Assignment3 – Report

Dorin Keshales – 313298424

1.

a. Thresholds:

- Filter words (in the lemma form) that occur less than 100 times in the corpus, in order to compute similarities.
- Filter words (appeared as context of a target word) that occur less than 75 times in the corpus.
- Limit to 100 most common contexts per word .
- Filter function words using the following set of POS tags: `{",", ".", "(", ")", "``", "''", ":", "$", 'IN', 'PRP', 'PRP$', 'WP', 'WP$', 'DT', 'WDT', 'CC', 'CD', 'PDT', 'Particle', 'UH', 'TO', 'EX', 'LS', 'MD', 'POS'}` and set of special function words whose POS tags are different.
- I used the POS tag `'IN'` in the dependency co-occurrence type in order to distinguish between the special case of prepositions to other function words connected with a dependency edge to the content word.
- Applied PPMI concept – turning negative PMI scores into 0 and later ignored these 0 PMI scores.

b. Number of words considered for similarity: 7984

c. The number of features considered in my computation for each co- occurrence type (rows in the matrix):

Co-occurrence Type	*Number of features*
Sentence	*9087*
Window	*9569*
Dependency	117608

2. 2^{nd} order similarity:

20 most similar words, for each of the target words, for each co-occurrence type , ordered by similarity in descending order is in **Appendix A**.

Conclusions from examining these lists

The first co-occurrence type is a sentence-sized window (also referenced in **Appendix A**'s tables as 'Sentence Co-occurrence'), characterized with capturing words that are topically-related to the target word. That is, this co-occurrence type induces topical similarities. Practically, most of the words in the lists under this co-occurrence type are topically-related to the target word, while the rest of the words are sister-terms of the target word. That is, words that, in terms of their reference, are at the same level in the hierarchy, i.e. have exactly the same hypernyms. For example, you can see that in the similarities table (with my manual similarity judgments) for the target word 'car' on page 4, the words marked with '+' in the 'top' column (as an abbreviation of topical related), which is next to the 'Sentence' column, are words that are topically-related to the word 'car' and the words marked with '-' are sister-terms of the word 'car', for example the word motorcycle. You can also check the similarities table of the target word 'piano' (on page 6) in which all words in the 'top' column(next to the 'Sentence' column) are marked with '+', i.e. all are topically-related to the word 'piano'. In addition, words that appear only on the 'Sentence Co-occurrence' list and not on the other list are most certainly topically-related words. For example, for the target word 'car ' the words engine, wheel, chassis, bmw, gt and crash and for the target word 'piano' the words composition, tenor, trio and pianist .

The second co-occurrence is a window of size of k=2 (also referenced in **Appendix A**'s tables as 'Window Co-occurrence'), i.e. the window obtained from taking 2 words before the target word and 2 words after it. This co-occurrence is mostly characterized by capturing words that are in the same semantic class as the target word, since it is a narrow window compared to the previous co-occurrence type that is considered a wide window. This insight makes sense since a narrow window catches the words surround the target word which are usually also the words that are syntactically directly related to it, but this co-occurrence type manages to catch some topically-related words as well. For example , if you take a look at the similarities table (with my manual similarity judgements) of the target word 'car' on page 4 (or check the similarities table of the word 'piano' on page 6), you can see that the words marked with '+' in the 'sem' column (as an abbreviation of semantically-related), which is next to the 'Window' column, are words that are in the same semantic class as the word 'car' and the words marked with '+' in the 'top' column (the column to the left of the 'sem' column) are topically-related to the target word 'car'. Some of the words in the 'Window' column are both topically and the semantically related to the target word 'car'. Basically, this co-occurrence type is something in the middle between the first co-occurrence and the third co-occurrence, but tends to have more similar results to the 3rd co-occurrence, i.e. words that are semantically related to the target word.

The third co-occurrence is syntactic relations based on dependency structure (also referenced in **Appendix A**'s tables as 'Dependency Co-occurrence'). The features of this co-occurrence are words that are connected to the target word by a dependency edge. This co-occurrence is characterized by capturing words that are in the same semantic class as the target word. That is, this co-occurrence type induces functional similarities – words that share the same semantic type and cohyponyms. For example , if you take a look in the similarities table (with my manual similarity judgments) of the target word 'car' on page 4 (or check the similarities table of the target word 'piano' on page 6), you can see that the words marked with '+' in the 'sem' column (as an abbreviation of semantically-related), which is next to the 'Dependency' column, are in the same semantic class as the word 'car'. Additionally, Words that appear in the 'Dependency' list and not in the other lists are most certainly semantically related to the target word, i.e. are in the same semantic class as the target word. For example, for the target word 'car' the words horse, aircraft, plane, locomotive, yacht and van and for the target word 'piano' the words drum and organ .

3. 1st order similarity:

20 top context attributes for each of the target words, for each of the 3 co-occurrence types, ordered by descending order of attributes with highest PMI values in the target word's vector, in **Appendix B**.

Short qualitative comparison between the 2nd order lists and the 1st order lists

First-order context vectors record directly observable features of a context, whilst second-order context vectors aggregate vectors themselves associated to the directly observable features of the context. In other words, the main difference between the 2nd order lists and the 1st order lists is that the words that appear in the 1st order lists are words that have appeared multiple times with the target word in the same context. whereas the words in second-order lists are words that relate to other words in the dictionary in a manner similar to that of the target word. That is, these words and the target word have similar features.

The distinctions I have made earlier for each of the co-occurrences in the 2nd order similarity regarding the types of similarities that each co-occurrence is more likely to capture, are:

- 'Sentence Co-occurrence' is more likely to capture topical similarity.
- 'Dependency Co -occurrence' is more likely to capture semantic-similarity.
- 'Window Co-occurrence' can capture both topical and semantic similarities but is more likely to capture semantic similarities.

In 1st order similarity:

- The 'Sentence Co-occurrence' list contains words that appeared in the context in which the word 'car' appeared. The more a word have appeared next to the target word or in context surrounding the target word, it is more likely for it to appear in the top20 list and even rank quite high in it.
- The 'Window Co-occurrence' list contains words that appeared within a window of two words on each side of the target word. In a same manner as before, the more windows a word

appeared in as a context of the target word, it is more likely for the word to appear in the top20 list and even rank quite high in it.

- The 'Dependency Co-occurrence' captures direct dependencies to the target word, such dependencies might not be captured with a narrow or wide window, i.e. by using the first two co-occurrences, since some words have no close connection of meaning with the target word when they appear in the context of the target word, but these words can be captured by syntactic dependencies.

For example, I've marked in red words in the 1st order similarity tables of the target words 'car' and 'piano' (In **Appendix B** on pages 27 and 34, respectively) from which it can actually be seen that these words were drawn from the context of the sentence.

4. MAP

+-- car --+

Sentence	top	sem	Window	top	sem	Dependency	top	sem
drive	+	-	driver	+	-	vehicle	+	+
driver	+	-	truck	+	+	truck	+	+
truck	+	+	motor	+	-	driver	+	-
vehicle	+	+	drive	+	-	motorcycle	-	+
motor	+	-	vehicle	+	+	racing	+	-
ford	+	-	racing	+	-	station	-	-
automobile	+	+	ford	+	-	locomotive	-	+
race	+	-	formula	+	-	automobile	+	+
auto	+	+	race	+	-	horse	-	+
formula	+	-	lap	-	-	motor	+	-
racing	+	-	motorcycle	-	+	traffic	+	-
toyota	+	-	automobile	+	+	aircraft	-	+
engine	+	-	bus	+	+	stock	-	-
motorcycle	-	+	bicycle	-	+	auto	+	+
wheel	+	-	stock	-	-	item	-	-
chassis	+	-	nascar	+	-	cyclist	-	-
bmw	+	-	traffic	+	-	plane	-	+
nascar	+	-	carriage	-	+	yacht	-	+
gt	+	-	toyota	+	-	van	+	+
crash	+	-	trailer	+	+	model	+	-

Topically related

N = # unique_topical_identified = 19 + 3 + 2 = 24

(car, Sentence)

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
rel	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1

AP(car, Sentence) = (1/1 + 2/2 + 3/3 + 4/4 + 5/5 + 6/6 + 7/7 + 8/8 + 9/9 + 10/10 + 11/11 + 12/12 + 13/13 + 0 + 14/15 + 15/16 + 16/17 + 17/18 + 18/19 + 19/20) / N = **0.777**

(car, Window)

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
rel	1	1	1	1	1	1	1	1	1	0	0	1	1	0	0	1	1	0	1	1

