Maastricht University

Master Research project
Final Report

Move-Based Game Distance

Authors
Adrian Sondermann
Dimitrios Gagatsis
Jivago de França Aires Galvão
Pascal Oliva Otero
Zsolt Harsányi

Supervisors
Cameron Browne
Matthew Stephenson

June 23, 2021

Abstract

This project involving measuring the similarity between various games according to a common property all games have, the moves during play. The main objective of this project is to determine whether a distance metric based purely on the types of moves made during two games is possible and to implement said metric. During several experiments, it was discovered that in order to make the metric reliable, it is not enough to compare just the types of the moves. This is due to the fact that there are several games that share the same move types but are still far from being the same. Therefore improvements were made to the implementation, such as incorporating elements from game descriptions in the form of game concepts which resulted in a more robust metric.

Contents

1 Introduction

Over the centuries, people from different parts of the world shared the habit of playing board games. The first known board game, Senet, dates back to 3,500 BC and was created in Egypt. These games are still popular in contemporary times among individuals, providing not only entertainment but also influencing the way people in different societies think. For instance, chess is used in many analogies to describe strategic actions in politics and in the corporate environment. There are a plethora of types of board games and they are played in many different ways and rules. Some popular games, such as Draughts, have variations across countries, with slightly different rules. According to previous research, board games are responsible for the similarities between cultures around the world. They affect people's behaviors, ethics, and traditions, which highlights the importance of measuring the similarity between board games. To measure the similarity between games is also important for the classification of games, detection of plagiarism and novelty of rule sets.

1.1 Motivation

Given the aforementioned reasons, it is useful to discover a way to measure game similarity. However, it is difficult to invent a reliable similarity metric for the general category of board games. Games come in many different types, categories and are played in many different ways, thus the possibility that any two given games can be represented in a format that allows direct comparison of their internal states is rare. Nevertheless, the most essential property that all games share is the moves during playing them. Hence, by using this property we are able to implement a distance metric to measure the distance between two games, i.e. how similar or dissimilar they are. As a consequence, the problem which is arising from the above and the main perspective of our project is the definition of a reliable distance metric that measures the distance between two games and hence, their similarity.

1.2 Distance Metrics

Distance metrics are a fundamental part of several machine learning algorithms and they are used in both supervised and unsupervised learning, generally to calculate data points. A reliable distance metric improves the performance of a machine learning model whether that's for classification tasks or clustering. Generally, it is challenging to devise a reliable distance metric that operates for a general case of games, thus we intend to implement such a distance metric that is supported with board games and more specifically with Ludii General game system[4]. In our approach, to measure how similar or dissimilar several games are, we are going to use distance metrics and more precisely moved-based distance metrics. In particular, moved-based distance metrics are defined as the distance metrics that measures the similarity of games according to their moves.

What does it mean for two games to be similar? Every game has its own board, rules, moves, and sequences of moves. Since it is difficult to answer the previous question - and there are contradictions between the possible answers - we are interested in finding out if it is possible to determine the similarity of the two games analogously their moves distance. Therefore, the implementation of our distance metric is about finding how close or not are the moves or sequences of moves

of two games.

To define such a distance metric mentioned before, we are using the Ludii General Game System (https://ludii.games/). "Ludii" is designed to play, evaluate, design, compare and optimize a variety of games, including board games (Figure 2). In this system, the games are described by ludemes, which are units of game-related information, including moves and move concepts.

We define the distance of our distance metric, consequently, and the similarity as follows:

$$Distance = 1 - Similarity \tag{1}$$

Specifically, the implemented distance metric will be used to estimate the similarity between games according to their moves (or sequences of moves). The values of both similarity and distance are between 0 and 1. If two games are identical, the distance between them will be close to 0 and the similarity close to 1. Analogously, the distance of two completely different games will converge to 1, and the similarity to 0.

1.3 Social Impact

Researchers over the years tried to discover the origins of games according to literary sources and archaeological findings. It is very often difficult to find these type of resources and time-consuming process, thus by using distance metrics, such as moved-based distance metrics, it is possible to construct family trees of many games according to their rules or moves similarities. By exploring these trees, we can detect similarities amongst games and conclude that there was a cultural exchange or similar social background [3]. In any case, it helps us to detect much faster and with reliability the origin of games and understand properly the human history. Additionally, another significant impact of the development of reliable distance metric for measuring the distance, and thus the similarity of two games is the recommendations of the computer-generated games. A user playing several games could receive recommendations based on their preferable games by comparing those games moves and sequences of moves. This could be an inspiration for the game development industry by creating games with similar moves to those of the most popular games. There are many other parameters to be considered for the game development, however, the moves during the game would be such an interesting parameter mainly for the board game development. For example, there are many genetically evolved ludeme based games research[1] that became popular and successful according to the "Evolutionary Game Design" research[2].

1.4 Research Questions

Our project aims to answer some research questions about the similarity of the games. The main goal of the project is to implement a move-based distance metric for a variety of board games to answer the following questions:

- What does it mean for a board game to be similar or different?
- Is it possible to provide sufficient insights to measure the underlying difference between two games, based on the observed move concepts during the trials, combined with the previously named game concepts?
- Are the designed move-based distance metrics faster and more reliable in contrast with the already existed distance metrics?

2 Concepts & Approaches

2.1 Previous Approaches

The measurement of the distance between games has been explored by previous researches, which mainly adopted approaches involving concepts borrowed from Biology, namely the genotype and the phenotype approaches.

In our context, the genotype approach focuses on the ludemes, that is, on the units of game-related information. Specifically, it refers to the characteristics and encoding of a given game, and the underlying graph representation of the game's board (Browne, 2008)[5]. An application of the genotype approach is given by building labelled trees of the game's rules and comparing the rules using the Levenshtein distance.

The phenotype approach focuses mainly on the way the game is played, according to its rules to detect the game categories. To classify the different games, this type of approach uses several quality measures, such as the average game length and the player's decisions during the games.

One research conducted by a group last year [7] proposed an experimental setup in which they applied a variety set of distance and similarity metrics to games present in the Digital Ludeme Project database. Furthermore, different metrics applied at the descriptive level and play-out level of a game separately. In particular, they show that the genotype metrics proposed are able to identify communities of closely related games. Also, they used the Bag of Words approach to producing a phylogenetic tree that resembles the cultural evolutionary model of some games (e.g. Mancala games). They proved that the phenotype measures demonstrate to be reflective of the game's features provided to them and produced very diverging trees and measures depending on the statistics used to aggregate features.

In terms of their approach, they focused on the genotype and phenotype distance. Firstly, for the genotype distance, they used three different approaches and variations of those. They are the following:

1. Ludeme Edit Distance
2. Board Graph Similarity
3. Bag of Words approach

The Ludeme Edit Distance adapts the Levenshtein string distance to Ludii, that is, it measures the number of ludemes that have to be inserted, deleted and/or edited so that one given game can be transformed into another. This approach was then adapted to a Tree Edit Distance, given the tree-like structures of the ludemes.

The Board Graph Similarity attempts to determine a distance metric for comparing two graphs, aiming at measuring the similarity between two given game boards. It considers undirected graph $G = (V, E)$, in which E is the set of edges, while V is the set of vertices. Then, it considers operations of the graph edit distance (GED), which are:

- Inserting
- Deleting
- Substitute

However, since the problem is NP-complete, the edit distance algorithm used has an exponential computational complexity when it comes to the graph's number of vertices. Therefore, a greedy algorithm and some heuristics were additionally used in the problem.

