PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 114 APRIL 2020 35–57

Using Word Embeddings and Collocations for Modelling Word Associations

Micha de Rijk, David Mareček

Institute of Formal and Applied Linguistics, Faculty of Mathematics and Physics, Charles University

Abstract

Word association is an important part of human language. Many techniques for capturing semantic relations between words exist, but their ability to model word associations is rarely tested in a real application. In this paper, we evaluate three models aimed at different types of word associations: a word-embedding model for synonymy, a point-wise mutual information model for word collocations, and a dependency model for common properties of words. The quality of the proposed models is tested on English and Czech by humans in an online version of the word-association game "Codenames".

1. Introduction

Association is one of the basic mechanisms of human memory. It is based on the past experience of a man and existing relationship between the phenomena of the real world (Morkovkin, 1970). A well-known word game called "Word associations" involves an exchange of words that are associated together. Its idea is based on the connection and production of other words in spontaneous response to a given word. In another version of the game, the associations between words must be strictly obvious, rather than the usual "first word that comes to mind", which can often require explaining how it is connected with the previous word.

Word associations can also be used in psychology or psychiatric evaluations. Jung (1910) theorized that people connect ideas, feelings, experiences and information by way of associations. Gough (1976) believes that word association can reveal something of a person's subconscious mind as it shows what things they associate together.

 Corresponding author: marecek@ufal.mff.cuni.cz
Cite as: Micha de Rijk, David Mareček. Using Word Embeddings and Collocations for Modelling Word Associations. The Prague Bulletin of Mathematical Linguistics No. 114, 2020, pp. 35–57. doi: 10.14712/00326585.002.

Words can be associated in many different ways, some of them are listed in the following overview:

- **Synonyms**: A synonym is a word that expresses the same concept as another word. For example, *drink* is a synonym of*beverage*.
- **Hypernyms**: We say that A is a hypernym of B if A describes a set of concepts that B belongs to. For example, *fruit* is a hypernym of *apple* because *apple* belongs to the set of objects described by the word *fruit*.
- **Hyponyms**: The opposite of a hypernym. The word *apple* is a hyponym of *fruit* because *apple* is a type of *fruit*.
- **Co-hyponyms**: The co-hyponym relation refers to words that have a hypernym in common such as *knife* and *fork* which are both hyponyms of the word *utensil*.
- **Meronyms**: A meronym is a word that is a part or member of another. For example, *sentence* is a meronym of *text* because a sentence is usually part of a text.
- **Holonyms**: A holonym is the opposite of a meronym. *Text* is a holonym of the word *sentence*.
- **Collocate**: A collocation is a set of words that co-occur more often than would be expected by random chance. The individual members of such a set are called collocates. For example, the individual words *code* and *source* are collocates because of their frequent co-occurrence in the compound word *source code*.

Word associations can also vary in strength based on the direction of the association. For example, the word *Eiffel* would be very strongly related to the word *tower*: when someone says *Eiffel*, *tower* immediately springs to mind. However, this relation does not hold as strongly when inverted. If someone says *tower*, words like *building* and *tall* spring to mind much more quickly than *Eiffel*. Similarly, *brick* is related to *tower*, but not to *Eiffel*. As such, word association cannot be treated as a symmetric or transitive relation.

Modelling these different types of word associations computationally is very challenging. There are many ways in which words can be associated. Gathering all of these associations for each individual word in a language is an immense task. In fact, we argue that it is infeasible to encode all such relations in manually constructed ontologies and databases. Two of them are given in Section 2.

Instead, in Section 3, we propose three unsupervised methods for modelling different types of word associations using large amounts of text as a source: a word-embedding model for synonymy, a point-wise mutual information model for word collocations, and a dependency model for common properties of words.

The main goal of this work is to evaluate the performance of the proposed methods. Since we do not have any appropriate annotated data, we test our models directly by humans through a simplified online game called “Codenames”. It is a single-player version of a very popular word association board game of the same name. In short, one player gives one-word hints to some of the words given on the board and the

other player guesses. The game itself and the evaluation procedure is described in Section 4.

In Section 5 we describe methods on how to employ the association measures in the Codenames game. In Section 6 we provide a detailed analysis of the performance of our models and two ensemble models which try to combine all the models together. A concise overview of our findings is given in Section 7. We also mention several promising directions for future research.

2. Association Databases

One of the approaches for computational word association that we considered is the use of ontologies and databases. We could rely completely on the word associations provided by manually entered data to build our models. Even though we finally decided not to use them in our system, we detail two of these approaches below.

2.1. WordNet

WordNet (Fellbaum, 1998) is a collection of synsets grouped into a semantic hierarchy. A synset is a collection of one or more words with the same meaning, i.e. synonyms. Because of the information it encodes, it excels at strict relations such as hypernyms, hyponyms, meronyms and holonyms. This would be a great addition to our application and a fruitful area for future research on computational models of word associations. For example, *continent* would be a useful hint for both *Africa* and *Australia*. However, it falls short when considering more free associations such as *Frodo* and *ring*, which cannot be classified as either hypernym, hyponym, meronym or holonym and is thus not captured in WordNet.

WordNet is not suitable for our purposes because it does not capture as many relations as we would like and is not as extensible as data-driven methods, which are able to capture even pop culture references such as the relation between *Frodo* and *ring*.

