contextual representation. x-coordinate is α for controlling localized context and λ stands for y-coordinate, controlling the KB combining weight. Deeper blue represents higher performance whereas red indicates the opposite.

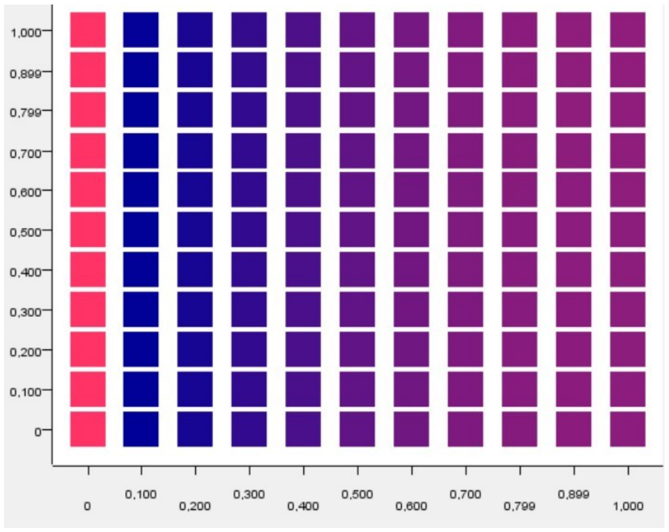

Figure 3: The grid search for ranked list based Contextual representation. x-coordinate is α for controlling localized context and y-coordinate stands for λ, controlling the KB combining weight. Deeper blue represents higher performance whereas red indicates the opposite.

[2] Meghdad Farahmand, Aaron Smith, and Joakim Nivre. 2015. A Multiword Expression Dataset: Annotating Non-Compositionality and Conventionalization for English Noun Compounds. In *Proceedings of the 11th Workshop on Multiword Expressions (MWE-NAACL 2015)*. Association for Computational Linguistics.

[3] Christian Hansen, Casper Hansen, Stephen Alstrup, Jakob Grue Simonsen, and Christina Lioma. [n. d.]. Neural Speed Reading with Structural-Jump-LSTM. In *Proceedings of the 2019 International Conference on Learning Representations, ICLR 2019*.

[4] Christian Hansen, Casper Hansen, Stephen Alstrup, Jakob Grue Simonsen, and Christina Lioma. 2019. Modelling Sequential Music Track Skips using a Multi-RNN Approach. In *Proceedings of the 2019 International Conference on Web Search and Data Mining, WSDM 2019, Sequential Skip Prediction Challenge*. in press.

[5] Kazuma Hashimoto and Yoshimasa Tsuruoka. 2016. Adaptive joint learning of compositional and non-compositional phrase embeddings. *arXiv preprint arXiv:1603.06067* (2016).

[6] Graham Katz and Eugenie Giesbrecht. 2006. Automatic identification of non-compositional multi-word expressions using latent semantic analysis. In *Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*. Association for Computational Linguistics, 12–19.

[7] Douwe Kiela and Stephen Clark. 2013. Detecting Compositionality of Multi-Word Expressions using Nearest Neighbours in Vector Space Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 1427–1432. http://aclweb.org/anthology/D13-1147

[8] Christina Lioma and Niels Dalum Hansen. 2017. A Study of Metrics of Distance and Correlation Between Ranked Lists for Compositionality Detection. *Cogn. Syst. Res.* 44, C (Aug. 2017), 40–49. https://doi.org/10.1016/j.cogsys.2017.03.001

[9] Christina Lioma, Birger Larsen, and Peter Ingwersen. 2018. To Phrase or Not to Phrase – Impact of User versus System Term Dependence Upon Retrieval. *Data and Information Management* 2, 1 (2018), 1–14.

[10] Christina Lioma, Jakob Grue Simonsen, Birger Larsen, and Niels Dalum Hansen. 2015. Non-compositional term dependence for information retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 595–604.

[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[12] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[13] Siva Reddy, Diana McCarthy, and Suresh Manandhar. 2011. An Empirical Study on Compositionality in Compound Nouns. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP-11)*. Chiang Mai, Thailand. http://aclweb.org/anthology-new/I/I11/I11-1024.pdf

[14] Bahar Salehi, Paul Cook, and Timothy Baldwin. 2014. Detecting non-compositional MWE components using Wiktionary. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1792–1797.

[15] Bahar Salehi, Paul Cook, and Timothy Baldwin. 2015. A word embedding approach to predicting the compositionality of multiword expressions. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 977–983.

[16] Sriram Venkatapathy and Aravind K Joshi. 2005. Measuring the relative compositionality of verb-noun (VN) collocations by integrating features. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 899–906.

[17] Majid Yazdani, Meghdad Farahmand, and James Henderson. 2015. Learning semantic composition to detect non-compositionality of multiword expressions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 1733–1742.