Creating artificial intelligence for the board game Codenames

Jan Hynek, Viktoriia Kariagina

January 2017

1 Introduction

Since early times of the computing machines, people tried to learn them to play different board games. And with their advancement, machines started to beat their masters. In 1997, it was the first time when machine, namely IBM Deep Blue, won against world chess champion Gary Kasparov. And recently, Google DeepMind team developed AlphaGo - program able to defeat current world champion Lee Se-Dol in the game of Go, which is considered far more difficult for computers to grasp. [1]

We decided to try something similar. We took different kind of game - popular board game called Codenames. It is a game with words, and it requires other kind of intelligence and creativity than traditional games such as Go and Chess.

It is a game for two teams - red and blue. There is 25 cards drawn from the deck of 400 words and put on the table. Both of the teams have 9 of the 25 cards marked. Aim of this game is to guess the all words belonging to the team. It is quite difficult as only one person from each team can see, which are marked. This person is also the only one who can give hints composed from a single word only (we will call him "hinter" further on). The rest of the team must guess, which words the "hinter" meant and point at them. First team, which guesses all of their words, wins.[1]

The key to the game lies in the hints. They must be general enough to associate more words together, but not too much to wrongfully connect other words which belong to the other team. From the logic of the game it can be seen that "hinter" that successfully connects as most words as possible gains advantage against the other team and therefore has higher chances to win.

And there lies our motivation to create such AI, which could theoretically outpower people in giving such hints. It quite often takes a lot of time to find good association that would nicely connect several words. It might be handy to input several of these words inside the script and get a hint for the game. And if such AI would be created, it could be possible to create mobile application.

[1] More about the rules: https://www.youtube.com/watch?v=J8RWBooJivg

And only a small step further, it could be possible to create mobile game and potentially monetize the whole idea.

2 Literature Review

Vizualisation of association words has been widely discussed and applied by many researchers and has been a focus for multiple investigations in this area. Association is a powerful data analysis technique that appears frequently in data mining literature. Xiao and Toivonen (2016) [2] propose a novel approach, LayerFolding, which selects nodes at increasing distances from the given node, according to their relatedness to it. This relatedness is calculated based on the shortest ways that are potentially consistent. Authors show that LayerFolding provides 79 per cent recall and exceeds performance of popular measures used before.

Researchers have been developing also many tools to visualize association rules for years, some of these tools can manage to work with most of association rules, but none of them could manage rules with multiple explanatory variables. Wong et al. (1999) [3], in their article were the first ones to present a novel vizualization technique, which solved many of the problem with providing results even with a large data.

In spite of this, there are still quite a lot of articles which are concerned with associations text mining, but it seemed interesting for us to apply word2vec algorithm as that algorithm is better in catching in the meanings as well. Word2vec was developed by the team of Google researchers led by Tomas Mikolov.

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Figure 1: Examples of five types of semantic and nine types of syntactic questions.

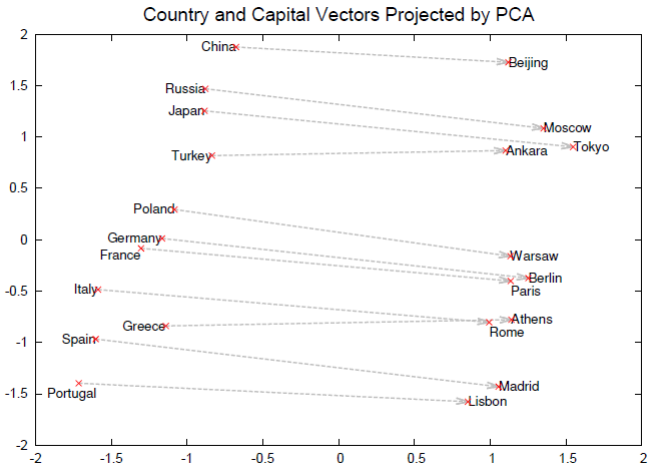

Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities.

Mikolov et al. (2013) [4], they introduced two novel model architectures for computing continuous vector representations of words from very large data sets. For measuring quality of the word vectors, they defined a comprehensive test set that contains five types of semantic questions, and nine types of syntactic questions. In the Figure 1, we can see two examples from each category.

In the other article published by the team of Mikolov [5], published also in 2013, introduced the Skip-gram model, which is an efficient method for learning high quality vector representations of words from large amounts of unstructured text data, as a result they found out that the word vectors can be somewhat meaningfully combined using just simple vector addition. Figure 2 shows ability of the model to automatically organize information and learn implicitly the relationships between words.