AP(car, Window) = (1/1 + 2/2 + 3/3 + 4/4 + 5/5 + 6/6 + 7/7 + 8/8 + 9/9 + 0 + 0 + 10/12 + 11/13 + 0 + 0 + 12/16 + 13/17 + 0 + 14/19 + 15/20) / N = **0.57**

(car, Dependency)

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
rel	1	1	1	0	1	0	0	1	0	1	1	0	0	1	0	0	0	0	1	1

AP(car, Dependency) = (1/1 + 2/2 + 3/3 + 0 + 4/5 + 0 + 0 + 5/8 + 0 + 6/10 + 7/11 + 0 + 0 + 8/14 + 0 + 0 + 0 + 0 + 9/19 + 10/20) / N = **0.3**

Same semantic class

N = # unique_ semantic _identified = 5 + 4 + 6 = 15

(car, Sentence)

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
rel	0	0	1	1	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0

AP(car, Sentence) = (0 + 0 + 1/3 + 2/4 + 0 + 0 + 3/7 + 0 + 4/9 + 0 + 0 + 0 + 0 + 5/14 + 0 + 0+ 0 + 0 + 0 + 0) / N = **0.137**

(car, Window)

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
rel	0	1	0	0	1	0	0	0	0	0	1	1	1	1	0	0	0	1	0	0

AP(car, Window) = (0 + 1/2 + 0 + 0 + 2/5 + 0 + 0 + 0 + 0 + 0+ 3/11 + 4/12 + 5/13 + 6/14 + 0 + 0 + 0 + 7/18 + 0 + 8/20) / N = **0.207**

(car, Dependency)

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
rel	1	1	0	1	0	0	1	1	1	0	0	1	0	1	0	0	1	1	1	0

AP(car, Dependency) = (1/1 + 2/2 + 0 + 3/4 + 0 + 0 + 4/7 + 5/8 + 6/9 + 0 + 0 + 7/12 + 0 + 8/14 + 0 + 0 + 9/17 + 10/18 + 11/19 + 0) / N = **0.495**

+-- piano --+

Sentence	top	sem	Window	top	sem	Dependency	top	sem
violin	+	+	violin	+	+	violin	+	+
flute	+	+	flute	+	+	viola	+	+
sonata	+	−	cello	+	+	guitar	+	+
cello	+	+	concerto	+	−	cello	+	+
concerto	+	−	sonata	+	−	bass	+	−
percussion	+	−	viola	+	+	flute	+	+
trumpet	+	+	op	+	−	keyboard	+	−
bass	+	−	string	+	−	percussion	+	−
saxophone	+	+	trumpet	+	+	drum	+	+
instrument	+	+	guitar	+	+	horn	+	+
viola	+	+	saxophone	+	+	saxophone	+	+
quartet	+	−	bass	+	−	trumpet	+	+
op	+	−	solo	+	−	instrument	+	+
composition	+	−	keyboard	+	−	vocal	+	−
tenor	+	−	instrument	+	+	orchestra	+	−
horn	+	+	quartet	+	−	organ	+	+
string	+	−	soloist	+	−	choir	+	−
trio	+	−	percussion	+	−	dance	+	−
orchestra	+	−	ensemble	+	−	music	+	−
pianist	+	−	acoustic	+	−	solo	+	−

Topically related

N = # unique_topical_identified = 20 + 6 + 6 = 32

(piano, Sentence)

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
rel	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

AP(piano, Sentence) = (1/1 + 2/2 + 3/3 + 4/4 + 5/5 + 6/6 + 7/7 + 8/8 + 9/9 + 10/10 + 11/11 + 12/12 + 13/13 + 14/14 + 15/15 + 16/16 + 17/17 + 18/18 + 19/19 + 20/20) / N = **0.625**

(piano, Window)

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
rel	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

AP(piano, Window) = (1/1 + 2/2 + 3/3 + 4/4 + 5/5 + 6/6 + 7/7 + 8/8 + 9/9 + 10/10 + 11/11 + 12/12 + 13/13 + 14/14 + 15/15 + 16/16 + 17/17 + 18/18 + 19/19 + 20/20) / N = **0.625**

(piano, Dependency)

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
rel	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

AP(car, Dependency) = (1/1 + 2/2 + 3/3 + 4/4 + 5/5 + 6/6 + 7/7 + 8/8 + 9/9 + 10/10 + 11/11 + 12/12 + 13/13 + 14/14 + 15/15 + 16/16 + 17/17 + 18/18 + 19/19 + 20/20) / N = **0.625**

Same semantic class

N = # unique_semantic_identified = 8 + 1 + 2 = 11

(piano, Sentence)

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
rel	1	1	0	1	0	0	1	0	1	1	1	0	0	0	0	1	0	0	0	0

AP(car, Sentence) = (1/1 + 2/2 + 0 + 3/4 + 0 + 0 + 4/7 + 0 + 5/9 + 6/10 + 7/11 + 0 + 0 + 0 + 0 + 8/16 + 0 + 0 + 0 + 0) / N = **0.51**

(piano, Window)

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
rel	1	1	1	0	0	1	0	0	1	1	1	0	0	0	1	0	0	0	0	0

AP(car, Window) = (1/1 + 2/2 + 3/3 + 0 + 0 + 4/6 + 0 + 0 + 5/9 + 6/10 + 7/11 + 0 + 0 + 0 + 8/15 + 0 + 0 + 0 + 0 + 0) / N = **0.544**

(piano, Dependency)

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
rel	1	1	1	1	0	1	0	0	1	1	1	1	1	0	0	1	0	0	0	0

AP(car, Dependency) = (1/1 + 2/2 + 3/3 + 4/4 + 0 + 5/6 + 0 + 0 + 6/9 + 7/10 + 8/11 + 9/12 + 10/13 + 0 + 0 + 11/16 + 0 + 0 + 0 + 0) / N = **0.83**

MAP - Topically related

MAP(Sentence) = average(AP(car, Sentence), AP(piano, Sentence)) = (0.777 + 0.625) / 2 = 0.701
MAP(Window) = average(AP(car, Window), AP(piano, Window)) = (0.57 + 0.625) / 2 = 0.597
MAP(Dependency) = average(AP(car, Dependency), AP(piano, Dependency)) = (0.3 + 0.625) / 2 = 0.462

MAP - Same semantic class

MAP(Sentence) = average(AP(car, Sentence), AP(piano, Sentence)) = (0.137 + 0.51) / 2 = 0.323
MAP(Window) = average(AP(car, Window), AP(piano, Window)) = (0.207 + 0.544) / 2 = 0.375
MAP(Dependency) = average(AP(car, Dependency), AP(piano, Dependency)) = (0.495 + 0.83) / 2 = 0.662

insights on these results

As you can see from the MAP results of the topical similarity – most of the topically-related words can be found in the first co-occurrence type's list (sentence-sized window), since it has the highest MAP score (0.701). This means that with a wide window (in this case a sentence-sized window) we are more likely to get topical similarity. In addition, we can see that the third co-occurence, which is the one based on dependencys, has the lowest MAP score (0.462) of all 3 co-occurences. So, we expect to find fewest topically-related words in the dependency's list. However, we can see that the MAP results of the semantic similarity indicate that most of the words that are in the same semantic class as the target word or semantically related to the target word, can be found in the list of the dependency co-occurrence type (MAP score of 0.662), which makes sense because dependencies features demonstrate syntactic relationships between the target word and the other words in the sentence.We can also notice that the MAP score of the first co-occurence, a sentence-sized window, is the lowest of all (0.323), so we expect to find fewest semantically-related words in its list.
Moreover, as expected, the second co-occurrence type (window of size k=2) has the middle MAP score in both similarity types, since this co-occurrence type obtains a narrow window approach that on one hands is more likely to capture semantic similarities because it catches the words surround the target word which are usually also the words that are syntactically directly related to it, but on the other hand it might also catch words that are topically-related to the target word.

5. Implementation details

a. PMI estimations

After applying all the required filters (in `filter_features` routine), I got a dictionary (`self.content_words_counts`) that its keys are only the most common words in the vocabulary, i.e. words that appeared at least 100 times in the corpus, and the value of each such key is a list with at most its 100 most common features, where next to each feature listed the number of times it has appeared as a context of this key.

Using this dictionary (`self.content_words_counts`), I created another dictionary (`self.context_counts`) in `get_context_counts` routine, where each key in it is an attribute (a feature) of at least one word from the list of common words and the value is a list of all the words in which this attribute was counted as their context.