2.2. University of South Florida Free Association Norms

The University of South Florida Free Association Norms (Nelson et al., 2004) is a database of free associations containing 72,000 word pairs collected from almost 750,000 responses produced by over 6,000 participants. More than 5,000 stimulus words were tested. While this is a great resource for human responses on word association, it has too many gaps to be suitable for a computational model. Even when we look at all the responses in addition to the 5,000 stimulus words, words that occur in the original Codenames board game such as *Amazon*, *Greece* and *horseshoe* do not occur in the database at all. These gaps can only be filled by repeating the experiment with these words as stimulus words. Moreover, this database exists only for English, limiting the applicability of this approach for other languages.

Although it is not suitable as a basis for a complete computational model, it is still useful as a resource on human word association. The database of word associations could be used to compare how similar the predictions made by a computational model are to human-level associations. While we do not perform this particular comparison ourselves, it could serve as an interesting evaluation metric for future work.

We think ontologies and association databases contain too many blind spots and often fail to encode unorthodox or out-of-the-box relations that would nonetheless be considered valid associations by humans. Building these resources also requires a lot of manual work and knowledge of the language involved, which becomes a recurring cost as semantic shifts occur in the existing vocabulary and new words enter the language. As such, we turn our focus towards automated methods for the extraction of word associations in the rest of this paper.

2.3. Other Related Works

Thawani et al. (2019) propose a novel word embeddings evaluation task by employing a large word association dataset called *Small World of Words* (De Deyne et al., 2018). It contains Word association and participant data for 100 primary, secondary, and tertiary responses to 12,292 cues, collected from over 90,000 participants.[1]

3. Methods

In this section, we propose three data-driven methods that can be used for measuring how much two words may be associated each to other.

We are not aware of any work in which more complex word-association models were built. We know of only one earlier attempt in this area, which is a Master thesis by Obrtlík (2018). However, they use word embeddings, which cover only synonymic relations. We describe this method in Section 3.1. Human word associations are not limited to this type of relation alone. Take, for example, the words *ice* and *cream* in a collocate such as *ice cream*. Therefore, in Sections 3.2 and 3.3, we propose two other methods based on word collocations.

3.1. Word Embeddings

Word embeddings are vector representations of words that are used in neural networks processing of textual data. They capture semantic similarity: words that occur in a similar context have a similar meaning and are grouped together in the word-embeddings vector space. Word embeddings capture synonymy, which makes them useful for word association. To create such embeddings efficiently, Mikolov et al. (2013) introduce the skip-gram model with negative sampling. Since then many additions to this technique have been proposed, such as adding topic information (Liu

[1] https://smallworldofwords.org/en/project

et al., 2015) and deriving the embeddings from dependency-based contexts (Levy and Goldberg, 2014).

For our word-association model, we use word embeddings enriched with subword information as described by Bojanowski et al. (2017). This method is called *fastText* and adapts the skip-gram model with negative sampling to represent a word as a combination of the character n-grams it contains. The benefit of this approach is that the representations of character n-grams are global and shared between all words, so more accurate representations for rare words are obtained. The *fastText* embeddings are available in many languages, which makes it easier to build the same model for other languages.[2]

The pre-trained model provides over 2.5 million word embeddings for English and 600,000 for Czech. We cut down on the size of this collection considerably to limit the computation time needed to compare against all of these embeddings when scoring a word. The model is ordered according to the word frequencies. To avoid clutter and make sure that we do not include words that people might not know, we limit the number of word embeddings in our model to the top $10,000$ words, sorted by their unigram frequency in Wikipedia.

For measuring similarity of two given words a and b represented by word embeddings A and B we compute the cosine distance as follows:

$$cosine\_distance(a,b) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

3.2. Sentence-level Collocations

A different type of word association may be covered by computing word co-occurrences. We can see this when we look at the collocation *Eiffel Tower*. When somebody says *"Eiffel"*, we quickly think *"Tower"*.

To find these collocations we need a large amount of text and a measure of association. The text is taken from the training sections of the CzEng 1.7 corpus (Bojar et al., 2016).[3] CzEng is a large parallel corpus for Czech and English, containing roughly 57 million sentence pairs and over 600 million words. The corpus bundles a large amount of data, including but not limited to text from subtitles, EU legislation, fiction and web pages.

For measuring word collocations we use pointwise mutual information (PMI)

$$PMI(a,b) = \log_2 \frac{p(a,b)}{p(a)p(b)},$$

[2] https://github.com/facebookresearch/fastText/blob/master/docs/pretrained-vectors.md

[3] http://ufal.mff.cuni.cz/czeng/

where $p(a, b)$ is the probability that *b* occurs after *a* and $p(a)$ is the probability of seeing *a*. For practical purposes we make a slight modification to the usual definition of PMI and say that $p(a, b)$ is the number of times *b* occurs **before or after** a. This way the direction of the relation does not matter for the strength of the association.[4]

Experimentally we find that simple bigrams as described above do not form a good model. Finding a good collocate for one word is easily done, but finding good collocates for multiple words, is rarely successful because there is too little overlap between collocates. What is a frequent collocate for one word, is almost never a frequent collocate for another. The problem here is that related words are often separated by function words in the text, which means they do not form a bigram and are not seen as a collocation.

To solve this we propose a collocation model over sentence-level word pairs. We define $p(a, b)$ as the probability that *a* and *b* occur in the same sentence. This provides a much broader scope for co-occurrences, which increases the chance of overlapping collocates when trying to find a high scoring collocate for multiple words. The upside of sentence-level collocations is that the model contains more word pairs, which means we will discover pairs we have not seen in the simple *bigram* method. For the bigrams we have seen before, we expect to get PMI values that are closer to the real distribution.