The other genotype approach involved the use of Basic Frequencies Bag of Words and TFIDF Bag of Words. The former approach takes a game's ludemic description to be equivalent to a text document and examines the probability or frequencies distributions of the words in different games, not considering the order or relationship between words. In the other case, it implements the term frequency-inverse document frequency (TFIDF) methodology, in which more importance is given to words that are not greatly shared between games.

Secondly, two approaches were explored based on the phenotype distance. Both of them use the Agents of Ludii to play the games several times and extract information from the course of these games to calculate similarities.

1. Measuring correlations among time series of different games

2. Manhattan distance of different Features

Finally, they tried two broadly different approaches focusing on the rules and equipment, and the playout characteristics of the games respectively. It seems that Phenotype distances between games in particular described a very different topology of resemblance over the games than the one that would be expected to arise from resemblance in terms of historical evolution, as could be seen by focusing on the well studied subset of the mancala family of games. Genotype distances were more effective at capturing historical similarity of games.

An example of phenotype approach used by a previous research group (Kirill et al., 2021)[8] is to use the so-called Play Trace Distance Metric, which captures how a game is played by using the play traces. A play trace of a game is a list of moves done during a play out. The research also used Tree Edit Distance Metric, comparing games through the calculation of the distance between the game trees generated by two given games. In both approaches, labelling algorithms were used, enabling game comparisons by storing information inside the phonetic structures that are generated. The research showed that using distance metrics that already exists on generated phonetic structures of two given games is a viable way for calculating distance metric.

2.2 Approach

In our approach, the main goal was to try to move away from the usual way of comparing games. The previous methods mainly used the game descriptions and rules themselves as a base for comparison as mentioned before, but they did not take into account the other common thing that is shared between all games, that is, the moves performed while playing the games. Making moves is an integral part of all board games and it might be a good way to find differences and similarities between them. Another benefit of this approach is that it is a fairly simple way to compare the games, requiring no intricate algorithms. In this research, we mainly focused on methods using the Move Concepts, that is, on a phenotype framework. However, there was also a combination of N-Grams and Game Concepts in one of the approaches used to measure the distance between games. In this case, both genotype and phenotype frameworks were used, through N-Grams and Game Concepts, respectively.

In Ludii, all the moves are recorded as they are played and are given a number that indicates the type of the move. These are called move concepts and this is what our approach is based on. We have implemented multiple, iteratively

improved versions of the metric and these are explained in detail in section 3.

2.3 Concepts

2.3.1 Ludii

The Ludii general game system, part of the Digital Ludeme Project (DLP), is a software that enables playing, comparing, evaluating, designing and optimising a variety of games. Under the DLP project, which is funded by the European Research Council (ERC), the software is intended to be used to model different traditional strategy games, aiming to model a representative sample of 1,000 strategy games in the Ludii Game Database. Ludii provides a platform encompassing a range of game design services, including automated game design, game optimisation and historical game reconstruction. The latter, in particular, aims at performing reconstruction of historical games using partial or unreliable information, which includes material evidence such as partial game boards and pieces [4].

The idea behind Ludii is the so-called ludemic approach, which takes into account the atomic constituents of games, describing key concepts related to games. Ludemes are units of game-related information, which have a tree structure representation of the game-related concepts. Below is an example of the game Tic-Tac-Toe represented in a ludemic form.

```
(game "Tic-Tac-Toe"
  (players 2)
  (equipment {
    (board (square 3))
    (piece "Nought" P1)
    (piece "Cross" P2)
  })
  (rules
    (play (to (empty)))
    (end (if (line 3) (result Mover Win)))
  )
)
```

2.3.2 Games in Ludii

In Ludii, a game can be represented by a $4-tuple =< Players, Mode, Equipment, Rules >$.

1. **Players:** Set of k players.
2. **Mode:** Type of the game (alternating, simultaneous and real time).
3. **Equipment:**
   - Containers: Description of the main board.
   - Components: Each described by a ludeme piece, with its name, owner and how it can move in the board.
4. **Rules:** Operations of the game, split in start, play and end.

2.4 Game Concepts

Ludii contains 523[1] different game concepts. These concepts are terms that describe an idea related to a given game and encompass something useful about it. They can be accessed as a Java BitSet using the Game-objects and are of heterogeneous value types, i.e. boolean, integer, etc. A game concept is a type of static concept that is present in Ludii. The concepts can be described by the following taxonomy [6]:

[1] As of June 23, 2021

1. **Properties:** General properties of the game.
2. **Equipment:** Equipment for playing the game.
3. **Rules:** Rules of the game.
4. **Metrics:** Metrics.
5. **Math:** Mathematics.
6. **Visual:** Important visual aspects.
7. **Implementation:** Internal implementation details, e.g. for performance predictions.

2.4.1 Playing in Ludii

Within Ludii, the process of playing a game can be described in a hierarchical fashion:

- **Action:** Atomic actions; when applied, modify a single property of the game state.
- **Move:** Sequence of actions.
- **Turn:** Sequence of moves by the same player.
- **Trial:** Sequence of turns.

Therefore, one Trial is an instance of exactly one played game, consisting of several turns.

2.4.2 Move Concepts

Every move has multiple binary move concepts, which can be queried throughout a function Ludii provides. These move concepts differentiate between the different types of moves that can be made in a game. For example, placing a new piece on the board is a different kind of move than pushing a piece to another spot on the board. These distinctions help us compare the games.

3 Methodology

To assess the move-based distance between games, we have to define a distance measurement returning a distance $d \in [0, 1]$ which estimates the dissimilarity of two games. Hence $d = 0$ indicates that both games are identical, while $d = 1$ means they are totally different. Note that the meaning of these estimates d will be explained later on in this report.

Our main interest is to implement a moved-based distance metric, which will focus on the moves made while playing the games repeatedly. Then the algorithm can evaluate the resulting move concepts in the trials. During the project, we noticed that some scenarios exist, where estimating a distance based only on move concepts can lead to extremely small distances d. Therefore, we decided to use the general game concepts for our more advanced approaches too.

One important aspect to keep in mind while designing the algorithm is that speed will be critical, because ultimately it will be employed to compare many games at once. As many random trials and thus a long runtime can be expected, we conclusively divided the data generation task from the actual distance metrics. Both parts of the methodology will be discussed now.

3.1 Data Generation

The data generation task is concerned with simulating a big, given number n of trials of a list of games within Ludii. For the simulation the user needs to be able to specify an AI, which

will play the games automatically. Throughout each trial, the corresponding move concepts can be requested from Ludii as a java BitSet. Finally, these sequences of BitSets will be stored in *Distance/log/{game}/{ai}/* as comma-separated values (CSV) files. Here {game} and {ai} are placeholders for the name of the game as well as the employed AI respectively.

Optionally, it is possible to store the trials themselves to file, such that they can be replayed using Ludii's graphical user interface. This is solely meant for testing purposes. Hence, this is optional.

In algorithm 1 the general procedure of the implementation is represented. The pseudocode is a bit simplified, because statements used for debugging and saving optional data other than the move concepts are omitted.