3 Methodology

3.1 Using tm package

When choosing the datasets, our first idea was that as the game often uses references from popular culture, therefore it might be useful to use reddit.com comments, popular site where people often discuss such topics. [6]

Our other idea was that people often use terms which have some kind of hierarchy - good hint for "Greece" and "England" would be "Europe", and for "Europe" and "Asia" would be "Continent". Therefore, we also used several different datasets composed from text obtained from wikipedia. We used first

billion characters from wikipedia dump[2] and we were also able to obtain our own dataset composed from list of good articles [7]

In the beginning of computing, we tried to use "tm" package, used for typical text mining. We obtained Document - Term matrix from several reddit comments, then we applied both k-means and hierarchical clustering. Idea behind was that we would create several levels (around 5) of clusters and try to find most similar words in a following way:

- algorithm A: making hint
  - make list of most frequent word associated with every word from the list of words from codenames
  - make intersects for every two words
  - rank them by occurence
- algorithm B: which words from the table should be connected
  - make several layers of clusters (circa 5) for every of the 400 words by dtm
  - from the list of the 9 words for the blue player, if there is word from the first level of cluster, make guess for these two words
  - if no word is found, find some word in higher level cluster

This method soon found its limits - every word then was only in one specific cluster. But the author of the game selected these words in such a way that they often have several very different meanings - e.g. "well", "press" or "square". It would be impossible to connect these words in every possible meaning. We tried to put the meanings of the words in two dimensional plane, even though the meanings of these words might have much more dimensions.

3.2 Application of word2vec package

Therefore, we decided to use word2vec algorithm for text and association mining, which we applied on several datasets. We used its skipgram method, which is useful in associating common words with a given word.

This algorithm takes huge corpus as an input and then after some calculations, creates n dimensional space, where every vector represents word obtained from the corpus.

We first had to tokenize the dataset, strip the punctuation and make all the words lowercase. Then, we pruned the vocabulary - omitted words with low occurence and also omitted stopwords.

Then, we created term-cooccurence matrix - such matrix which shows us, how often words happens to be close to each other. We used skip - quintgram

[2]suggested here: https://code.google.com/archive/p/word2vec/

mechanism. This mechanism accounts for the occurence of the word, irrespective of its position, in the maximal distance of 5. [3]

Then, algorithm obtained in the package text2vec calculated vectorspace from the TCM. [8] We decided to run the word2vec algorithm, skipgram method with 100 dimensionality and 20 epoch training. In comparision with results from google, we had less dimensions and smaller datasets.

3.3 Creation of the game mechanism

When we succesfully trained the data we decided to obtain simulation of the game. That was quite easy, as the basic mechanism works is really simple:

- Sample 25 words from base of 400
- Sample 9 out of these for the blue team
- Sample 8 from the rest for the red team
- Sample one from the rest for the killer[4]

3.4 Choosing words for finding associations

To simplify our problem, we focused only on the words chosen for the blue team. As we already have vectors representing words, we now need to calculate how far these vectors are from each other. We could either calculate euclidean distance. But there appears other, computationally more handy way - cosine similarity. [9]

We therefore created list of the highest cosine similarity for every word from the list. This is already embedded inside the object "game" (see code).

To choose the best combination to connect, we specify set of thresholds and use following mechanism:

- Is the fifth distant word in this set over the given threshold?
- if yes, choose this set to try to find the best word[5]
- if no, move to another set
- if no set satisfies this rule, try the same mechanism but for the fourth distant word until you get to the second distant.

[3] explained here: http://bit.ly/2kJDbn4

[4] card, which if chosen, ends the game and the opposite team wins

[5] It seems to me at this moment that this algorithm needs rewriting, as there might be better set, but the algorithm chooses the first one

3.5 Finding associations

So now we know which words to connect. To find the right hinting word, it must be related to all of these words. We created two functions to solve this problem.

- function 1
  - take all vectors of the words which you decided to connect
  - add them together[6]
  - find the words with lowest cosine similarity to the resulting vector
- function 2
  - take n words with the lowest cosine similarity for each of the words chosen
  - find intersect from these words and simply add their cosines together
  - find the words with the highest resulting value

4 Results

4.1 Model training

As we had two different approaches when choosing data, we decided that it would be interesting to compare these approaches using similar relationship tests mentioned in Figure 1. We used five of them in our test, and also added 4 new just to see how the trained model works. Results from the test are in Appendix.[7] What seemed interesting that model trained on wikipedia dataset seems to be better in semantic realtionships (was able to find Oslo as an capital for Norway), on the other hand reddit dataset performed better in syntactic relationships (correctly chose easy for its superlative easiest).[8]

4.2 Codenames AI

We ran our algorithm on 13 randomly chosen games out of which only 10 surpassed artificially chosen thresholds, based on best practice. Results are in appendix, and words that would make good hints for at least two chosen words are bolded, and somewhat good hints are in italics. Such qualification does not have any rigorous rules, all is based only on our intuition.

What seems interesting is that the first algorithm chooses very often common words with low semantic value, which were not omitted by stopwords() function.

[6] We also averaged these vectors. But the results stayed the same, because length of the vector does not affect the cosine.

[7] Interesting thing is that when when the model with reddit data was trained with 50 dimensions, it was very inaccurate. On the other hand, wikipedia dataset provided similar results both with lower and higher dimensionality

[8] Quite funny is observation "Merkel to Germany is like typhoon to Japan"

This can be seen especially on the results based on wikipedia dataset. Even though even these stopwords could be omitted as well, it won't probably improve the effectiveness of this algorithm by much. This dataset produced good hint for only three of the overall 20 tries, all on wikipedia dataset.