Our method works with morphological lemmas of words. We introduce a frequency cutoff[5] into our model, if a unigram has a lower frequency than this cutoff, we exclude it. In our experiments, we use a frequency cutoff of 1,000, which means the model does not consider hints that occur less than one thousand times in the corpus. Such a model takes roughly the 16,000 most frequent words in CzEng.

3.3. Dependency-level Collocations

In the sentence-level model, we considered many word pairs, many of those have nothing in common. To reduce noise, we introduce *dependency-level* collocations model, in which the considered word pairs are restricted to words between which there is a dependency relation. We define $\mathtt{count}(x, y)$ as the number of times *y* and *x* occur in the same sentence and have a dependency relation.

[4] *Arthur* will be just as related to *king* as *king* is to *Arthur*. Even though it might be feasible to extract the direction of the relation from the syntactic makeup of the collocation or its syntactic context, this falls outside of the scope of this paper. We instead choose to generalize and say that the co-occurrence of two words counts equally towards either direction regardless of context.

[5] The frequency cutoff is an important factor in the quality of the model. Setting the cutoff too high results in a model that is too general and cannot accurately target any particular word on the board. Setting the cutoff too low will result in very obscure words entering the model, which is problematic if these words fall outside the vocabulary of a player. A cutoff that is too low will also suffer from data sparsity. For example, if a word occurs only once or twice in the data, it has a high PMI value for the words it co-occurs with, even though the real distribution might be much different. In this case, the PMI value is most likely not representative of the actual distribution.

The CzEng corpus we use has already been annotated with syntactic and semantic information. The annotation is separated into the analytical layer and the tectogrammatical layer. The syntactic trees were created automatically by the Treex[6] pipeline. For our purposes, we use the tectogrammatical trees, which exclude function words so content words are related through dependency edges directly. This is very useful for our dependency-level method. We work only with tectogrammatical lemmas and ignore all special tectogrammatical nodes with lemmas starting with a hash sign (#). Additionally, we strip away information about reflexivity of verbs from the lemmas which is encoded with *_se* and *_si*.

The dependency-level collocations are bidirectional, the hint can be both the dependent as well as the head of the dependency relation. Whether the direction of the dependency relation plays a role in the quality of the model, would be an interesting direction for future research.

4. Evaluation

The task of word association is the retrieval of associated lexical items in response to a word prompt. In order to make this task more appealing to participants, we test word association by humans in the context of a word association game called *Codenames* (Chvátil, 2015).[7] The game is available in many languages, but we focus our efforts on English and Czech because these languages are well represented in our group of participants.

4.1. Codenames Board Game

The game is played in two teams of 2 or more players, each team has one spymaster and one or more agents. The game board consists of 25 cards with a word written on it. There are nine cards that belong to the team that starts the game, eight that belong to the opposing team, seven neutral cards and one assassin card, which loses the game for the team that selects it. Both spymasters get to see which cards belong to which team, but the agents do not. Each turn one of the spymasters gives a hint to their agents for one or more cards that belong to their team. The spymaster also gives a number that signals to how many of their own cards the hint is related. The agents then proceed to guess cards until they select one that does not belong to their team or they voluntarily end their turn because they do not see any more cards that are related to the hints that their spymaster has given.

The goal of the game is to turn over all of the cards that belong to your team. As a spymaster, you help achieve this goal by giving hints to your agents that are associated with your own cards and unambiguous as possible. As an agent, you will

[6] http://ufal.mff.cuni.cz/treex

[7] https://czechgames.com/en/codenames/

Figure 1. A regular game of Codenames

try to turn over as many cards of your own team as possible using the hints given by your spymaster, without selecting any cards that do not belong to your team and avoiding the assassin at all cost. The game ends when either team has turned over all of their cards, in which case that team wins, or one of the teams has selected the assassin in which case that team loses.

A regular game of Codenames along with the board pieces used to play the game can be seen in Figure 1. The blue and red spy cards are used to cover words that have been selected during the game. The card in the top left corner is visible only to the spymasters and shows which cards belong to which team. There are some restrictions to the hint that the spymaster can give. The hint has to be one word and cannot be any morphologically related form of a visible word on the board. For example, if one of the words on the table is *fly* and it has not been selected yet, it disallows hints such as *fly*, *flown*, *flight* and *butterfly*.

The aim of the spymaster is to provide hints that are related to the cards belonging to their team. When playing the game with other people, it can be quite challenging to give a good hint for multiple words, say *mozart, 3*, and even more so to correctly guess *symphony*, *concert* and *piano* when you get such a hint as an agent.

4.2. Our Implementation of the Game

There also exists a two-player variant, which is detailed in the rule book of the board game.[8] For our purposes, we adapt this two-player version into a version for one human player (the agent) and a computer (the spymaster) who gives hints to the human player. The game is made more regular by putting the computer and the player on the same team and introducing a dummy team that opposes them. The

[8] `https://czechgames.com/files/rules/codenames-rules-en.pdf`

Figure 2. Screenshot of our online implementation of the game.

dummy team turns over one of their own cards at random during their turn and then passes the game back to the player.

Our implementation of the game as a tool for evaluating word-associations is available online[9] and the code is available on GitHub[10]. It includes code for running the web application, generating models and the anonymized data from our experiments. A major focus while designing the application was to increase the number of games played, so we can collect more data for evaluation.