Algorithm 1 Trial Simulation (SimulatorApp.java)

```
1:  function TRIALRUN(game, ai, numTrials)
2:      Initialize trial for game
3:      Initialize context for game and trial
4:      for i=0 to numTrials-1 do
5:          Start game with context
6:          for p=1 to #players in game do
7:              Initialize AI for game
8:          while trial not over do
9:              execute a single move
10:         Initialize moveConceptsList
11:         for move in trial.moves do
12:             Add move to moveConceptsList
13:         Close all AI's for game
14:         Initialize (CSV) file according to section 3.1
15:         Initialize writer for file
16:         for set in moveConceptsList do
17:             Write set as String
18:         Close writer
```

3.2 Metrics

The second part of the methodology is concerned with constructing and implementing distance metrics, which have to return the distance estimate d as defined in the beginning. These have to be computed by assessing the move concepts throughout the simulated trials. The ideas behind them as well as details about the code will be clarified in the following subsections. All implemented approaches implement the interface `DistanceMetric`, which is already defined in Ludii.

3.2.1 Occurrence of Move Concepts

Our initial idea is solely based on the occurrences of individual move concepts. Hence, we create a BitSet indicating whether a move concept arises for each game to evaluate. Afterwards, computing the degree of overlap between BitSets of dif-

ferent games in range 0...1 results in a first sensible estimate.

More formally, let C be the total number of concept enums defined in Concept.java. Moreover, let g be an arbitrary game in Ludii with corresponding trials $t_{g,1}, ..., t_{g,n}$. Then, the BitSet M_g can be determined from the trials:

$$M_g = \{m_1, m_2, ..., m_C\} \; \forall g$$
$$m_i = \begin{cases} 1, & \text{if } \exists j \in 1, ..., n : \#m_{t_{g,j}} \geq 1 \\ 0, & \text{otherwise} \end{cases} \; \forall i \quad (2)$$

In eq. (2) $\#m_{t_{g,j}}$ indicates the number of occurences of move concept m in one of the n games' trials. The next step is to compute d:

$$d = 1 - \frac{|M_g \cap M_h|}{|M_g \cup M_h|} \; \forall \text{games } g, h \quad (3)$$

Algorithm 2 contains pseudocode of the major functions in MoveConceptOverlap.java. In order to reduce the computation time to its minimum, we have included a private attribute `moveConceptsByGame` of type HashMap, which allows to store the data per game. Thus, the CSV files only need to be read once, when comparing a game the first time. For every other comparison, the BitSet will be re-used. Hence, the time needed is minimized.

3.2.2 Frequency of Move Concepts

The second approach aimed to consider the frequencies of individual concepts, rather than evaluating boolean indicator variables. Thus, storing a BitSet, which denotes whether a move concept has occurred is not sufficient anymore. This issue can be resolved by applying HashMaps with integer keys and decimal values. These HashMaps

Algorithm 2 Overlap of occurring Move Concepts (MoveConceptOverlap.java)

function DISTANCE(**gameA**, **gameB**)
  moveConceptsA:= **getMoveConcepts(gameA)**
  moveConceptsB:= **getMoveConcepts(gameB)**
  Return estimate **d** wrapped as Score object by assessing both BitSets

function OVERLAP(**moveConceptsA**, **moveConceptsB**)
  Initialize **intersection** to **moveConceptsA** AND **moveConceptsB**
  Initialize **union** to **moveConceptsA** OR **moveConceptsB**
  Return **d** as defined in eq. (3)

function GETMOVECONCEPTS(**game**)
  if HashMap **this.moveConceptsByGame** contains BitSet for **game** **then**
    return the BitSet
  Initialize **moveConcepts** as empty BitSet
  Initialize **folder** containing all the CSV files of **game**
  for **csv** in **folder.listFiles()** being a file ending with ".csv" **do**
    Initialize **reader**
    line := ""
    while next **line** $\neq$ null **do**
      if **line** not empty **then**
        Split **line** by delimiter
        Parse each substring to int
        Set Bits in **moveConcepts**
    Close **reader**
  Put **moveConcepts** in **this.moveConceptsByGame**
  Return **moveConcepts**

can be seen as a sparse representation of high dimensional frequency vectors.

Due to the sparseness and high dimensional vector space of the vectors, we planned to utilize the cosine similarity measure. This firstly results in a low complexity of computation, as only non-zero frequencies need to be considered. Secondly, frequencies being zero for a single game will be prevented from dominating the distance estimates, which would be the case with many other measures, i.e. euclidean distance. Another advantage of this measure in combination with non-negative frequency vectors is that its outcome is neatly bounded in $[0, 1]$. Therefore, we conclusively decided to apply the cosine similarity measure.

To describe the approach more precisely, let C be the number of concepts defined and g be an arbitrary game in Ludii with corresponding trials $t_{g,1}, ..., t_{g,n}$. For each trial let $M_{g,j,l} = \{m_1, ..., m_C\}$ be the move concept sets for trials $j \in \{1, ..., n\}$ and trial length l. Then the sparse frequencies F_g over all trials can be calculated as in eq. (4).

$$\begin{aligned} F_g &= \{f_1, f_2, ..., f_C\} \ \forall g \\ f_i &= \frac{\sum_{j=1}^{n} \sum_{M \in t_{g,j} : m_i = 1} 1}{\sum_{j=1}^{n} \sum_{M \in t_{g,j}} 1} \ \forall i \end{aligned} \tag{4}$$

Eq. (5) expresses the formula for the estimate d to return based on the cosine similarity.

$$\begin{aligned} d &= 1 - cos(\theta) \\ &= 1 - \frac{F_g \cdot F_h}{||F_g||\ ||F_h||} \\ &= 1 - \frac{\sum_{i=1}^{C} F_{g,i} F_{h,i}}{\sqrt{\sum_{i=1}^{C} F_{g,i}^2} \sqrt{\sum_{i=1}^{C} F_{h,i}^2}} \ \forall \text{games } g, h \end{aligned} \tag{5}$$

Algorithm 3 contains the pseudocode of the implementation. In order to reduce the computation time the same trick as in algorithm 2 is reused. Hence, it is not displayed again.

3.3 Sequences of Move Concepts

Next to the frequencies of move concepts another important property of the trials, which has not been yet evaluated, is the order of moves. Hence, we can assess sequences of concepts by creating n-grams per individual trial.

An easy initial way to store these would be a two-dimensional list of BitSets, i.e. `ArrayList<ArrayList<BitSet>>` in java. On the one hand, this approach is not particularly space-efficient, because each BitSets is stored up to n times and we need to store the n-grams themselves. But on the other hand, it is possible to fine-tune the data structure to the designed algorithm.

Based on the n-grams, different methods for comparing them were developed in past years. One promising approach was proposed by Kondrak [9]. It defines n-gram similarity/distance recursively equivalent to the recursive definition of the standard levenshtein edit distance. The issue remaining with an approach designed to compare two strings is that our n-grams consist of BitSets and there are multiple trials per game. Comparing all the trials would increase computation time exponentially.

Thus, we started with a metric that utilizes frequencies of n-grams. This allows us to capture both, frequencies and sequences, without suffering the loss of speed. Moreover, it allows minimizing the storage complexity too. It is possible to store a value pair of an integer and decimal number. The integer is the hashed value of the BitSet n-gram, which serves as key for the dec-

Algorithm 3 Frequencies of Move Concepts (MoveConceptFrequenciesCosineSimilarity.java)

function DISTANCE(**gameA**, **gameB**)
  frequenciesA:= **getMoveConceptFrequencies(gameA)**
  frequenciesB:= **getMoveConceptFrequencies(gameB)**
  Return estimate **d** wrapped as Score object by calling 1.0 - **cosineSimilarity**

function COSINESIMILARITY(**fA**, **fB**)
  nominator := **dotProduct(fA, fB)**
  Initialize **denominator** as product of both norms
  Return **nominator** / **denominator**

function DOTPRODUCT(**fA**, **fB**)
  ret := 0.0
  for key, **value** in entry set of **fA do**
    d:= **value** multiplied by **fb.getOrDefault(key, 0.0)**
    ret = **ret** + **d**
  Return **ret**

function NORM(**f**)
  ret := 0.0;
  for d in values of **f do**
    ret = **ret** + d*d
  Return square root of **ret**

function GETMOVECONCEPTFREQUENCIES(**game**)
  ▷ As defined in algorithm 2, but returning a **HashMap<Integer, Double>** of frequencies

imal frequency. Therefore a HashMap will be employed for storage purposes.