With those dictionaries in my hand, I was able to create the columns matrix (`self.word_to_attributes_matrix`) for which I had to calculate the PMI score for each (common) word and attribute. The matrix creation performed in `get_word_to_attributes_matrix` routine, using the subroutine `compute_PMI` for computing the PMI score for each word and attribute.
* Before computing all the PMI scores, I first calculated the `total_num_pairs` variable which is the normalization factor, also marked as # (*,*), and known as twice the number of co-occurrences observed in the corpus (in our case, what's left of the corpus after applying all the filters).

<u>The computation of the PMI score in `compute_PMI`, is as follows:</u>

- `p_x = sum(self.content_words_counts[x].values()) / self.total_num_pairs`
  The probability that the word-attribute co-occurrence will have x as the word, i.e. the total number of co-occurrences of x with each of its features (`self.content_words_counts[x].values()`) normalized by `self.total_num_pairs`.

- `p_y = self.context_counts[y] / self.total_num_pairs`
  The probability that the word-attribute co-occurrence will have y as the attribute, i.e. the total amount of times y appeared as an attribute (`self.context_counts[y]`) normalized by `self.total_num_pairs`.

- `p_x_y = (self.content_words_counts[x][y]) / self.total_num_pairs`
  The probability that the word-attribute co-occurrence will have x as the word and y as the attribute, i.e. the total number of co-occurrences of x with the attribute y (`self.content_words_counts[x][y]`) normalized by `self.total_num_pairs`.

- `max(np.log(p_x_y / (p_x * p_y)), 0)`
  Finally, we calculate the PMI score of the current word-attribute pair, using the PMI formula and by applying the PPMI concept in which we turn negative PMI scores into 0.
  All 0 PMI scores are later ignored in `get_word_to_attributes_matrix`.

This way in `get_word_to_attributes_matrix`, we go through every word-attribute pair and calculate it's PMI score. We do it for each word with each one of its features and all these PMI scores of that specific word are saved in a dictionary where each key is the attribute with whom the PMI score was computed, and the value is the PMI score. This dictionary is basically the features vector of that word. All the features vectors together compose `self.word_to_attributes_matrix` .

b. The efficient algorithm for computing all similarities for a target word

After computing `self.word_to_attributes_matrix`, which is the columns matrix, as mentioned earlier, I use this matrix in `get_attribute_to_words_matrix` routine, in order to create the rows matrix – `self.attribute_to_words_matrix`.

This matrix will contain for each attribute used, a list of its PMI scores, where next to each PMI score is the identifier of the word with which this PMI score was computed.

```
self.attribute_to_words_matrix = defaultdict(list)
```

```
for i, attributes in enumerate(self.word_to_attributes_matrix):
    for att in attributes:
        self.attribute_to_words_matrix[att].append((i, attributes[att]))
```

Now, we have got both matrices – `self.word_to_attributes_matrix` and `self.attribute_to_words_matrix`, and we can compute the cosine similarity between a target word and all other words in the dictionary (the words in the columns – the keys of `self.word_to_attributes_matrix`). The computation is done in `cosine_similarity` routine.

<u>We use the efficient algorithm in order to compute the numerator of the cosine similarity formula, as follows:</u>

- `word_attributes = self.word_to_attributes_matrix[word_index]`

  First, we use the word index (the index of the target word for which we want to compute all similarities) the `cosine_similarity` routine received as input, in order to extract the target word's features vector.

- `similarity_results = [0] * len(self.common_lemmas)`

  Next, we initialize the similarity results vector.

- And then we use the efficient algorithm:

```
# For each attribute of the target word – att.
for att in word_attributes:

    # And for each PMI score computed for att with a certain word – v.
    for v in self.attribute_to_words_matrix[att]:

        # Compute the multiplication of the PMI score computed to the target-attribute
        pair (target-att pair) with the PMI score computed to the certain_word-
        attribute pair (v-att pair) and store the result in position - v id.
        similarity_results[v[0]] += word_attributes[att] *
                                    self.word_to_attributes_matrix[v[0]][att]
```

  After those loops are over we have the computed cosine-similarity numerator for each word in the `similarity_results` vector.

  * Side note: both matrices do not contain PMI values of 0, so there are no unnecessary multiplication operations during the computation in the efficient algorithm.

<u>The next step is calculating the denominator of the cosine for each of the words in the vocabulary:</u>

```
# Sum the PMI score squares of each attribute of the target word.
sum_u_att_squares = sum([word_attributes[att] * word_attributes[att] for att in
                         word_attributes])

for i in range(len(self.common_lemmas)):

    # Sum the PMI score squares of each attribute of the current word.
```

```
v_attributes = self.word_to_attributes_matrix[i]
sum_v_att_squares = sum([v_attributes[att] * v_attributes[att] for att in
                         v_attributes])
```

and then divide each numerator value (each cell) in the `similarity_results` vector with its denominator (The root of the multiplication result.):

```
similarity_results[i] /= np.sqrt(sum_u_att_squares * sum_v_att_squares)
```

to get the final similarity results.

6. Word2Vec Experiment

2) Word2Vec – 2nd order similarity:

20 most similar words, for each of the target, for each co-occurrence type , ordered by similarity in descending order is in **Appendix C**.

Conclusions from examining these lists

From examining these lists, we can see that with the vectors version of bow5 (bag of words - a window approach with k=5) we are more likely to get words that are topically-related to the target word. That is, the 'Bag of words' contexts induces topical similarities. Practically, most of the words in the 'Bag of Words' list are topically-related to the target word, where the rest of the words are sister-terms of the target word. For example, you can see that in the similarities table (with my manual similarity judgments) for the target word 'car' that's on page 13, the words that are marked with '+' in the 'top' column (as an abbreviation of topical-related), which is next to the 'Bag of words' column, are topically-related to the word 'car' and the words that are marked with '-' are sister-terms of 'car'. For example, the words 'motorbike', 'motorcycle', 'moped' and 'bike' are sister-terms of the word 'car'. You can also check the similarities table for the target word 'piano' (on page 15) in which all words in the 'top' column (next to the 'Bag of words' column) are marked with '+', i.e. all are topically-related to the word 'piano'.
Additionally, We can see that in both similarities tables (for 'car' and 'piano' target words) there are relatively many words that were marked with '+' in the 'sem' column (next to the 'Bag of words' and 'top' columns), most of these words are both - topically-related and semantically-related to the target word (marked with '+' in both columns – 'top' and 'sem'). The other words that appear as semantically-related to the target word but not as topically-related to it, are exactly the sister-terms I mentioned before, which are considered as semantic relations.
Moreover, words that appear in the 'Bag of Words' list and not in 'Dependency-Based' list are most certainly topically-related words. For example, for the target word 'car' the words driver, mid-engined, front-engined, mercedes-benz, rear-engined, etc. and for the target word 'piano' the words concerto, concertos, sonatas, etc.

However, with the version of dependency-based vectors we are more likely to get words that are semantically-related to the target word - in the same sematic class as the target word. That is, Dependency contexts induces functional similarities. For example , if you take a look in the similarities table (with my manual similarity judgements) for the target word ‘car’ on page 13 (or check the similarities table of the word ‘piano’ on page 15), you can see that the words which are marked with ‘+’ in the 'sem' column (as an abbreviation of semantically-related), which is next to the ‘Dependency-Based’ column, are in the same semantic class as the word ‘car’. We can also see that most of these words are also topically-related to the target word 'car' (are marked with ‘+’ in the ‘top’ column, next to the ‘Dependency-Based’ column). Additionally, Words that appear in the ‘Dependency-Based’ list and not in the ‘Bag of Words’ list are most certainly semantically-related to the target word, i.e. are in the same semantic class as the target word. For example, for the target word ‘car’ the words racecar, jeep, limo, taxicab, speedboat, wagon, etc.

3) Word2Vec – 1st order similarity:

20 top context attributes for each of the target words, for each of the 3 co-occurrence types, ordered by descending order of attributes with highest PMI values in the target word’s vector, in **Appendix D**.

Short qualitative comparison between the 2nd order lists and the 1st order lists

First-order context vectors record directly observable features of a context, whilst second-order context vectors aggregate vectors themselves associated to the directly observable features of the context. In other words, the main difference between the 2nd order lists and the 1st order lists is that the words that appear in the 1st order lists are words that have appeared multiple times with the target word in the same context. whereas the words in second-order lists are words that relate to other words in the dictionary in a manner similar to that of the target word. That is, these words and the target word have similar features.