A screenshot of our implementation is given in Figure 2. We use blue for the players own cards, red for the enemy team's cards, yellow for the neutral cards and black to indicate the assassin. Gray cards have not been selected by the player yet and can be of any type. The status bar on the top right shows the name of the AI that generates the hints. On the bottom left the player can see the current hint as well as a history of the previous hints provided by the AI. On the bottom right we show the current turn, the score that the player would achieve if they guessed all of their cards in this turn and the number of own cards that the player has left. "End turn" allows the player to end their turn without selecting an incorrect card.

4.3. Evaluation Metrics

We establish micro-averaged precision, recall and f-score for measuring the quality of individual models tested on the Codenames game. The true positives are the cards

[9] `https://ufal.mff.cuni.cz/david-marecek/codenames/`

[10] `https://github.com/mderijk/codenames`

clicked by the player which were their own, false positives are the cards that the player clicked which were not their own, and the false negatives are the cards that the player should have clicked but didn't.

We also provide a baseline win rate for completeness. Although this statistic is useful, it also demands much more data to arrive at accurate results. As such, it is less suitable for our purposes since gathering enough data to compute this measure would require a lot of time. We therefore do not consider this metric for our models. By considering the decisions taken by the player instead of tallying how many games were won and lost, we are able to provide more accurate metrics and evaluate models using fewer games.

5. Aggregation of Scores

The three methods proposed in Section 3 give us a similarity score: a number that describes how much two given words are associated with each other. However, for the purpose of evaluation, we will need to generate hints that are associated potentially with multiple words at once and not associated with the enemy words. Therefore, we have to find a way to aggregate these similarity scores into one number, which we will call the **aggregate score**. We also refer to aggregation as *weighting* because of the weights that are applied to the similarity scores when they are fed to the aggregation method.

Our general strategy is to split the similarity scores into four groups: own, enemy, neutral and assassin. Each containing the similarity scores for the hint and a word from the player's own cards, the enemy's cards, the neutral cards or the assassin cards, respectively. Another categorization we make is a more simple one. We divide the words on the board into positive and negative words, where the player's own words are the positive words while all other words are the negative words.

The simplest aggregation method is to sum the similarity scores for all the positive words. This works well because the more related a word is to the hint the more it contributes to the aggregated score. The problem with this approach is that it does not take the negative scores into account at all. For a hint with 3 positive words with a score of 10, there might also be a negative word with a score of 15. This is problematic because a player will be very likely to choose the negative word over one of the positive words, thus making an incorrect decision and wasting a turn. This last point reveals an important point in the decision-making process: selecting certain types of cards is worse than others. Thus, when choosing a hint, we should also factor the type of card into the equation.

For this purpose, we introduce **weights**. These weights consist of four integers, one for player, enemy, neutral and assassin scores. The similarity scores for each category are multiplied by their category-specific weight before they are fed to the aggregation

method.

$$similarity\_score(hint, word) * weight(category(word))$$

For example, if the model is considering the hint *apple*, and there is the positive word *pie*, which has a PMI score of 14.051 for *apple*, and a negative word *tasty*, let's say the assassin, with a similarity score of 9.468. Now we apply the weights, say 1 for positive words, and 2 for negative words. The similarity score of the assassin in relation to the hint becomes 18.936 instead of 9.468, and the score for the positive word is still 14.051. The aggregation method can then decide that $18.936 > 14.051$ and reject the hint because the risk that the player will select the assassin is too high.

In the next sections, we show several weighting schemes which can be applied to the relatedness scores of the models (PMI and cosine distance) to find the best hint in a game. The different weighting schemes are different functions for aggregating the similarity scores of the words in a game given a potential hint. They give different priorities to different types of cards. All our models use the same weights: the positive and the neutral words are multiplied by 1, the negative words by 1.2, and the assassin word is multiplied by 2. We would really like to avoid the assassin because this ends the game immediately, hurting our recall considerably. We also want to avoid clicking enemy cards because it costs the player a turn. Clicking a neutral card is not as bad because it is similar to getting a new hint by ending the turn and we also get to eliminate another card from play without penalty.

5.1. Combined Maximum

To calculate CombinedMax we first determine a threshold by taking the maximum similarity score from the list of negative words N. We then sum the scores from the list of positive words P that are above the threshold to get the aggregate score.

$$CombinedMax(P, N) = \sum_{x}^{P} \begin{cases} x & \text{if } x \geq \max(N) \\ 0 & \text{otherwise} \end{cases}$$

This way a hint only scores well if it relates to many words that are more similar to the hint than the most similar negative word. This implicit negative threshold is the most distinctive feature of this model.

This method is very sensitive to the weights we apply to the negative words. If we set the weights too high, this method is very good at finding the blind spots of a model. For example, for a collocations model, it might find a hint for which there is one positive word with a high PMI score while the rest of the scores are zero. The reason that this happens is that when the weights are high, there are very few positive words that can cross the implicit negative threshold if there is a negative word with a score higher than zero. Therefore, the chance that the model will find a hint for which all words have a PMI value of zero except for one positive word, is very high.

5.2. Mean Difference

The Mean Difference method simply takes the difference between the averaged score of the positive cards P and the averaged score of the negative cards N.

$$MeanDiff(P, N) = \frac{\sum_x^P x}{|P|} - \frac{\sum_y^N y}{|N|}$$

The problem with Mean Difference arises when there is a high variance within one of the classes. For instance, it does not account for situations where there is one negative word that has a really high similarity score with the hint and overshadows the positive words leading the player to click an incorrect card. As such, it is not as good at the start of the game when the mean can obscure negative words with high similarity if it is surrounded by many negative words with low similarity to the hint. Near the end of the game, this method becomes much better because each peak in similarity of individual words is reflected more strongly in the mean of either class.