To estimate the final distance between two (very) sparse frequency vectors, we once again employed the cosine similarity measure.

More formally, let g be an arbitrary game with trials $t_{g,1}, ..., t_{g,n}$. Furthermore, let $G_{g,j} = \{G_{g,j,1}, ..., G_{g,j,l-N+1}\}$ be the n-grams for trials $j \in 1, ..., n$, trial length l, and n-gram length N. Then, the frequencies F_g for all C^N possible n-grams for game g can be expressed as follows:

$$F_g = \{f_1, f_2, ..., f_{C^N}\} \ \forall g$$
$$f_i = \frac{\sum_{j=1}^{n} \sum_{G \in G_{g,j}: hash(G)=i} 1}{\sum_{j=1}^{n} \sum_{G \in G_{g,j}} 1} \ \forall i \qquad (6)$$

To estimate d using the cosine similarity measure eq. (5) can be reused. Details about the implementation can be found in algorithm 4 and algorithm 5. Note that in `getNGrams(game)` lines will be skipped, if they are the n-th consecutive empty line. This is done because most games consist of a substantial part of moves having no concepts assigned. This can lower the distance score artificially, as these "empty" moves are shared between games, without any notable corresponding action performed by the players. Partially empty n-grams are not discarded, as they often indicate the end of a trial and thus hold valuable information.

3.4 Game Concepts

Throughout the project it became visible that some games cannot be separated solely by sequences of move concepts, i.e. Go and Hex, even though for most games sound, up to near perfect (Knight Moves games). These results will be presented and discussed later on. Next to moves,

Algorithm 4 Frequencies of N-Grams 1 (MoveConceptNGramCosineSimilarity.java)

function DISTANCE(**gameA**, **gameB**)
  frequenciesA:= **getNGrams(gameA)**
  frequenciesB:= **getNGrams(gameB)**
  Return estimate **d** wrapped as Score object by calling 1.0 - **cosineSimilarity**

function COSINESIMILARITY(**fA**, **fB**)
  nominator := **dotProduct(fA, fB)**
  Initialize **denominator** as product of both norms
  Return **nominator** / **denominator**

function DOTPRODUCT(**fA**, **fB**)
  ret := 0.0
  for key, **value** in entry set of **fA do**
    d:= **value** multiplied by **fb.getOrDefault(key, 0.0)**
    ret = **ret** + **d**
  Return **ret**

function NORM(**f**)
  ret := 0.0;
  for d in values of **f do**
    ret = **ret** + d*d
  Return square root of **ret**

Algorithm 5 Frequencies of N-Grams 2 (MoveConceptFrequenciesCosineSimilarity.java)

function GETNGRAMS(**game**)
  if HashMap **this.nGramFrequencies** contains frequencies for **game then**
    return the frequencies
  Initialize **frequencies** as HashMap
  Initialize **counter** as HashMap
  count := 0
  Initialize **folder** containing all the CSV files of **game**
  for csv in **folder.listFiles()** being a file ending with ".csv" **do**
    Initialize **nGram** as LinkedList
    i := 0
    Initialize **reader**
    line := ""
    while next **line** ≠ null **do**
      if line is **this.n**-th consecutive empty line **then**
        Continue
      If **i**≥**this.n**: **nGram.remove(0)**
      Add BitSet from **line** to **nGram**
      i++
      if i ≥ **this.n then**
        hash := **nGram.hashCode()**
        Increase **counter** at key **hash**
        count++
    Close **reader**
  for key, **value** in entry set of **counter do**
    Put (**key**, **value**/**count**) as entry in **frequencies**
  Put **frequencies** in **this.nGramFrequencies**
  Return **frequencies**

there is another property, which all games share, at least in the Ludii project. Every game object has boolean concepts, which can be calculated and retrieved in BitSet form. Thus, comparing the game concept sets is achievable as described in section 3.2.1 and the pseudocode can be found in algorithm 6.

This metric has no problems with computing thousands of estimates at once, as it does not read from CSV files. Instead, java's BitSets are evaluated, which are pre-computed in the game object. Moreover, BitSets are designed to perform very fast when applying logical operators such as the logical "and" as well as logical "or". Both can be found in the function `overlap`.

Algorithm 6 Game Concepts (GameConceptsOverlap.java)

function DISTANCE(**gameA**, **gameB**)
Initialize **booleanConceptsA**:= boolean concepts of **gameA**
Initialize **booleanConceptsB**:= boolean concepts of **gameB**
Return estimate **d** wrapped as Score object by assessing both BitSets

function OVERLAP(**conceptsA**, **conceptsB**)
Initialize **intersection** to **conceptsA** AND **conceptsB**
Initialize **union** to **conceptsA** OR **conceptsB**
Return 1.0 - (|**intersection**|/|**union**|)

3.5 Combining it all

To conclude our approaches, we designed another metric combining sequences and frequencies of move concepts with game concepts. The metric has two distance metric attributes, one as lined out in section 3.3, the other metric is introduced in section 3.4. Furthermore both can be weighted by a decimal weight attribute $w_i \in [0, 1],\ i = 1, 2$ with constraint $w_1 + w_2 = 1$. This yields a fine-tunable estimator.

Algorithm 7 Combined Metric (CombinedConcepts.java)

function DISTANCE(**gameA**, **gameB**)
Initialize **score1** by calling **distance** of first metric
Initialize **score2** by calling **distance** of second metric
Return **w1***score1 + **w2***score2 wrapped as Score object

4 Experiments & Results

4.1 Experiments

The experiments were conducted with three different groups of games, Benchmark games, Knight Moves games and Draught games, each to assess different aspects of the distances between those games.

In the experimentation, a total of 1,000 trials were conducted using random AI for each game and the results were computed. To check whether the given amount of simulations was enough to give meaningful results, the same process was conducted again and then the results were compared to the previous ones, both sets of results were added to a symmetric matrix, for the purposes of comparison. The difference in the results computed on both procedures were zero, meaning the 1,000 trials were sufficient to compute the results. Then, the distances between the games were measured using different

approaches described in the methodology.

The group of benchmark games was created to encompass games from different categories, played in diverse geographical locations. The trials conducted on this group involved 18 games, including the widely known Tic-Tac-Toe and Chess. As expected, given the different types of games included in this group, the results showed there were instances of games far apart from each other, as well as those with medium distances and also games that were very similar, i.e., small distance between them.

To assess the distance between games of the same category, a group of Draught games was created, which included 26 variations of Draught games.

The third group of games tested was the Knight Moves games, which are the same games but computed using different ludemes. This game is basically a chess game in which all the pieces are knights. The trials on this group involved 20 Knight Moves games.

All the tables shown in the results section are reduced tables, meaning that they do not include the whole games of the category involved, only the more relevant results. The full tables are in the appendix section.

4.2 First Approach: Computing the overlap of the bit sets

The first approach of our project for the comparison of the game distance, therefore for the similarity, was to compare the overlap of the move concepts between the games as mentioned in section 3.