The distinctions I have made earlier for each of the co-occurrences in the 2nd order similarity regarding the types of similarities that each co-occurrence is more likely to capture, are:

- 'Bag of Words' (BoW5) is more likely to capture topical similarity.
- 'Dependency-Based' is more likely to capture semantic-similarity.

In 1st order similarity:

- The 'Bag of Words' list contains words that appeared in the context in which the word 'car' appeared. The more a word have appeared next to the target word or in context surrounding the target word, it is more likely for it to appear in the top20 list and even rank quite high in it.
- The 'Dependency-Based' captures direct dependencies to the target word, such dependencies might not be captured with a window of size k=5, i.e. by using the ‘Bag of Words’ vectors version, since some words have no close connection of meaning with the

target word when they appear in the context of the target word, but these words can be captured by syntactic dependencies.

For example, I've marked in red words in the 1st order similarity tables of the target words 'car' and 'piano' (In **Appendix D** on pages 41 and 43, respectively) from which it can actually be seen that these words were drawn from the context of the sentence.

4) MAP:

+-- car --+

Bag of Words	top	sem	Dependency-Based	top	sem
cars	+	+	truck	+	+
truck	+	+	suv	+	+
automobile	+	+	vehicle	+	+
vehicle	+	+	minivan	+	+
motorbike	-	+	cars	+	+
motorcycle	-	+	speedboat	-	+
driver	+	-	racecar	+	+
minivan	+	+	automobile	+	+
suv	+	+	motorcar	+	+
lorry	+	+	jeep	+	+
motorcar	+	+	limousine	+	+
mid-engined	+	-	minibus	+	+
limousine	+	+	lorry	+	+
front-engined	+	-	limo	+	+
moped	-	+	motorcycle	-	+
motorhome	+	+	bike	-	+
mercedes-benz	+	-	motorhome	+	+
bike	-	+	taxicab	+	+
rear-engined	+	-	roadster	+	+
three-wheeled	+	-	wagon	-	+

<u>Topically related</u>

N = # unique_topical_identified = 16 + 6 = 22

(car, Bag of Words)

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
rel	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0	1	1	0	1	1

AP(car, Bag of Words) = (1/1 + 2/2 + 3/3 + 4/4 + 0 + 0 + 5/7 + 6/8 + 7/9 + 8/10 + 9/11 + 10/12 + 11/13 + 12/14 + 0 + 13/16 + 14/17 + 0 + 15/19 + 16/20) / N = **0.619**

(car, Dependency-Based)

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
rel	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	0	1	1	1	0

AP(car, Dependency-Based) = (1/1 + 2/2 + 3/3 + 4/4 + 5/5 + 0 + 6/7 + 7/8 + 8/9 + 9/10 + 10/11 + 11/12 + 12/13 + 13/14 + 0 + 0 + 14/17 + 15/18 + 16/19 + 0) / N = **0.668**

Same semantic class

N = # unique_semantic_identified = 14 + 8 = 22

(car, Bag of Words)

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
rel	1	1	1	1	1	1	0	1	1	1	1	0	1	0	1	1	0	1	0	0

AP(car, Bag of Words) = (1/1 + 2/2 + 3/3 + 4/4 + 5/5 + 6/6 + 0 + 7/8 + 8/9 + 9/10 + 10/11 + 0 + 11/13 + 0 + 12/15 + 13/16 + 0 + 14/18 + 0 + 0) / N = **0.582**

(car, Dependency-Based)

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
rel	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

AP(car, Dependency-Based) = (1/1 + 2/2 + 3/3 + 4/4 + 5/5 + 6/6 + 7/7 + 8/8 + 9/9 + 10/10 + 11/11 + 12/12 + 13/13 + 14/14 + 15/15 + 16/16 + 17/17 + 18/18 + 19/19 + 20/20) / N = **0.909**

+-- piano --+

Bag of Words	top	sem	Dependency-Based	top	sem
violin	+	+	violin	+	+
cello	+	+	cello	+	+
harpsichord	+	+	harpsichord	+	+
clarinet	+	+	saxophone	+	+
viola	+	+	clarinet	+	+
flute	+	+	guitar	+	+
bassoon	+	+	trombone	+	+
violoncello	+	+	mandolin	+	+
oboe	+	+	vibraphone	+	+
concerto	+	-	marimba	+	+
saxophone	+	+	accordion	+	+
accordion	+	+	pianoforte	+	+
harp	+	+	bassoon	+	+
trombone	+	+	fortepiano	+	+
sonatas	+	-	violoncello	+	+
trumpet	+	+	trumpet	+	+
mandolin	+	+	harmonica	+	+
pianoforte	+	+	clavinet	+	+
vibraphone	+	+	clavichord	+	+
concertos	+	-	euphonium	+	+

Topically related

N = # unique_topical_identified = 20 + 7 = 27

(piano, Bag of Words)

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
rel	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

AP(piano, Bag of Words) = (1/1 + 2/2 + 3/3 + 4/4 + 5/5 + 6/6 + 77/ + 8/8 + 9/9 + 10/10 + 11/11 + 12/12 + 13/13 + 14/14 + 15/15 + 16/16 + 17/17 + 18/18 + 19/19 + 20/20) / N = **0.74**

(piano, Dependency-Based)

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
rel	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

AP(piano, Dependency-Based) = (1/1 + 2/2 + 3/3 + 4/4 + 5/5 + 6/6 + 77/ + 8/8 + 9/9 + 10/10 + 11/11 + 12/12 + 13/13 + 14/14 + 15/15 + 16/16 + 17/17 + 18/18 + 19/19 + 20/20) / N = **0.74**

Same semantic class

N = # unique_semantic_identified = 17 + 7 = 24

(piano, Bag of Words)

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
rel	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0

AP(piano, Bag of Words) = (1/1 + 2/2 + 3/3 + 4/4 + 5/5 + 6/6 + 7/7 + 8/8 + 9/9 + 0 + 10/11 + 11/12 + 12/13 + 13/14 + 0 + 14/16 + 15/17 + 16/18 + 17/19 + 0) / N = **0.675**

(piano, Dependency-Based)

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
rel	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

AP(piano, Dependency-Based) = (1/1 + 2/2 + 3/3 + 4/4 + 5/5 + 6/6 + 77/ + 8/8 + 9/9 + 10/10 + 11/11 + 12/12 + 13/13 + 14/14 + 15/15 + 16/16 + 17/17 + 18/18 + 19/19 + 20/20) / N = **0.833**

MAP - Topically related

MAP(Bag of Words) = average(AP(car, Bag of Words), AP(piano, Bag of Words)) = (0.619 + 0.74) / 2 = 0.6795
MAP(Dependency-Based) = average(AP(car, Dependency-Based), AP(piano, Dependency-Based)) = (0.668 + 0.74) / 2 = 0.704

MAP - Same semantic class

MAP(Bag of Words) = average(AP(car, Bag of Words), AP(piano, Bag of Words)) = (0.582 + 0.675) / 2 = 0.628
MAP(Dependency-Based) = average(AP(car, Dependency-Based), AP(piano, Dependency-Based)) = (0.909 + 0.833) / 2 = 0.871

insights on these results

We can see that the MAP results for the semantic similarity indicate that most of the words that are in the same semantic class as the target word, can be found in the list of 'Dependency-Based' vectors version, since it has the highest MAP score (0.871). This result makes sense since the dependencies features demonstrate syntactic relationships between the target word and the other words in the sentence.

Moreover, The MAP results of the topical similarity show that most of the topically-related words can be found in the 'Dependency-Based' list, which is quite a surprise since the 'Bag of words' contexts are more identified with topical-similarity than 'Dependency-Based' contexts, since the 'Bag of words' contexts consist of words within a window of 5 words on each side of the target word, which are more likely to capture topically-related words (it's close to the sentence-sized window co-occurrence - it's basically 11-words sentences). Yet, let's notice that the difference between the 'Dependency-Based' MAP score (0.704) and the 'Bag of words' MAP score of (0.6795) is only 0.0245 which is negligible, meaning that both vectors versions are good at capturing topical similarity .

Anyway, from a second look at the MAP scores in both types of similarity (and on the similarity tables), we can see that all MAP scores are quite high, which may indicate that in the word2vec method we get a lot of words that are both topically and semantically related to the target word. This causes the MAP scores to be quite high and to reduce the gap between the 2 vectors versions when closely examining the MAP scores for each type of similarity. In other words, in the word2vec method, in both 'Bag of Words' and 'Dependency-Based' lists there are few words that are only topically-related to the target word or only semantically-related to the target word.