5.3. Most Similar

This weighting method is different from the others since it does not aggregate the similarity scores of the positive and negative words. Rather, the most_similar method in Gensim[11] (Řehůřek and Sojka, 2010) works by performing vector arithmetic, adding the embeddings of the positive words to each other and subtracting the negative vectors. The method then returns the words whose vectors are closest to the resulting point.

This method performs well in targeting positive words. However, because it subtracts negative vectors and literally "stays away" from the negative words, it can easily suffer from one simple mistake: including too many negative words. In other words, it assigns too much weight to negative words and starts generating hints that are specifically not referring to negative words, rather than providing hints that are similar to the positive words. To resolve this issue we let it take only the assassin word into account for the negative words.

5.4. Top-n

The top-n ($n \in 1, 2, 3$) methods are an adaptation of the CombinedMax function. The formula is the same, except for the fact that P is restricted to the n highest values in P. The distributional characteristics of these functions are very interesting because we have some control over its behaviour by setting n. If we take the Top-1 method, we will simply get the hint with the highest similarity score among all pairs of hint and

[11] https://radimrehurek.com/gensim/

target words. This results in hints with very large similarity scores which are usually highly associative. The Top-2 method is generally more mixed, with one hint with a high similarity score and one hint with a moderate similarity score. And if we look at the Top-3 method, we often get three words with moderately high similarity scores. Of course, there is a lot of variation depending on the number of words still on the board. The top-n methods are still similar to CombinedMax in the sense that they only have an upper bound n and no lower bound. A top-n is also allowed to give hints for less than n words.

6. Results

In this section, we show the evaluation results of the proposed word association models and aggregation methods. Since we used an iterative approach in the design of our methods, we dedicate one section to each iteration of models. In Section 6.1 we establish a baseline for our models by Monte Carlo simulation. Then we analyze the results of the first test run in Section 6.2 and discuss the improvements we made to our models in Section 6.3. The Top-n models are combined in Section 6.4. Finally, we discuss the performance of a combination of a word embedding and collocation model in Section 6.5.

6.1. Random Baseline

The baseline is set by a scenario in which hints do not provide any help to the player whatsoever, which is equivalent to the situation where there are no hints at all and cards are chosen randomly by the player.[12] We perform a Monte Carlo simulation of playing the game by repeatedly selecting cards at random. We simulate 10 million games in this way, from which we obtain the results displayed in the first row of Table 1. The baseline for the win rate, the chance to win the game by selecting cards at random, is 0.39%. This is a very low number, on average this means the player wins only one game out of 257 games.

If the generated hints provide any semantic meaning related to the player's words more so than to the other team's words, we would expect the average win rate to be higher than the baseline. The same can be said for precision, recall and f-score.

6.2. Initial Models

The results for our initial models are shown in Table 1. All of our models perform above the baseline, which means they are better than random chance. Globally, it seems that better results were obtained for Czech. We hypothesize that this is caused

[12]The end turn button that is present in the game is not modelled as a possible action because a player clicking cards randomly does not gain any additional information from getting a new hint, while the opposing team does have the opportunity of turning over an additional card.

Setup		Player decisions					
		CZ			EN		
	Aggregation	P	R	F_1	P	R	F_1
Random baseline							
	—	0.389	0.339	0.362	0.389	0.339	0.362
Word embeddings							
	MostSimilar	0.616	**0.753**	**0.677**	0.563	**0.607**	0.584
	CombinedMax	0.558	0.578	0.568	**0.567**	0.606	**0.586**
Sentence-level collocations							
	CombinedMax	0.507	0.490	0.498	0.500	0.466	0.482
Dependency-level col.							
	CombinedMax	**0.629**	0.654	0.641	0.547	0.497	0.521
	MeanDiff	0.575	0.636	0.604	0.546	0.544	0.545

Table 1. Micro-averaged precision, recall and f-score for our initial models.

by the fact that our group of English speaking players consists mostly of second language learners, while most of the players for Czech were native speakers.

Comparing the f-scores for Czech, we can see that the best method is the WordEmbeddings with MostSimilar aggregation (0.677), followed by the two Dependency level collocations models (0.641 and 0.604), and then the WordEmbeddings model with CombinedMax aggregation (0.568). For English, the best models are both aggregations of WordEmbeddings (0.584, 0.586), followed by Dependency-level collocations (0.521, 0.545).

The Sentence-level collocations were outperformed by the other two models for both languages with f-scores of 0.498 and 0.482. Even though we expected that the lack of data for the dependency model might hurt its performance, it seems that the constraints on the word pairs lead to more accurate results. In the following evaluations, we do not continue with the Sentence-level collocation model, since it did not prove to be promising. Although it contains many more word pairs, dependencies seem to capture more accurate relations between words thus producing better hints.