The following figure illustrates the move-distance between Draughts Games by using the overlap of their move concepts.

Draughts Game	Brazilian Draughts	Cage	Dama	Damas	English Draughts	Game of Solomon	HexDame	International Draughts
Brazilian Draughts		0,5	0	0,375	0	0,4	0	0
Cage	0,5		0,5	0,3333	0,5	0,75	0,5	0,5
Dama	0	0,5		0,375	0	0,4	0	0
Damas	0,375	0,3333	0,375		0,375	0,625	0,375	0,375
English Draughts	0	0,5	0	0,375		0,4	0	0
Game of Solomon	0,4	0,75	0,4	0,625	0,4		0,4	0,4
HexDame	0	0,5	0	0,375	0	0,4		0
International Draughts	0	0,5	0	0,375	0	0,4	0	

Figure 1: Draughts Games Move Concept Overlap

4.3 Second Approach: frequency and cosine similarity of the move concepts

This approach was tested with the three different categories of games. First, with the 18 Benchmark Games provided by Ludii. Subsequently was tested on the whole Draughts category, which includes 26 games, with similar concepts as the are all leaping games. And finally with the Knight Moves games, which are 20 tests games with the same moves, basically a Chess game with only knights as pieces, but with different ludemes.

	Chess	English Draughts	Go	Hex	Yavalath	Tic-Tac-Toe
Chess		0.5312	0.9977	1.000	1.000	1.000
English Draughts	0.5312		0.9914	1.000	0.9999	1.000
Go	0.9977	0.9914		0.0054	0.0054	0.0046
Hex	1.0000	1.0000	0.0054		0.0000	0.0011
Yavalath	1.0000	0.9999	0.0054	0.0000		0.0011
Tic-Tac-Toe	1.0000	1.0000	0.0046	0.0011	0.0011	

Figure 2: Benchmark Games Move Frequency distance

In the table in the figure 2, which contains some of the more relevant results of the Benchmark Games analysis, it is possible to see that there are very different relations between games, with games that are very far apart, games in medium-range, and very close games. A good example of games being very distant are Chess and Tic-Tac-Toe, since this metric is only based in moves, these two games are completely far apart since they do not share one single type of move, in Tic-Tac-Toe it is only possible to add pieces to the board, in Chess pieces are already in the board and they change positions.

In games of medium-range, there are examples such as Chess and English Draughts which have some similarities but of course, have many differences. These games are similar in some types of moves, such as captures and the fact that the pieces are on the board at the beginning of the game and the pieces are moved from one point to another, but they are also different in terms of the leaping movements of the Draughts and the many different types of moves that chess has.

Finally, it is possible to see results of games that with this specific metric of only moves, are very similar, an example of this are the games Hex and Go, the goals of these games are very different, and the way of attacking the game is also distant, but in terms of only and only the moves, both are an empty board where on each turn a player adds one piece (and in the case of Go occasionally capture an enemy piece which explains why the distance it is not exactly $d = 0$) which explains the reason of why are these game so close with this metric.

The table in the figure 3 evaluates with the move frequency approach the Draught Games category.

As we see the results are much closer compared to the Benchmark Games. This of course

Draught Games	Brazilian Draughts	Cage	Damas	English Draughts	Game of Solomon	International Draughts
Brazilian Draughts		0.2172	0.2209	0.0039	0.0119	0.0014
Cage	0.2172		0.4001	0.2281	0.2418	0.2131
Damas	0.2209	0.4001		0.2148	0.2142	0.2275
English Draughts	0.0039	0.2281	0.2148		0.0050	0.0090
Game of Solomon	0.0119	0.2418	0.2142	0.0050		0.0213
International Draughts	0.0014	0.2131	0.2275	0.0090	0.0213	

Figure 3: Draughts Games Move Frequency Distance

makes sense since every game is inside the same category. As mentioned before the largest distance was at most $d = 0.4001$. And the average distance of the whole matrix of move frequency which included the 28 Draughts was $d = 0.0581$

At last, the approach was tested on the previously mentioned 20 Knight Moves Games.

	Knight Moves 1	Knight Moves 2	Knight Moves 3	Knight Moves 4	...	Knight Moves 20
Knight Moves 1		0.0000	0.0044	0.0044	...	0.0044
Knight Moves 2	0.0000		0.0044	0.0044	...	0.0044
Knight Moves 3	0.0044	0.0044		0.0000	...	0.0000
Knight Moves 4	0.0044	0.0044	0.0000		...	0.0000
...	...	...	...	...		...
Knight Moves 20	0.0044	0.0044	0.0000	0.0000	...	

Figure 4: Knight Moves Games Move Frequency Distance

For these games, the ideal result of a distance metric would be exactly $d = 0$ in all the scenarios, since they are all the same game but developed with different ludemes. The table in the figure 4 is just a section part of a full 20 by 20 matrix in which we can see that every result is very close to $d = 0$ (the larges result was smaller than $d = 0.01$). As it was men-

tioned, these games have different ludemes and they were simulated with artificial intelligence that follows these ludemes, thus in some games the frequency of some moves, for example, the captures, are slightly different to others.

4.4 Third Approach: N-Grams Cosine Similarity

As it was mentioned before the N-Grams are used to capture sequences of moves in the games. The algorithm was developed with the possibility of utilizing 2-Grams, 3-Grams and 5-Grams. In the following table in the figure 5 it is shown comparisons between different Knight Moves games each one with 2, 3 and 5 grams to identify the differences between them.

	Knight Moves 4			Knight Moves 16			Knight Moves 19		
	2-gram	3-gram	5-gram	2-gram	3-gram	5-gram	2-gram	3-gram	5-gram
Knight Moves 1	0.007	0.009	0.016	0.007	0.009	0.016	0.000	0.000	0.000
Knight Moves 7	0.007	0.008	0.016	0.007	0.008	0.016	0.000	0.000	0.000
Knight Moves 9	0.011	0.015	0.029	0.011	0.015	0.029	0.001	0.001	0.002
Knight Moves 18	0.007	0.009	0.016	0.007	0.009	0.016	0.000	0.000	0.000

Figure 5: N-Grams with Knight Moves Games

In this table it is evidenced how even in very similar games such as Knight Moves, there is a direct relationship between the number of grams and the distance among them. Since with more grams the possible sequences have more variations resulting in larger distances. In the cases where the distance is $d = 0$, it stays identical independently of the grams used in the algorithm.

To also evaluate this approach, it was tested on the Draughts games with the 2, 3 and 5 - grams. In figure 6 there are some of the results comparing 6 games of this category.

We can see that, similarly to the Knight Moves, there is an upper trend in the distance between the games while the number of grams is bigger.

	Game of Solomon			Damas			International Draughts		
	2-grams	3-grams	5-grams	2-grams	3-grams	5-grams	2-grams	3-grams	5-grams
Brazilian Draughts	0.0114	0.0178	0.035	0.1977	0.2312	0.2236	0.0006	0.0009	0.0014
Cage	0.0845	0.0666	0.0806	0.2507	0.2558	0.2267	0.0761	0.0465	0.0284
English Draughts	0.0043	0.0063	0.0141	0.1928	0.229	0.2244	0.0077	0.0104	0.0139

Figure 6: N-Grams with Draughts Games

4.5 Fourth Approach: Game Concepts

As mentioned, this approach is very similar to the first approach with the difference that is based on the game concepts instead of the move concepts.