Comparison between the results of items 2-4 from the first part and the corresponding results for the word2vec experiment:

For Item 2

From the comparison of the types of similarity obtained for each co-occurrence type in item 2 in the first part and the same comparison made in item 2 in the word2vec experiment, It can be concluded that both 'Sentence Co-occurrence' (from the first part) and the 'Bag of Words' (BOW5) vectors version tend to capture topically-related words to the word target, whereas the 'Dependency Co-occurrence' (from the first part) and the 'Dependency-Based' vectors version tend to capture semantically-related words i.e. words that are in the same semantic class as the target word.
The 'Window Co-occurrence' from the first part can be associated with both capturing topically-related words to the target word and capturing semantically-related words to the target word, but in practice this type of co-occurrence is more likely to capture semantic similarity.

For Item 3

I must mention that in the word2vec experiment some of the words in both 1st order and 2nd order lists were quite strange and less typical ones compared to the words I got in the lists in the first part. I guess this is due to the use of a different corpus or the entire Wikipedia's corpus.
Anyway, regarding the comparisons I have made in the first part and in the word2vec experiment regarding the differences between the 1st order lists and the 2nd order lists, I don't think there is a difference between these comparisons, since in both the first part and word2vec experiment, the changes are due to the fact that first-order context vectors record directly observable features of a context, whilst second-order context vectors aggregate vectors themselves associated to the directly observable features of the context.

For Item 4

From the MAP results in the first part, I inferred that topical-similarity is more likely to appear on the 'Sentence Co-occurrence' list and semantic-similarity is more likely to appear on the 'Dependency Co-occurrence' list. However, In the word2vec experiment I got that semantic-similarity is more likely to appear on the 'Dependency-Based' list and for topical-similarity I got non-conclusive results that theoretically the 'Dependency-Based' is more likely to include topical-related words in its list than the 'Bag of Words' but the difference between the MAP scores was so small that I consider both vectors versions to be good at extracting topical-related words. But, as I have already mentioned in my answer of item 4 in the word2vec experiment, the MAP scores in both types of similarity (and on the similarity tables) were quite high, which may indicate that in the word2vec method we get a lot of words that are both topically and semantically related to the target word. This causes the MAP scores to be quite high and to reduce the gap between the 2 vectors versions when closely examining the MAP scores for each type of similarity. This is also the difference between the results in the first part to the results on the word2vec experiment.

In conclusion, the ‘Sentence Co-occurrence’ from the first part and the vectors version of ‘Bag of Words’ (k=5) should be better at capturing topical similarity, whereas the ‘Dependency Co-occurrence’ from the first part and the vectors version of ‘Dependency-Based’ should be better at capturing semantic similarity. However, with the use of the word2vec vectors we get that most of the words that appear in the lists are both topically and semantically related to the target word, making both vectors versions good for topical and semantic similarity.

As a side note, of course that for a system that should be good at capturing semantic similarity I would recommend using the ‘Dependency-Based’ vectors and for a system that should be good at capturing topical similarity I would recommend using the ‘Bag of Words’ vectors.

Appendix A: 2nd order similarity

+-- car --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
drive	driver	vehicle
driver	truck	truck
truck	motor	driver
vehicle	drive	motorcycle
motor	vehicle	racing
ford	racing	station
automobile	ford	locomotive
race	formula	automobile
auto	race	horse
formula	lap	motor
racing	motorcycle	traffic
toyota	automobile	aircraft
engine	bus	stock
motorcycle	bicycle	auto
wheel	stock	item
chassis	nascar	cyclist
bmw	traffic	plane
nascar	carriage	yacht
gt	toyota	van
crash	trailer	model

+-- bus --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
rail	rail	train
commuter	commuter	rail
transit	passenger	tram
transportation	transit	taxi
transport	tram	ferry
station	train	transit
passenger	freight	road
line	metro	railway

freight	terminal	vehicle
operate	route	traffic
train	line	route
tram	station	passenger
route	transport	boat
depot	ferry	subway
metro	railway	freight
terminal	operate	transport
hub	taxi	truck
traffic	junction	cable
connect	stop	automobile
ferry	airport	commuter

+-- hospital --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
medical	clinic	clinic
clinic	medical	school
care	patient	college
health	nursing	library
surgeon	psychiatric	museum
medicine	health	airport
surgery	care	station
doctor	library	prison
physician	centre	hall
treatment	facility	center
facility	center	park
rehabilitation	surgery	hotel
patient	physician	campus
center	sick	office
psychiatric	surgeon	headquarters
dr	rehabilitation	jail
nh	nurse	store
emergency	school	theatre
establishment	office	institute
surgical	dental	town

+-- hotel --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
restaurant	resort	restaurant
shop	restaurant	resort
inn	casino	store
pub	shop	palace
owner	apartment	shop
resort	store	casino
apartment	owner	theater
store	pub	apartment
dining	palace	estate
retail	luxury	farm
bar	mall	inn
luxury	residence	house
tourist	cafe	building
chain	retail	station
plaza	plaza	mill
supermarket	cottage	complex
purchase	inn	castle
cafe	lobby	factory
house	nightclub	hospital
downtown	nearby	supermarket

+-- gun --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
cannon	fire	cannon
rifle	cannon	weapon
artillery	rifle	mortar
fire	artillery	pistol
mortar	arm	artillery
assault	battery	rifle
weapon	bullet	engine
ammunition	mortar	battery
tank	weapon	sword
trench	machine	rocket

sniper	enemy	camera
bullet	turret	ammunition
battery	rocket	tube
enemy	tank	missile
pistol	assault	machine
armament	pistol	firing
firing	ammunition	tank
grenade	kill	knife
battalion	mm	aircraft
soldier	load	blade

+-- bomb --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
bombing	bomber	torpedo
raid	attack	bombing
luftwaffe	bombing	missile
bomber	plane	explosion
injure	torpedo	earthquake
explode	raid	shell
explosion	explosive	bomber
enemy	fighter	rocket
explosive	aircraft	weapon
airfield	luftwaffe	grenade
attack	destroy	ball
blast	enemy	fire
aircraft	pilot	bullet
destroy	explosion	air
target	crash	chemical
terrorist	airfield	destroy
raf	terrorist	raid
fly	allied	charge
torpedo	ship	flood
weapon	kill	strike

+-- horse --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
rid	bike	dog
jockey	ride	car
racing	breed	motorcycle
thoroughbred	thoroughbred	animal
rider	pig	thoroughbred
ride	carriage	bike
5th	sheep	bicycle
handicap	dog	volunteer
regiment	guard	bull
cyclist	regiment	motor
race	rid	cavalry
bike	deer	auto
artillery	riding	cat
guard	cattle	vehicle
pig	bull	man
breeder	motorcycle	player
hunt	rider	person
breed	bicycle	goat
cavalry	cart	truck
detachment	goat	infantry

+-- fox --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
cbs	cbs	cbs
nbc	nbc	nbc
affiliate	abc	abc
abc	network	cbc
affiliation	broadcast	paramount
network	cnn	bbc
programming	news	cable
broadcast	programming	radio
news	anchor	smith
broadcasting	television	television

channel	switch	entertainment
show	pb	tv
espn	tv	espn
anchor	radio	pb
newscast	broadcasting	itv
cnn	bbc	sport
television	channel	hudson
switch	espn	anderson
televise	show	shaw
kid	cbc	network

+-- table --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
row	bottom	list
column	list	category
list	row	watchlist
contain	column	page
bottom	top	system
heading	contain	infobox
key	reference	content
element	header	scene
following	entry	picture
content	box	history
header	content	column
example	heading	diagram
place	database	section
bit	article	ranking
text	pool	map
finish	second	image
point	footnote	top
comparison	following	season
ranking	section	tower
html	template	hole

+-- bowl --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
bowler	wicket	cup
batsman	inning	league
pace	bowler	super
consecutive	batsman	wrestling
wicket	pro	playoff
inning	bat	baseball
super	super	all-ireland
victory	league	soccer
steelers	season	tennis
score	consecutive	nfl
ncaa	match	ncaa
tournament	win	junior
colt	final	all-star
patriot	ncaa	football
cowboy	occasional	olympic
bat	victory	eurovision
nfl	overall	championship
cup	competition	cricket
season	all-star	golf
game	score	rugby

+-- guitar --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
bass	bass	drum
drum	drum	bass
keyboard	instrument	keyboard
vocal	keyboard	piano
acoustic	vocal	vocal
instrument	acoustic	flute
percussion	piano	cello
solo	string	saxophone
flute	flute	trumpet
rhythm	solo	violin

band	vocalist	instrument
guitarist	percussion	percussion
saxophone	violin	tenor
backing	saxophone	viola
piano	rhythm	string
drummer	melody	organ
vocalist	ensemble	guitarist
trumpet	trumpet	horn
string	cello	music
lineup	tune	intro