6.3. Improved Models

In this section, we improve our dependency and word embedding models by introducing new aggregation methods. We start with the Top-1 aggregation method,

Setup		Player decisions					
		CZ			EN		
	Aggregation	P	R	F_1	P	R	F_1
Random baseline							
	—	0.389	0.339	0.362	0.389	0.339	0.362
Sentence-level collocations							
	CombinedMax	0.507	0.490	0.498	0.500	0.466	0.482
Dependency-level col.							
	CombinedMax	0.629	0.654	0.641	0.547	0.497	0.521
	MeanDiff	0.575	0.636	0.604	0.546	0.544	0.545
	Top-1	**0.722**	0.633	0.675	**0.693**	0.678	**0.685**
	Top-2	0.621	**0.778**	**0.691**	0.646	**0.711**	0.677
	Top-3	0.552	0.644	0.595	0.655	0.611	0.632
Word embeddings							
	MostSimilar	0.616	0.753	0.677	0.563	0.607	0.584
	CombinedMax	0.558	0.578	0.568	0.567	0.606	0.586
	Top-1	**0.789**	**0.789**	**0.789**	**0.768**	0.735	**0.751**
	Top-2	0.608	0.690	0.647	0.614	0.735	0.669
	Top-3	0.574	0.626	0.599	0.667	**0.786**	0.722

Table 2. Micro-averaged precision, recall and f-score for our improved models.

which always tries to find a hint for only one word.[13] At this point, we introduce a number that shows how many target words the hint relates. We show this number to the player together with the hint for all models other than the initial models evaluated in Section 6.2. For the Top-2 and Top-3 methods, it might be the case that they do not manage to find a hint for the intended amount of words. In such cases, the number provided by the model will reflect the actual number of words that it has managed to target with the given hint. The results are shown in Table 2.

During testing, we notice a major effect of knowing the number of words that the system is hinting at. The player now knows when they have exhausted a hint and can stop using it. If the hint was for only one word and the player has selected this card, they will now press the end turn button to gain a new hint whereas previously they might have continued guessing using the same hint which would have been similar to random guessing.

[13]It is not possible to win the game this way through association alone because the maximum number of hints a player can get is 8, which can be achieved by manually ending the turn 7 times in a row. Even though this is not as fun for our participants, it provides a useful baseline.

For the Dependency model, we can see that the Top-1 and Top-2 models provide a significant improvement over the previous models for both Czech and English. The Top-3 model outperforms the original CombinedMax only for English, while it is significantly worse for Czech. The Top-2 dependency model achieves a noteworthy recall compared to the other dependency models. In this case, high recall means that players on average get much closer to turning over all of their cards and winning the game. The Top-1 models achieve the highest precision across the board. This is not surprising, it is easy to give a good hint for one word, but much harder to give a good hint for two or more words and still have the player guess both of them.

For the WordEmbedding models, we see that the Top1 model performed best for both Czech and English. The Czech Top-3 model performs poorly similar to the Dependency models, however, the English Top-3 model performs very well. The Top-2 word embedding models are considerably worse than their Top-1 counterparts, contrary to what we see for the dependency models.

We observe across all models that the Top-1 model has higher precision than recall and for the Top-2 and Top-3 models this relation swaps and the recall is higher than the precision. The only anomaly is the English Top-3. Curiously, its precision is much higher than for the Czech model.

6.4. Threshold Models

We would like to build a model that can give hints for 1, 2, and 3 words depending on the situation. Naturally, we would like to prioritize hints that target more words, so we propose a threshold model which gives hints using the Top-3 model while these hints score above some threshold and switches to the Top-2 model when no hint from the Top-3 model passes this threshold anymore. Similarly, it will switch to the Top-1 model if the score threshold for the Top-2 model can no longer be surpassed by any hint. In order to build this model, we will first need to determine adequate thresholds.

To determine these thresholds we studied the decisions made by players playing with the Top-1, Top-2, and Top-3 dependency models. For each method, we manually select a threshold value that reasonably separated hinted positive cards from the others.

We create the threshold models Top-N for both Dependency-level collocations and Word embeddings. A model consists of three submodels which we have already tested individually so we can see if there is an improvement. Hints are chosen by querying the Top-3, Top-2 and Top-1 models in that order and selecting the first hint from the model that passes its respective threshold, defaulting to the Top-1 model if none of the thresholds is passed.

Table 3 compares the results of the Top-N models and individual models. The Dependency model performed very poorly, it did not manage to outperform even the worst individual model, which was the Top-3 model. The performance of the English model is exceptionally bad when contrasted with the performance of its worst

Setup	Player decisions					
	CZ			EN		
Aggregation	P	R	F_1	P	R	F_1
Random baseline						
—	0.389	0.339	0.362	0.389	0.339	0.362
Dependency-level col.						
Top-1	**0.722**	0.633	0.675	**0.693**	0.678	**0.685**
Top-2	0.621	**0.778**	**0.691**	0.646	**0.711**	0.677
Top-3	0.552	0.644	0.595	0.655	0.611	0.632
Top-N	0.598	0.585	0.591	0.570	0.476	0.519
Word embeddings						
Top-1	**0.789**	**0.789**	**0.789**	**0.768**	0.735	**0.751**
Top-2	0.608	0.690	0.647	0.614	0.735	0.669
Top-3	0.574	0.626	0.599	0.667	**0.786**	0.722
Top-N	0.673	0.623	0.647	0.738	0.679	0.707

Table 3. Micro-averaged precision, recall and f-score for the threshold models.

submodel and performs much worse than the Czech model in this regard. We hypothesize that the lower threshold for the English Top-3 model has contributed significantly to this poor performance. For the Czech model, there was a much smaller gap between the threshold of the Top-3 and Top-2 model. In addition, we can say that the threshold method has not had the desired effect. While we would expect that the Top-N model would perform equally or better than the worst performing model, our English dependency model performed much worse than the worst individual model.