Knight Moves	Knight Moves 1	Knight Moves 2	Knight Moves 3	...	Knight Moves 20
Knight Moves 1		0.20	0.23	...	0.19
Knight Moves 2	0.20		0.18	...	0.11
Knight Moves 3	0.23	0.18		...	0.08
...	...	...	...		...
Knight Moves 20	0.19	0.11	0.08	....	

Figure 7: Game Concept Overlap in Knight Moves Games

The table in Figure 7 shows the overlap of game concepts between Knight Moves which as also mentioned before in terms of moves is the same game, but in terms of the ludeme and the setting of the game can differ up to $d = 0.3$.

4.6 Fifth Approach: Combining N-Grams and Game Concept

Finally, the last approach is a combination of the sequences of moves captured by the N-Grams algorithm and the game concepts captured by the GameConcept algorithm. It is important to find

the correct distribution of importance (weight) for each part of the algorithm. The following tables will show some results of Benchmark Games comparisons, these results are the product of working with the 3-Grams algorithm. Each table has its own balance of weights.

Benchmark Games	Chess	English Draughts	Go	Hex	Yavalath	Tic-Tac-Toe
Chess		0.4531	0.7632	0.8264	0.832	0.8149
English Draughts	0.4531		0.7118	0.7717	0.7793	0.7534
Go	0.7632	0.7118		0.4037	0.3998	0.3916
Hex	0.8264	0.7717	0.4037		0.1043	0.2559
Yavalath	0.832	0.7793	0.3998	0.1043		0.2002
Tic-Tac-Toe	0.8149	0.7534	0.3916	0.2559	0.2002	

Figure 8: Benchmark Moves Games Weight: 0.3 3-Gram and 0.7 Game Concepts

In the first table, the 3-Grams receives 0.3 of importance, while the game concepts 0.7. The average distance in the full 18 Benchmark Games matrix was $d = 0.6043$. The two closest pair in the whole matrix were Taikyoku Shogi with Tai Shogi with a $d = 0.0555$ and Backgamon with Plakoto with a $d = 0.0685$. On the other end, the furthest away pair of games in the whole Benchmark matrix were Taikyoku Shogi with Yavalath with a $d = 0.8433$.

Benchmark Games	Chess	English Draughts	Go	Hex	Yavalath	Tic-Tac-Toe
Chess		0.4863	0.8308	0.876	0.88	0.8678
English Draughts	0.4863		0.7941	0.837	0.8424	0.8239
Go	0.8308	0.7941		0.2919	0.2891	0.2832
Hex	0.876	0.837	0.2919		0.0745	0.183
Yavalath	0.88	0.8424	0.2891	0.0745		0.1431
Tic-Tac-Toe	0.8678	0.8239	0.2832	0.183	0.1431	

Figure 9: Benchmark Moves Games Weight: 0.5 3-Gram and 0.5 Game Concepts

The second table distributes the relevance in the same proportion for both parts with 0.5 and 0.5. The average distance in the full 18 Benchmark Games matrix was $d = 0.6353$. The two closest pair in the whole matrix were Taikyoku Shogi with Tai Shogi with a $d = 0.0515$ and Backgamon with Plakoto with a $d = 0.0685$. On the other end, the furthest away pair of games in the whole Benchmark matrix were Taikyoku Shogi with Yavalath with a $d = 0.8881$.

Benchmark Games	Chess	English Draughts	Go	Hex	Yavalath	Tic-Tac-Toe
Chess		0.5194	0.8985	0.9256	0.928	0.9207
English Draughts	0.5194		0.8765	0.9022	0.9054	0.8943
Go	0.8985	0.8765		0.1801	0.1784	0.1749
Hex	0.9256	0.9022	0.1801		0.0447	0.11
Yavalath	0.928	0.9054	0.1784	0.0447		0.0861
Tic-Tac-Toe	0.9207	0.8943	0.1749	0.11	0.0861	

Figure 10: Benchmark Moves Games Weight: 0.7 3-Gram and 0.3 Game Concepts

Finally, the third table gives the 3-grams 0.7 of importance while the game concepts part of the algorithm receives 0.3. The average distance in the full 18 Benchmark Games matrix was $d = 0.6664$. The two closest pair in the whole matrix were Taikyoku Shogi with Tai Shogi with a $d = 0.0327$ and Backgamon with Plakoto with a $d = 0.0344$. On the other end, the furthest away pair of games in the whole Benchmark matrix were Taikyoku Shogi with Yavalath with a $d = 0.9328$.

The algorithm was also tested with the 20 Knight Moves games, the next table in the figure 11 shows the distance using the 3-Grams algorithm and the distribution of weights as 0.3 for the 3-Grams and 0.7 for the game concepts.

It is possible to see that, since in the Knight Moves games the main difference is in the game concepts, the result of evaluating them with more weight in the game concepts than in the

	Knight Moves 1	Knight Moves 2	Knight Moves 3	...	Knight Moves 20
Knight Moves 1		0.14	0.16	...	0.14
Knight Moves 2	0.14		0.13	...	0.08
Knight Moves 3	0.16	0.13		...	0.06
...	...	...	...		0.03
Knight Moves 20	0.14	0.08	0.06	...	

Figure 11: Knight Moves Games Weight: 0.3 3-Grams and 0.7 Game Concepts

moves, makes them be farther apart.

Finally, in the figure 12, the algorithm was tested for Draughts Games with 3-Grams and the distribution of weights as 0.3 for the 3-Grams and 0.7 for the game concepts.

	Brazilian Draughts	Cage	Damas	English Draughts	Game of Solomon	International Draughts
Brazilian Draughts		0.2689	0.1446	0.0596	0.2287	0.0003
Cage	0.2689		0.3238	0.2378	0.3375	0.2685
Damas	0.1446	0.3238		0.1828	0.3213	0.1453
English Draughts	0.0596	0.2378	0.1828		0.1869	0.0608
Game of Solomon	0.2287	0.3375	0.3213	0.1869		0.2308
International Draughts	0.0003	0.2685	0.1453	0.0608	0.2308	

Figure 12: Draughts Games Weight: 0.3 3-Grams and 0.7 Game Concepts

The results are a bit larger than the ones given by the second approach. The average for the whole matrix including every Draught game was a distance of $d = 0.1486$.

5 Discussion

5.1 Move Concept Approaches

Occurrence of Move Concepts First of all, we did several experiments with our approaches to measure the distance of the moves in order to measure the similarity among games -which is one of the main tasks of this research-. Moreover, after reviewing and evaluating the results of every approach, the main objective was to improve those results into more accurate and reasonable results.

More specifically, our first approach was about computing the overlap of the move concepts between Draughts games. By using the Occurrence of Move Concepts, as described in section 3.2.1, we observe the results of the first approach method about the distance between Draught games in the figure from section 5.1 1. For example, the distance between Cage and Game of Solomon, according to our results is $d = 0.75$, as a consequence, their similarity is 0.25. This happens because the Game of Solomon has fewer move concepts than the cage game and we do not consider any weights about the overlapping concepts comparisons in this method. In general, the first approach seems that is not that reliable because only considers the appearance of move concepts. In certain circumstances, games have much more move concepts than others and this gives dissimilar results from what we expected, considering two games from the same category that could be really close in reality.

Frequency and cosine similarity of the Move Concepts The second approach focus exclusively in the frequencies of the moves. As it is shown in the results section in the figure 4, about the Knight Moves comparisons, this approach works quite good with these kinds of games, since all the results in the matrix are very close to $d = 0$ and as it has been told before these are the same game.