+-- piano --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
violin	violin	violin
flute	flute	viola
sonata	cello	guitar
cello	concerto	cello
concerto	sonata	bass
percussion	viola	flute
trumpet	op	keyboard
bass	string	percussion
saxophone	trumpet	drum
instrument	guitar	horn
viola	saxophone	saxophone
quartet	bass	trumpet
op	solo	instrument
composition	keyboard	vocal
tenor	instrument	orchestra
horn	quartet	organ
string	soloist	choir
trio	percussion	dance
orchestra	ensemble	music
pianist	acoustic	solo

Appendix B: 1st order similarity

+-- car --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
driver	touring	('parking', 'compmod', '↑')
truck	accident	('wash', 'compmod', '↑')
accident	parking	('car', 'conj', '↓')
racing	truck	('truck', 'conj', '↓')
car	driver	('armoured', 'amod', '↓')
ford	crash	('race', 'dobj', '↑')
crash	racing	('drive', 'partmod', '↓')
formula	formula	('concept', 'compmod', '↓')
motor	steal	('race', 'amod', '↓')
drive	motor	('accident', 'compmod', '↑')
wheel	ford	('f1', 'compmod', '↓')
race	stock	('race', 'partmod', '↓')
vehicle	cable	('car', 'conj', '↑')
passenger	drive	('armored', 'amod', '↓')
speed	bomb	('bomb', 'compmod', '↑')
engine	buy	('ferry', 'compmod', '↑')
front	concept	('crash', 'compmod', '↑')
sport	race	('drive', 'dobj', '↑')
sell	passenger	('fit', 'nsubjpass', '↑')
train	fast	('hit', 'adpmod', '↑', 'by')

+-- bus --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
depot	taxi	('tram', 'conj', '↑')
bus	depot	('subway', 'conj', '↑')
transit	terminal	('route', 'nsubj', '↑')
terminal	subway	('rout', 'nsubj', '↑')
operator	tram	('stop', 'compmod', '↑')
taxi	interchange	('terminus', 'compmod', '↑')
interchange	commuter	('rail', 'conj', '↑')
stop	transit	('bus', 'conj', '↑')

transportation	operator	('paint', 'nsubjpass', '↑')
transport	shelter	('interchange', 'compmod', '↑')
truck	shuttle	('accessible', 'adpmod', '↑', 'by')
metro	truck	('serial', 'compmod', '↓')
terminus	frequent	('rail', 'conj', '↓')
express	stop	('stand', 'compmod', '↑')
regular	parking	('passenger', 'compmod', '↑')
connect	route	('bus', 'conj', '↓')
passenger	rapid	('shelter', 'compmod', '↑')
travel	lane	('hybrid', 'amod', '↓')
route	terminus	('terminal', 'compmod', '↑')
transfer	connect	('travel', 'partmod', '↓')

+-- hospital --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
psychiatric	psychiatric	('doctor', 'adpmod', '↑', 'at')
bed	bed	('teaching', 'amod', '↓')
clinic	clinic	('rush', 'adpmod', '↑', 'to')
hospital	teaching	('hospital', 'appos', '↓')
dr	dr	('mercy', 'compmod', '↓')
teaching	rush	('hospital', 'conj', '↑')
surgery	recover	('stay', 'compmod', '↑')
trust	hospital	('hospital', 'appos', '↑')
patient	admit	('clinic', 'conj', '↓')
doctor	patient	('hospital', 'conj', '↓')
care	treat	('psychiatric', 'amod', '↓')
medical	memorial	('transport', 'adpmod', '↑', 'to')
memorial	mental	('bed', 'compmod', '↑')
emergency	doctor	('teaching', 'compmod', '↓')
cancer	emergency	('child', 'adpmod', '↓', 'for')
heart	stay	('hopkins', 'compmod', '↓')
treatment	medical	('take', 'adpmod', '↑', 'to')
medicine	cancer	('eye', 'compmod', '↓')
facility	care	('visit', 'adpmod', '↑', 'in')
private	st	('mary', 'poss', '↓')

+-- hotel --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
hotel	hilton	('hilton', 'compmod', '↓')
luxury	casino	('resort', 'conj', '↓')
inn	lobby	('hotel', 'appos', '↑')
resort	luxury	('lobby', 'compmod', '↑')
plaza	plaza	('neutral', 'compmod', '↓')
restaurant	resort	('restaurant', 'adpmod', '↑', 'at')
stay	hotel	('hotel', 'appos', '↓')
chain	restaurant	('casino', 'conj', '↓')
palace	lodge	('hotel', 'conj', '↓')
guest	tourism	('plaza', 'compmod', '↓')
owner	chain	('resort', 'compmod', '↓')
room	stay	('vega', 'adpmod', '↓', 'in')
purchase	room	('savoy', 'compmod', '↓')
shop	convert	('luxury', 'compmod', '↓')
bar	palace	('hotel', 'conj', '↑')
tower	tourist	('stay', 'adpmod', '↑', 'at')
spring	nearby	('restaurant', 'conj', '↓')
grand	purchase	('operate', 'adpmod', '↑', 'a')
store	owner	('palace', 'compmod', '↓')
operate	check	('convert', 'adpmod', '↑', 'into')

+-- gun --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
anti-aircraft	anti-aircraft	('anti-aircraft', 'compmod', '↓')
turret	turret	('rifle', 'amod', '↓')
machine	machine	('deck', 'compmod', '↓')
ammunition	barrel	('aim', 'dobj', '↑')
barrel	ammunition	('barrel', 'compmod', '↑')
gun	mortar	('mounted', 'amod', '↓')
rose	mm	('machine', 'compmod', '↓')
battery	battery	('man', 'dobj', '↑')
rifle	rose	('fire', 'rcmod', '↓')
shot	deck	('battery', 'adpmod', '↑', 'of')

deck	n	('armstrong', 'compmod', '↓')
fit	rifle	('mm', 'compmod', '↓')
assault	gun	('gun', 'conj', '↓')
artillery	shield	('arm', 'adpmod', '↑', 'with')
weapon	shot	('ammunition', 'conj', '↓')
n	instal	('shoot', 'adpmod', '↑', 'with')
heavy	mount	('mount', 'nsubjpass', '↑')
shoot	artillery	('jump', 'dobj', '↑')
mount	lewis	('mount', 'nsubj', '↑')
arm	jump	('rifle', 'conj', '↑')

+-- bomb --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
explode	explode	('luftwaffe', 'nsubj', '↓')
luftwaffe	luftwaffe	('pipe', 'amod', '↓')
ira	petrol	('ally', 'adpmod', '↓', 'by')
explosion	disposal	('explode', 'nsubj', '↑')
bombing	atomic	('plant', 'partmod', '↓')
atomic	blast	('explode', 'compmod', '↑')
explosive	ira	('aim', 'nsubjpass', '↑')
raid	explosive	('parcel', 'compmod', '↓')
bomb	conventional	('wave', 'adpmod', '↓', 'in')
drop	raid	('cluster', 'compmod', '↓')
injure	explosion	('raid', 'dobj', '↓')
bomber	hydrogen	('blast', 'compmod', '↑')
target	pipe	('explosion', 'compmod', '↑')
plane	drop	('atomic', 'amod', '↓')
nuclear	raf	('aircraft', 'adpmod', '↓', 'by')
sink	terrorist	('ira', 'compmod', '↓')
destroy	suicide	('drop', 'dobj', '↑')
least	squad	('disposal', 'compmod', '↑')
damage	target	('hydrogen', 'compmod', '↓')
weapon	allied	('alert', 'compmod', '↑')

+-- horse --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
rid	harness	('harness', 'compmod', '↓')
thoroughbred	crazy	('pony', 'conj', '↓')
jockey	trojan	('trojan', 'compmod', '↓')
horse	thoroughbred	('thoroughbred', 'amod', '↓')
rider	mule	('mule', 'conj', '↓')
trailer	pony	('bull', 'conj', '↑')
ride	carriage	('crazy', 'compmod', '↓')
cattle	rid	('race', 'adpmod', '↑', 'for')
breed	jockey	('rid', 'partmod', '↓')
stake	stable	('ride', 'dobj', '↑')
racing	riding	('wagon', 'conj', '↓')
wild	rider	('man', 'adpmod', '↑', 'on')
cavalry	racing	('rider', 'conj', '↓')
artillery	ride	('troop', 'adpmod', '↑', 'of')
farm	cattle	('rid', 'dobj', '↑')
brigade	sheep	('ride', 'conj', '↓')
trail	breed	('breeding', 'compmod', '↑')
regiment	wild	('race', 'partmod', '↓')
race	steal	('fall', 'adpmod', '↑', 'from')
animal	artillery	('tram', 'compmod', '↑')