For the WordEmbeddings model, the picture looks slightly better. The Top-N models perform worse than the best individual models, but better than the worst individual model. While this performance is certainly better than that of the Top-N Dependency model, it does not improve over the best individual model in any way. When we look at Figure 3 we see that the threshold model did not prevent the player from selecting cards with low similarity scores. The number of positive cards selected which were not hinted at in the current turn is much higher, which explains why the model has higher precision than the Top-2 and Top-3 models. Therefore, we conclude that the threshold system successfully improves the precision of the model. However, this happened at the cost of the recall. And it still performs worse than the Top1 model across all statistics.

All Top-N models suffered in terms of recall when compared to the individual models. None of them has higher recall than the lowest recall of any of their submod-

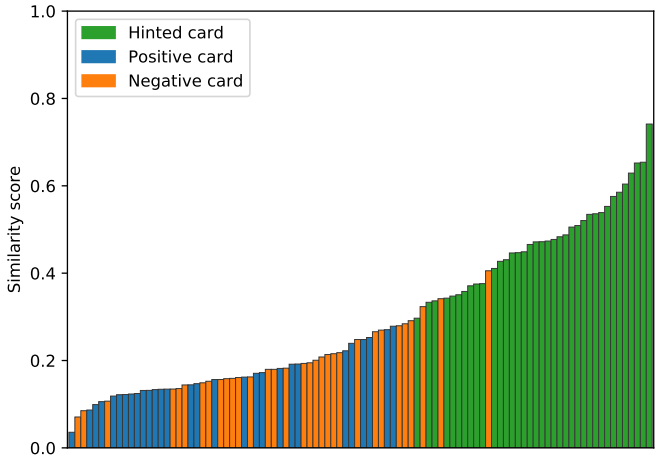

Figure 3. Similarity scores for each card clicked by players across several games for the Czech Top-N word embedding model

els. Precision, on the other hand, increased considerably in comparison to the Top-3 and Top-2 models.

The threshold model did not live up to expectations because it did not prevent the player from clicking on cards with low similarity scores in regards to the hint. We suspect that the thresholds we selected were far from optimal and a different ensemble approach might achieve better results. Finding a good way to combine the Top-1, Top-2, and Top-3 methods to achieve the same or better performance than either of the individual methods is an interesting direction for future research.

6.5. Combined Models

Lastly, we would like to test a model that combines both the dependency collocations model and the word embedding model. Since the threshold system turned out to be a poor ensembling method, we have to consider a new way in which we can combine our models. One method is to find a mapping between PMI values and cosine similarity. However, one of these measures is normalized and the other is not and their scales are radically different, so this relationship can be hard to find through trial-and-error and is in the worst case non-linear. Instead, we choose to perform ensembling through mutual agreement, where we let both models predict hints until

Setup	Player decisions					
	CZ			EN		
Aggregation	P	R	F_1	P	R	F_1
Random						
Baseline	0.389	0.339	0.362	0.389	0.339	0.362
Dependency-level col.						
Top-N	0.598	0.585	0.591	0.570	0.476	0.519
Word embeddings						
Top-N	0.673	0.623	0.647	0.738	0.679	0.707
Dependency col. & Word emb.						
Top-N combined	0.642	0.678	0.659	0.711	0.697	0.704

Table 4. Micro-averaged precision, recall and f-score for the combined Top-N dependency and word embedding model.

one of the models gives a hint that the other model has also predicted for the current board state.

We expect an ensemble model that combines word embeddings and collocations to perform better than the individual models since they both model different types of association. Word embeddings capture similarity while collocations usually capture other types of relations. Combining the best of both models should lead to better results.

We test an ensemble model that combines the Top-N dependency and Top-N word embedding models described in Section 6.4. In Table 4 we can see the results of combining dependency and word embedding models by finding hints through mutual agreement between models. The combined model performed similarly to the best models included in them with an f-score of 0.659 for Czech and 0.704 for English. The f-score of the Top-N word embedding model is slightly lower for Czech (0.647) and slightly higher for English (0.707).

Although these results are promising, they do not significantly improve the results of the models they combine. The model is successful at mimicking the performance of the best submodel, but it does not select the best hint from either model depending on what is best in a given situation. This is due to the ensembling method used. As such, more research on good ensembling methods is needed to find models that do improve above the performance of their internal parts.

In Table 5 we show the number of games played and the number of decisions made for each model. The number of decisions for a model is the sum of all the cards clicked by players in the games played with that model.

Setup		Player decisions CZ		EN	
	Aggregation	#G	#D	#G	#D
Sentence-level collocations					
	CombinedMax	17	148	62	520
Dependency-level col.					
	CombinedMax	17	159	68	556
	MeanDiff	18	179	67	601
	Top1	10	79	10	88
	Top2	11	124	10	99
	Top3	10	105	10	84
	TopN	10	92	10	86
Word embeddings					
	most_similar	22	242	65	630
	CombinedMax	25	233	77	741
	Top1	10	90	13	112
	Top2	14	143	13	140
	Top3	11	108	13	138
	TopN	10	98	11	103
Dep. collocations & WE					
	TopN - mutual	10	95	11	97

Table 5. The number of games played (#G) and the number of decisions (#D) made for all models tested.

7. Conclusions

We have provided both a theoretical and practical framework for the evaluation of computational models of word associations. We started out by establishing a baseline for the task of Codenames with a single human player. After this, we explored several methods all of which performed well above the baseline. The restriction on syntactically dependent words proved a definite improvement over broad sentence-level word pairs for the collocation model for both evaluated languages.