When the approach was tested on the Draughts Games it was also possible to see good improvements comparing to the first approach. With the move frequency distance, the draughts

games are way closer, as expected since they are in the same category. The figure 3 shows that the two furthest away games (which are also the furthers away game in the full matrix) are within a distance of $d = 0.4001$ compared to the $d = 0.7500$ of the first approach. The average of the full matrix was $d = 0.1200$.

Finally, when the approach was tested with the Benchmark games, it is possible to see in the figure 2 the results variate a lot depending on the game, some game completely far apart like Chess and Tic-Tac-Toe with distance $d = 1$, some games with a mid-range distance such as Chess and English Draughts with a distance of $d = 0.4531$, and very close games like Hex and Go with a distance of $d = 0.0054$. This last result was important to realize that this algorithm, although it showed the best results compared to the first approach, it also had problems with some specific games. One of these cases was the Hex and Go comparison, these two being so similar is referred to the fact that the possible moves in these games are pretty much the same, add one piece to the board on every player turn, although the goal, strategies and structure of the game are way different this approach only focuses on the moves concepts.

N-Grams Cosine Similarity The third approach idea is to use N-Grams to capture the sequences of moves in games and use them to compare the distance between the games. The implemented algorithm has the possibility to utilize 2, 3 and 5 Grams to identify the similarity between games. In our experimentation part, we used the Knight games and the Draught Games, as we observe in the figures 5 and 6 respectively.

According to the above results, it becomes clear that there is a direct relationship between the number of Grams and the distance, also in very similar games. For instance, when the number of Grams is larger, the possible sequences of moves have more variations and as a consequence largest distances. In the case where the distance is $d = 0$, stays constant independently of the number of grams used. We conclude that there is an ascending trend in the distance between the games while the number o grams is bigger.

5.2 Game Concept Approaches

Game Concepts As it can be seen in the results of the Moves Frequency approach, there are games that cannot be separated using exclusively the move concepts, the Hex and Go example. That is the reason an approach that focuses only on game concepts was included. This algorithm works very similarly to the first approach, it takes the game concepts, saves the overlap and gives a distance metric. In the results section, the figure 7 was introduced to prove that the Knight Moves Games have different concepts even though they are all the same game. This metric was used as a base to later combine it with the move concepts metrics and find a better approach.

5.3 Combination of N-Grams and Game Concepts

After introducing the move concepts, sequences of moves captured by the N-Grams algorithm and the game concepts with the GameConcept algorithm, the idea to combine them was born. Additionally, the need to find the correct distribution of the importance(weights) for every part of the algorithm was rose.

According to the above mentioned, the main disadvantage of the first approach was unreasonable distance results for games from the same category. Furthermore, using the move fre-

quency algorithm and the algorithm of the N-Gram sequences of moves, lead us to another reliability problem. This is clearly illustrated with the results from the Hex and Go comparison (figure 2). Those games are completely different in practice, but they are very similar to the previously mentioned moves metrics.

In the search of finding a way to separate games like Hex and Go , an approach that combines game concepts and the previous moves approaches was developed. This new approach merges the concept approach and the n-grams, as it was mentioned before, to find the correct balance between them, the approach was tested with three different proportions in the Benchmark because this group includes a bigger variety of game categories. The first one gives more weight (0.7) to the game concepts and less weight (0.3) to the moves concepts part. The second one with equal proportions (0.5 and 0.5) and finally the third one with more importance (0.7) to the moves than to the game concepts (0.3). Using the Hex vs Go example as reference we see in the tables in the figures 8, 9, 10, the three results for this specific comparisons were $d = 0.4037$, $d = 0.2919$ and $d = 0.1801$ respectively. Looking at these three results, the one that gives a more logical metric is the $d = 0.4037$ coming from the (0.3) n-gram and (0.7) because since the game is very close in moving terms and far apart in game concepts, a mid-range distance was expected. Also to verify that these different weights do not change the expected result from other games we see how it affects mid range games such as Chess and English Draughts $d = 0.4531$ $d = 0.4863$ and $d = 0.5191$ and in far apart games such as Chess and Tic-Tac-Toe. The results for Chess versus English Draughts were $d = 0.4531$, $d = 0.4863$ and $d = 0.5191$, in all three scenarios they stayed in the mid-range spectrum and in the Chess vs Tic-Tac-Toe the given results were $d = 0.8149$, $d = 0.8678$ and 0.9207, they remain as far apart games.

After the analysis for these games, we perceive the best weight combination between the three was (0.3) n-grams and (0.7) game concepts. However, the inconvenience for this algorithm is evidenced in the Knight moves games. It has been mentioned before that the algorithm should look for a distance $d = 0$ in these specific games, but since they do not share the same concepts, by giving the largest importance to this part of the algorithm the distance between those games increases as it is shown in the figure 11.

6 Conclusion

In this research, different approaches were used to measure the similarity between board games in Ludii. The main focus involved the use of move concepts, although in some of the approaches there was the use of game concepts as well. In the last approach taken, a combination of move concepts and game concepts was applied, by using N-Grams algorithm to capture the sequence of moves and GameConcept algorithm, respectively.

The research was conducted with the aim to answer the following research questions:

- What does it mean for a board game to be similar or different?
- Is it possible to provide sufficient insights to measure the underlying difference between two games, based on the observed move concepts during the trials, combined with the previously named game concepts?
- Are the designed move-based distance metrics faster and more reliable in contrast with

the already existed distance metrics?

As for the first of these questions, what became obvious through this project is that defining what makes two games similar or different is a very complex issue that might not have an objective answer. This of course makes designing a metric for comparing games a challenging task. However, our approach is about the measurement of the similarity, as claimed, by the move based distances between games and with this objective, we can define the distance, hence the similarity between them. Therefore, to answer the first research question, we defined the move and the game concepts to measure the underlying differences between the two games. According to our approaches, if two games have overlapping move and game concepts then they tend to be similar. Otherwise, if two games have overlapping move concepts(or game concepts) but different game concepts(or move concepts) then they tend to be different.

As mentioned above, the similarity between games is a controversial problem to be answered. Nevertheless, after combining our approaches with frequency and N-grams we get a better result, but still, there is a place for improvement by adding more parameters, such as the game concepts. As a consequence, instead of using a move based approach, a "game concept" approach could be more reliable. The previously mentioned could answer the second research question of our project.

Finally, to answer the last research question, the implemented distance metric for measuring the similarity among games is fast. However, the time-consuming process is the running of the trials by using some AIs. By using a random AI the speed is acceptable, but when we use other AIs, which follow a strategy, to get more accurate comparisons, the speed of the algorithm decreases significantly. Finally, as regards the reliability compared to previous metrics, is comparable in performance and better in some cases.

7 Future Work

Due to time constraints in this project, there remain possible adjustments, which are not yet implemented. Firstly, we were not able to implement the promising similarity metric for n-grams as proposed by Kondrak [9] and introduced earlier in the report. Furthermore, we have primarily focused on the cosine similarity measure, hence a future task could involve testing and comparing different measures. As the data is very high dimensional and sparse, dimensionality reduction might be applicable too.

Because the project was initially based solely on moves, we only started thinking about game concepts in the last phase of the project. At that point, we discovered that it is possible some differences between games are not captured by move concepts on their own. Thus, we implemented the approach based on boolean game concepts. But, as we already know, there exist more complex non-boolean game concepts. Moreover, some game concepts are way more significant than others, i.e. ending conditions being one of the most important concepts.