+-- fox --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
hound	hound	('hound', 'conj', '↓')
fox	sac	('morning', 'adpmod', '↑', 'on')
affiliation	coyote	('beaver', 'conj', '↑')
kid	affiliation	('televise', 'adpmod', '↑', 'on')
affiliate	affiliate	('hare', 'conj', '↑')
nbc	cnn	('sac', 'conj', '↑')
abc	hunting	('abc', 'conj', '↑')
switch	kid	('ohio', 'compmod', '↑')
cbs	net	('hunting', 'compmod', '↑')
programming	fox	('subspecies', 'adpmod', '↑', 'of')

20th	matthew	('cat', 'conj', '↓')
entertainment	programming	('kit', 'compmod', '↓')
news	nbc	('net', 'compmod', '↑')
sport	news	('affiliation', 'compmod', '↑')
channel	switch	('twentieth', 'compmod', '↓')
morning	abc	('gray', 'amod', '↓')
picture	twentieth	('kid', 'compmod', '↑')
originally	20th	('squirrel', 'compmod', '↑')
network	distribute	('mask', 'compmod', '↑')
soccer	hunt	('sport', 'compmod', '↑')

+-- table --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
table	periodic	('following', 'compmod', '↓')
row	picnic	('periodic', 'compmod', '↓')
gospel	tennis	('list', 'adpmod', '↓', 'below')
column	table	('picnic', 'amod', '↓')
tennis	column	('tennis', 'compmod', '↑')
following	row	('truth', 'compmod', '↓')
key	coffee	('table', 'conj', '↑')
element	database	('column', 'adpmod', '↑', 'of')
format	bottom	('sit', 'adpmod', '↑', 'at')
content	following	('picnic', 'compmod', '↓')
text	chair	('content', 'adpmod', '↓', 'of')
contain	knight	('medal', 'compmod', '↓')
top	content	('table', 'conj', '↓')
finish	salt	('salt', 'compmod', '↑')
round	indicate	('periodic', 'amod', '↓')
section	entry	('rotary', 'amod', '↓')
medal	round	('chair', 'conj', '↑')
data	half	('update', 'dobj', '↑')
list	upper	('compare', 'nsubj', '↑')
turn	top	('row', 'adpmod', '↑', 'of')

+-- bowl --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
quiz	quiz	('batsman', 'rcmod', '↑')
rose	batsman	('over', 'dobj', '↓')
pace	super	('bowl', 'conj', '↑')
bowl	right-handed	('select', 'adpmod', '↑', 'to')
steelers	subdivision	('quiz', 'compmod', '↓')
right-handed	rose	('bowl', 'conj', '↓')
super	bermuda	('air', 'adpmod', '↑', 'during')
batsman	alley	('alley', 'amod', '↑')
pro	pro	('invitation', 'adpmod', '↑', 'to')
bowler	lawn	('green', 'dobj', '↓')
sugar	bowler	('rose', 'compmod', '↓')
consecutive	hawaii	('subdivision', 'compmod', '↑')
orange	cotton	('ball', 'rcmod', '↑')
wicket	sugar	('hand', 'nsubjpass', '↑')
cotton	dust	('compass', 'compmod', '↓')
formerly	liberty	('pace', 'dobj', '↓')
inning	bowl	('bermuda', 'compmod', '↓')
nfl	orange	('game', 'appos', '↓')
rename	wicket	('pro', 'compmod', '↓')
victory	selection	('barrow', 'compmod', '↑')

+-- guitar --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
acoustic	acoustic	('vocal', 'appos', '↑')
gibson	amplifier	('vocal', 'appos', '↓')
line-up	vintage	('amplifier', 'conj', '↓')
amplifier	gibson	('flourish', 'compmod', '↑')
playing	rhythm	('sang', 'conj', '↓')
backing	bass	('intro', 'compmod', '↑')
rhythm	keyboard	('tune', 'dobj', '↑')
keyboard	synthesizer	('gibson', 'compmod', '↓')
percussion	percussion	('bass', 'appos', '↓')
signature	vocal	('amplifier', 'compmod', '↑')

bass	backing	('vocal', 'conj', '↓')
drum	tenor	('keyboard', 'conj', '↓')
saxophone	hero	('rhythm', 'compmod', '↓')
hero	chord	('vocal', 'conj', '↑')
guitar	electric	('acoustic', 'amod', '↓')
vocal	harmony	('solo', 'conj', '↓')
electric	playing	('bass', 'compmod', '↓')
string	tune	('bass', 'conj', '↓')
piano	drum	('consist', 'adpmod', '↑', 'on')
violin	signature	('synthesizer', 'conj', '↓')

+-- piano --+

Sentence Co-occurrence	Window Co-occurrence	Dependency Co-occurrence
opus	opus	('harp', 'conj', '↑')
sonata	sonata	('saxophone', 'conj', '↑')
op	concerto	('cum', 'compmod', '↑')
cello	op	('aid', 'amod', '↓')
viola	recital	('percussion', 'conj', '↑')
trio	viola	('viola', 'conj', '↑')
lesson	trio	('cello', 'conj', '↑')
concerto	cello	('op', 'conj', '↓')
violin	lesson	('compose', 'adpmod', '↑', 'for')
clarinet	synthesizer	('recital', 'compmod', '↑')
arrangement	violin	('train', 'adpmod', '↑', 'on')
pianist	clarinet	('trio', 'adpmod', '↑', 'for')
piano	harp	('ph', 'compmod', '↓')
trumpet	trumpet	('flute', 'conj', '↓')
saxophone	saxophone	('roll', 'dobj', '↓')
flute	suite	('voice', 'conj', '↑')
organ	flute	('organ', 'conj', '↑')
composition	arrangement	('violin', 'conj', '↑')
horn	organ	('violin', 'conj', '↓')
keyboard	soprano	('lesson', 'compmod', '↑')

Appendix C: Word2Vec – 2nd order similarity

car

Bag of Words	Dependency-Based
cars	truck
truck	suv
automobile	vehicle
vehicle	minivan
motorbike	cars
motorcycle	speedboat
driver	racecar
minivan	automobile
suv	motorcar
lorry	jeep
motorcar	limousine
mid-engined	minibus
limousine	lorry
front-engined	limo
moped	motorcycle
motorhome	bike
mercedes-benz	motorhome
bike	taxicab
rear-engined	roadster
three-wheeled	wagon

bus

Bag of Words	Dependency-Based
buses	minibus
tram	tram
metrobus	buses
intercity	jeepney
busses	taxicab
fixed-route	motorcoach
minibus	taxi
inter-city	trolleybus
ksrtc	lorry
commuter	truck
apsrtc	metrobus
msrtc	streetcar
inter-urban	busses
dial-a-ride	ferryboat
mini-bus	trolley
light-rail	tramcar
rail	railcar
transit	railmotor
trolleybus	intercityexpress
limited-stop	train

+-- hospital --+

Bag of Words	Dependency-Based
clinic	sanatorium
hospitals	hospice
infirmary	sanitorium
hospice	hospitals
lying-in	sanitarium
dispensary	clinic
polyclinic	infirmary
sanatorium	polyclinic
convalescent	dispensary
mulago	orphanage
addenbrooke	poorhouse
bethlem	almshouse
psychiatric	workhouse
maudsley	institutet
siriraj	leprosarium
sanitarium	rikshospitalet
in-patient	heliport
incurables	gaol
orthopaedic	guesthouse
westmead	motherhouse

+-- hotel --+

Bag of Words	Dependency-Based
motel	motel
restaurant	hotels
doubletree	casino
sheraton	restaurant
hotels	inn
ritz-carlton	guesthouse
sofitel	tavern
westin	cafe
ramada	ritz-carlton
casino	nightclub
kempinski	travelodge
mansion	pizzeria
inn	roadhouse
cafe	boardinghouse
tavern	café
apartments	condo
boutique	brewpub
nightclub	sheraton
marriott	steakhouse
travelodge	brasserie