Second algorithm seems to be more successful. When trained on the wikipedia dataset, it was able to find good hint in 6 out of 10 times. And in 3 other cases out of the rest, it produced hint that was somewhat connecting these words and cooperating player would probably identify at least one of these words.

Regarding the datasets, wikipedia performed much better. The cause might be seen in the previous testing, where was seen that reddit dataset performed better in syntactic relationships, but underperformed in semantics. As this game is based primarily on semantics of the words, we can observe similar behaviour happening here.

5 Conclusion and further work

We can see that in the case of the second algorithm trained on the wikipedia dataset, we could create fully autonomous AI. Still, there would be a lot of work needed to be done.

- Even though we know there probably is hint in the top three chosen words, it still does not come always on the first position and we won't be able produce meaningful hint composed from one word only, as it is required by the game. I would love to make
- We should definitely make some adjustments to the choosing function, which determines which words are going to be chosen for making hints. Such as that in the case of two words, no threshold is needed and this function always takes the two most similar words.
- We focused in this algorithm for connecting the words only. In real game, we need to pay attention for cards of the opponent team (and the killer) as well, as we want to make hints which connect only cards of our team.
- We could make interactive interface as well and try how does this algorithm performs with real people.
- If the algorithm would have interactive interface, it would be possible to create self-learning mechanism, probably using neural networks. Therefore, if made as an web app or mobile app and shared across social networks(or sent to mechanical turk), it would be possible to have very accurate mechanism quite soon.

References

[1] David Silver and Demis Hassabis. Alphago: Mastering the ancient game of go with machine learning.

[2] Ping Xiao and Hannu Toivonen. Layerfolding: Discovering creative links in word association networks, 2016.

[3] Paul Whitney Pak Chung Wong and Jim Thomas. Visualizing association rules for text mining, 1999.

[4] Kai Chen Greg Corrado Jeffrey Dean Tomas Mikolov, Ilya Sutskever. Distributed representations of words and phrases and their compositionality, 2013.

[5] Greg Corrado Tomas Mikolov, Kai Chen and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[6] Kaggle. May 2015 Reddit Comments. Retrieved from https://www.kaggle.com/reddit/reddit-comments-may-2015.

[7] Wikipedia. Wikipedia:good articles/all — Wikipedia, the free encyclopedia.

[8] Vector representations of words retrieved from https://www.tensorflow.org/tutorials/word2vec/.

[9] Christian S. Perone. Machine learning : Cosine similarity for vector space models (part iii).

6 Appendix

6.1 Training of the word2vec model on different datasets

6.1.1 Wikipedia - good articles

	1	2	3
paris-france+germany	berlin	vienna	munich
man-woman+queen	king	ii	prince
athens-greece+norway	norwegian	airport	oslo
switzerland-swiss+dutch	netherlands	belgium	denmark
lucky-luckiest+easiest	pleased	quite	busy
merkel-germany+japan	trax	typhoon	jein
gambling-cards+alcohol	drinking	addiction	drug
tail-human+animal	feathers	tips	ears
beer-prague+kiev	ale	drink	coffee

Table 1: Results from the test

6.1.2 Reddit comments

	1	2	3
paris-france+germany	berlin	amsterdam	london
man-woman+queen	king	holy	mate
athens-greece+norway	iceland	oklahoma	sweden
switzerland-swiss+dutch	netherlands	belgium	holland
lucky-luckiest+easiest	easy	difficult	hard
merkel-germany+japan	lakrits	cand	classifies
gambling-cards+alcohol	smoking	drinking	drugs
tail-human+animal	pony	claws	carve
beer-prague+kiev	beers	drink	ice

Table 2: results from the test

6.2 Results from Codenames AI

6.2.1 Wikipedia

	chosen words	first algorithm	second algorithm
1	ship line	ships near carried	ships **fleet** carried
2	opera dance	casino inspired **music**	**theatre** *ballet* **musical**
3	rock wave park	which another match	*1970s* popular 1960s
4	card deck	another which gun	joker's **tarot** check
5	heart hospital	which where time	**blood** brain **suffering**
6	march shot	after before when	*february december april*
7	heart fall arm	**body** inside *hole*	my your like
8	mail charge	which another also	**delivered** *service fraud*
9	yard lap	*touchdown* pass **ran**	*touchdown* yards pass
10	war washington fall	during beginning following	**military** civil 1945

Table 3: Results from the choosing and hinting function

¿

6.2.2 Reddit comments

	chosen words	first algorithm	second algorithm
1	ship line	half going back	*space* base move
2	opera dance	ghost dice casino	jazz mixes turtle
3	rock wave park	back around one	*band* out up
4	card deck	cards one unless	cards gift check
5	heart hospital	time after where	cause **pain** death
6	march shot	once night went	until before 11
7	heart fall arm	*hole head hand*	cause *mind* eyes
8	mail charge	up king from	send via **receive**
9	yard lap	off shoe store	yards td *touchdown*
10	war washington fall	**us** where we	**us** part came

Table 4: Results from the choosing and hinting function