Large improvements to our model were made by aggregating the similarity scores of the words on the board and weighting them more cleverly. Our best Dependency models achieved an f-score of 0.691 for Czech and 0.685 for English. The Word Embedding models based on the same aggregation technique, in turn, outclassed these models with f-scores of 0.789 and 0.751 for Czech and English, respectively. The model that got closest to helping the player turn over all their cards, was the Top-2 Dependency model for Czech with a recall of 0.778. For English, the best model in this regard was the Top-3 Word Embeddings model with a recall of 0.786.

We made several attempts to build ensemble models that combine the best performing models to boost their performance. We were not successful in this regard, our Top-N dependency model achieved f-scores of 0.591 and 0.519 for Czech and English respectively. The Top-N word embeddings performed better, with an f-score of 0.647 for Czech and 0.707 for English, but neither outperformed the best individual Top-n model for their respective language. A final attempt at combining dependency and word embedding models by finding hints through mutual agreement between models performed similarly to the best models included in them with an f-score of 0.659 for Czech and 0.704 for English. Although these results are promising, we believe that many better ensembling methods still remain.

We have shown that both dependency-level collocation models and word embedding models can provide hints of considerable quality, given the right constraints. Dependency models manage to capture several types of relations between words which the player is able to pick up on, while the word embedding models excel at finding semantically similar hints.

8. Future Work

We have provided an overview of only the most basic methods and we believe that many improvements can still be made to achieve better performance on the Codenames word association task. For example by finding better ensemble methods to combine models that give hints for a different number of words, as well as successfully combining models of different types such as collocation and word embedding models.

The methods we use are themselves simple baselines for the technique that they are based on. There exist many more measures of association other than pointwise

mutual information (see Pecina (2010) for an extensive list of such association measures) and there have been many improvements in recent years over the fastText word embeddings that we tested, many of which might surpass our best word embedding models when compared. This could be a fruitful direction for future research.

The same framework can be used to analyze the effect of time taken between receiving the word prompt and making a decision. We did not incorporate any timing mechanism in our application, so it is not possible to extract this type of information from our dataset. However, it is easy to modify the application and record this data as well, so this is nonetheless an interesting avenue for future work.

While this paper was mainly focused on the computational side of word association, it must be noted that a human baseline for the Codenames word association task would be very useful to give more context to the results achieved on this task. Similarly, comparing the predictions made by the models to human-level word associations would be a useful direction in this area.

Acknowledgements

This work has been supported by the grant 18-02196S of the Czech Science Foundation. We would also like to thank our testers, who helped us with evaluating the proposed models by playing the online game and thank the anonymous reviewers for many insightful comments and suggestions.

Bibliography

Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. ISSN 2307-387X. doi: 10.1162/tacl_a_00051.

Bojar, Ondřej, Ondřej Dušek, Tom Kocmi, Jindřich Libovický, Michal Novák, Martin Popel, Roman Sudarikov, and Dušan Variš. CzEng 1.6: Enlarged Czech-English Parallel Corpus with Processing Tools Dockered. In Sojka, Petr, Aleš Horák, Ivan Kopeček, and Karel Pala, editors, *Text, Speech, and Dialogue: 19th International Conference, TSD 2016*, number 9924 in Lecture Notes in Computer Science, pages 231–238, Cham / Heidelberg / New York / Dordrecht / London, 2016. Masaryk University, Springer International Publishing. ISBN 978-3-319-45509-9. doi: 10.1007/978-3-319-45510-5_27.

Chvátil, Vlaada. Codenames. Czech Games Edition, 2015.

De Deyne, Simon, Danielle Navarro, Amy Perfors, Marc Brysbaert, and Gert Storms. The "Small World of Words" English word association norms for over 12,000 cue words. *Behavior Research Methods*, 51, 10 2018. doi: 10.3758/s13428-018-1115-7.

Fellbaum, Christiane, editor. *WordNet: an electronic lexical database*. MIT Press, 1998. doi: 10.7551/mitpress/7287.001.0001.

Gough, Harrison G. Studying creativity by means of word association tests. *Journal of Applied Psychology*, 61(3):348–353, 1976. doi: 10.1037/0021-9010.61.3.348.

Jung, Carl G. The association method. *The American journal of psychology*, 21(2):219–269, 1910. doi: 10.2307/1413002.

Levy, Omer and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, 2014. doi: 10.3115/v1/P14-2050.

Liu, Yang, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. Topical word embeddings. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

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

Morkovkin, V. V. *Ideographic Dictionaries*. 1970.

Nelson, Douglas L., Cathy L. McEvoy, and Thomas A. Schreiber. The University of South Florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, & Computers*, 36(3):402–407, 2004. doi: 10.3758/BF03195588.

Obrtlík, Petr. Computer as an intelligent partner in the word-association game codenames. Master's thesis, Brno University of Technology, Brno, 2018.

Pecina, Pavel. Lexical association measures and collocation extraction. *Language resources and evaluation*, 44(1-2):137–158, 2010. doi: 10.1007/s10579-009-9101-4.

Řehůřek, Radim and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. URL `http://is.muni.cz/publication/884893/en`.

Thawani, Avijit, Biplav Srivastava, and Anil Singh. SWOW-8500: Word Association task for Intrinsic Evaluation of Word Embeddings. In *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP*, pages 43–51, Minneapolis, USA, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-2006. URL `https://www.aclweb.org/anthology/W19-2006`.

Address for correspondence:
David Mareček
`marecek@ufal.mff.cuni.cz`
Institute of Formal and Applied Linguistics, Faculty of Mathematics and Physics, Charles University, Malostranské náměstí 25, 118 00 Praha, Czechia