+-- gun --+

Bag of Words	Dependency-Based
guns	guns
cannon	handgun
howitzer	machinegun
sub-machine	howitzer
flamethrower	pistol
belt-fed	rifle
37mm	shotgun
smoothbore	firearm
pistol	cannon
shkas	musket
105mm	crossbow
40mm	autocannon
gatling	phaser
recoilless	flamethrower
76mm	revolver
3-inch	carbine
rifle	machine-gun
88mm	weapon
large-caliber	carronade
autocannons	pounder

+-- bomb --+

Bag of Words	Dependency-Based
bombs	bombs
detonated	firebomb
detonates	landmine
detonate	car-bomb
booby-trap	grenade
detonating	torpedo
firebomb	ied
car-bomb	warhead
exploded	bomber
detonation	bomblets
warhead	missile
500-pound	nuke
b61	detonator
laser-guided	booby-trap
blast	kamikaze
explosives	munition
detonations	explosives
landmine	machinegun
tallboy	a-bomb
thermonuclear	firebombs

+-- horse --+

Bag of Words	Dependency-Based
horses	horses
standardbred	goat
saddlebred	dog
gelding	stallion
thoroughbred	mule
stallion	bronc
racehorses	cow
dog	unicycle
riderless	greyhound
gaited	bareback
bronc	camel
percheron	appaloosa
pony	saddlebred
trotting	colt
harness	zebu
chariot	donkey
appaloosa	sidesaddle
sulky	racehorse
racehorse	elephant
greyhound	pony

+-- fox --+

Bag of Words	Dependency-Based
abc	daystar
cbs	nbc
nbc	byutv
wxyz-tv	kron
msnbc	wolf
ctv	cbs-tv
wsvn	familynet
familynet	abc
wttg	wccb
wjbk	oln
wfxt	wjar
espn	hdnet
wofl	telefutura
cnn	woodchuck
oln	nbc-tv
blitzer	soapnet
nesn	cinemax
wesh	wdiv
wb	mundofox
wgn-tv	coyote

+-- table --+

Bag of Words	Dependency-Based
tables	tables
sortable	leaderboard
wikitable	sideboard
look-up	chessboard
foosball	textbox
toc	taskbar
bulleted	gameboard
ping-pong	worksheet
billiard	tray
table-tennis	viewport
textbox	dais
tray	flowchart
lookup	playfield
wikitables	mantelpiece
brackets	stepladder
header	cladogram
footer	letterbox
tabular	windowsill
menu	bookcase
carom	wikitable

+-- bowl --+

Bag of Words	Dependency-Based
xliii	bowls
xlii	superbowl
xliv	arenabowl
bowls	wcws
xlvi	wnit
tostitos	nlcs
xli	arenacup
xxxviii	postseason
xlv	nit
xxxv	xliii
xxxix	llws
xxxvii	beanpot
xlvii	xlv
xxxvi	triplemanía
xxxiv	alcs
bluebonnet	nlds
gator	cup
xxviii	kvalserien
xxxii	tourney
xxxi	cws

+-- guitar --+

Bag of Words	Dependency-Based
harmonica	saxophone
mandolin	bass
bass	mandolin
drums	harmonica
guitars	accordion
keyboards	trombone
accordion	violin
banjo	banjo
saxophone	guitars
12-string	cello
ukulele	piano
trombone	vibraphone
fiddle	sax
autoharp	trumpet
melodica	autoharp
percussion	clarinet
vibraphone	sitar
tambourine	fiddle
vocals	drums
fretless	marimba

+-- piano --+

Bag of Words	Dependency-Based
violin	violin
cello	cello
harpsichord	harpsichord
clarinet	saxophone
viola	clarinet
flute	guitar
bassoon	trombone
violoncello	mandolin
oboe	vibraphone
concerto	marimba
saxophone	accordion
accordion	pianoforte
harp	bassoon
trombone	fortepiano
sonatas	violoncello
trumpet	trumpet
mandolin	harmonica
pianoforte	clavinet
vibraphone	clavichord
concertos	euphonium

Appendix D: Word2Vec – 1st order similarity

car

Bag of Words	Dependency-Based
car	adpmod:byI_commute
racing	amod_street-legal
mygale	conj_hovercraft
bmw	amod_newly-designed
driver	amod_liter
motor	amod_late-model
cars	poss_brink
dealership	compmod_m1918
parked	adpmod:fromI_tossed
rear-drive	compmod_high-wing

hospital

Bag of Words	Dependency-Based
hospital	compmod_siriraj
bethlem	compmod_safdarjung
moorfields	adpmod:of_nuova
psychiatric	compmod_strangeways
hospitals	amod_maximum-security
infirmary	compmod_armley
foundling	compmod_combermere
siriraj	compmod_fresnes
maudsley	compmod_eastview
westmead	compmod_desloge

bus

Bag of Words	Dependency-Based
bus	adpmod:byI_commute
buses	amod_east-bound
ksrtc	conj_hovercraft
samtrans	adpmod:onI_ridership
seabus	compmod_airlink
smrt	compmod_operates
connexxion	conj_busses
inter-city	compmod_xpt
intercity	compmod_yrt
msrtc	dobjI_onboard

hotel

Bag of Words	Dependency-Based
hotel	compmod_nymphenburg
hotels	compmod_dolmabahçe
westin	compmod_whitwell
radisson	conj_guesthouses
kempinski	compmod_ravenscourt
sofitel	compmod_hanworth
ramada	compmod_siriraj
biltmore	compmod_strangeways
ritz-carlton	compmod_safdarjung
sheraton	compmod_armley

+-- gun --+

Bag of Words	Dependency-Based
gun	num_88mm
guns	adpmod:withI_rearmed
submachine	compmod_m1918
machine	compmodI_fellatio
gatling	num_60mm
sub-machine	adpmod:ofI_hilt
howitzer	compmod_karabiner
rifle	compmod_t-55
40mm	adpmod:forI_cartridges
11-inch	compmodI_cwt

+-- horse --+

Bag of Words	Dependency-Based
horse	amod_gaited
horses	nsubjpassI_spooked
standardbred	compmodI_drovers
thoroughbred	dobjI_shoe
trotting	adpmod:byI_commute
riding	rcmod_roams
racing	poss_quixote
ridden	compmod_ch-53e
galloping	appos_pony
bred	compmod_poitevin

+-- bomb --+

Bag of Words	Dependency-Based
bomb	num_88mm
bombs	adpmod:on_pentagon
detonated	compmodI_splashes
detonates	adpmod:byI_unharmed
detonate	adpmod:withI_rearmed
atomic	compmod_single-car
exploded	compmod_m1918
detonating	compmod_phishing
bomber	compmodI_fellatio
bombing	rcmod_bursts

+-- fox --+

Bag of Words	Dependency-Based
fox	amod_crab-eating
vulpes	appos_canadensis
news	compmodI_subchannels
movietone	conj_mattel
owned-and-operated	compmod_jacki
nbc	conjI_raimi
cbs	conjI_corgan
terrier	conjI_leda
affiliate	conj_bluebird
cnn	conj_nolte

+-- table --+

Bag of Words	Dependency-Based
table	adpmod:inI_rows
sortable	adpmod:offI_knocks
tables	adpmod:onI_seventh
tennis	amod_ten-foot
lookup	compmod_abydos
billiards	amod_25-metre
foosball	adpmod:intoI_face
periodic	adpmod:belowI_added
billiard	adpmod:throughI_throw
toc	compmod_three-judge

+-- guitar --+

Bag of Words	Dependency-Based
guitar	conjI_bongos
bass	amod_end-blown
guitars	conj_back-up
drums	adpmod:onI_harris
keyboards	adpmod:forI_adagio
nyckelharpa	adpmod:onI_thompson
glockenspiel	appos_mandolin
6-string	dep_keyboard
tambourine	adpmod:ofI_virtuoso
banjo	adpmod:onI_foster

+-- bowl --+

Bag of Words	Dependency-Based
bowl	adpmod:against_auburn
xli	conjI_preseason
bowls	adpmod:against_oklahoma
xliv	adpmod:inI_2-1
xxxviii	conj_play-offs
xliii	compmod_pan-pacific
xlii	adpmod:forI_stadiums
xlvi	adpmod:atI_clinched
xlv	adpmod:intoI_win
super	amod_25-metre

+-- piano --+

Bag of Words	Dependency-Based
piano	adpmod:forI_adagio
violin	amod_end-blown
sonata	conjI_bongos
cello	adpmod:onI_harris
concerto	adpmod:ofI_virtuoso
op	adpmod:onI_thompson
harpsichord	conj_back-up
concertos	compmod_kreutzer
viola	compmod_vomeronasal
violoncello	adpmod:onI_supplemented