Throughout the whole project, the trials were simulated using random agents only. This was one of the constraint to the project. During the experiments and their evaluation it became visible that some games differ from others by strategic depth while playing them, therefore resulting in sequences of moves with different frequencies. Thus, we propose to utilize AI's and random players. We expect this would allow to capture

more underlying differences, especially in strategically rich games like Chess or Go.

If the designed metrics yield sensible results, is it possible to answer further questions based on the approach:

- Can games that are a subset of other games be identified, i.e. whether any concept of one game is also found in another one?
- Is it possible to construct a classifier for the games based upon the results?

References

[1] Cameron Browne. Yavalath: Sample chapter from evolutionary game design. *J. Int. Comput. Games Assoc.*, 35:20–27, 2012.

[2] Cameron Browne. Ai for ancient games: Report on the digital ludeme project. *KI - Künstliche Intelligenz*, 34, 07 2019.

[3] Cameron Browne and Frederic Maire. Evolutionary game design. *Computational Intelligence and AI in Games, IEEE Transactions on*, 2:1 – 16, 04 2010.

[4] Cameron Browne, Matthew Stephenson, Éric Piette, and Dennis J. N. J. Soemers. A practical introduction to the ludii general game system. In Tristan Cazenave, Jaap van den Herik, Abdallah Saffidine, and I-Chen Wu, editors, *Advances in Computer Games*, pages 167–179, Cham, 2020. Springer International Publishing.

[5] Cameron Bolitho Browne. *Automatic generation and evaluation of recombination games*. PhD thesis, Queensland University of Technology, 2008.

[6] Browne Cameron, Piette Eric, and Stephenson Matthew. Ludii game and math concept taxonomies. 2021.

[7] Heinze E., Bapat P., Wu Z., Alferez S. V., A. V. Oria Ladeira, and M. Niebisch. Technical report of measuring game distance (tech. rep.). 2020.

[8] Shcherbakov Kirill, Abraham Frederic Marvin, Giannilias Theodoros, Penroz Valenzuela Diego, and Franzoni Darnois Guillaume. Final report game distance metric. 2021.

[9] Grzegorz Kondrak. N-gram similarity and distance. In *International symposium on string processing and information retrieval*, pages 115–126. Springer, 2005.

A Appendix

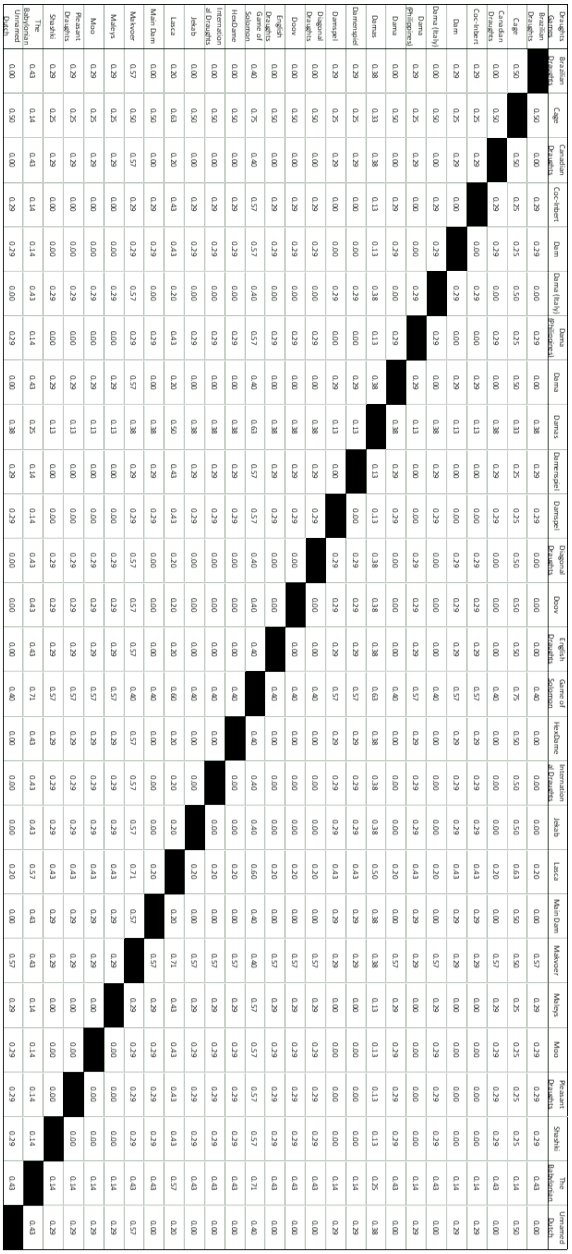

Figure 13: MoveConcept Overlap with Draught Games

Draughts Games	Brazilian Draughts	Cage	Canadian Draughts	Coc-Inbert	Dam	Dama (Italy)	Dama (Philippines)	Dama	Damas	Damenspiel	Damspel	Diagonal Draughts	Doov	English Draughts	Game of Solomon	HexDame	International Draughts	Jekab	Lasca	Main Dam	Makvoer	Maleys	Moo	Pleasant Draughts	Shashki	The Babylonian	Unnamed Dutch
Brazilian Draughts		0.50	0.00	0.29	0.29	0.00	0.29	0.00	0.38	0.29	0.29	0.00	0.00	0.00	0.40	0.00	0.00	0.00	0.20	0.00	0.57	0.29	0.29	0.29	0.29	0.43	0.00
Cage	0.50		0.50	0.25	0.25	0.50	0.25	0.50	0.33	0.25	0.25	0.50	0.50	0.50	0.75	0.50	0.50	0.50	0.63	0.50	0.50	0.25	0.25	0.25	0.25	0.14	0.50
Canadian Draughts	0.00	0.50		0.29	0.29	0.00	0.29	0.00	0.38	0.29	0.29	0.00	0.00	0.00	0.40	0.00	0.00	0.00	0.20	0.00	0.57	0.29	0.29	0.29	0.29	0.43	0.00
Coc-Inbert	0.29	0.25	0.29		0.00	0.29	0.00	0.29	0.13	0.00	0.00	0.29	0.29	0.29	0.57	0.29	0.29	0.29	0.43	0.29	0.29	0.00	0.00	0.00	0.00	0.14	0.29
Dam	0.29	0.25	0.29	0.00		0.29	0.00	0.29	0.13	0.00	0.00	0.29	0.29	0.29	0.57	0.29	0.29	0.29	0.43	0.29	0.29	0.00	0.00	0.00	0.00	0.14	0.29
Dama (Italy)	0.00	0.50	0.00	0.29	0.29		0.29	0.00	0.38	0.29	0.29	0.00	0.00	0.00	0.40	0.00	0.00	0.00	0.20	0.00	0.57	0.29	0.29	0.29	0.29	0.43	0.00
Dama (Philippines)	0.29	0.25	0.29	0.00	0.00	0.29		0.29	0.13	0.00	0.00	0.29	0.29	0.29	0.57	0.29	0.29	0.29	0.43	0.29	0.29	0.00	0.00	0.00	0.00	0.14	0.29
Dama	0.00	0.50	0.00	0.29	0.29	0.00	0.29		0.38	0.29	0.29	0.00	0.00	0.00	0.40	0.00	0.00	0.00	0.20	0.00	0.57	0.29	0.29	0.29	0.29	0.43	0.00
Damas	0.38	0.33	0.38	0.13	0.13	0.38	0.13	0.38		0.13	0.13	0.38	0.38	0.38	0.63	0.38	0.38	0.38	0.50	0.38	0.38	0.13	0.13	0.13	0.13	0.25	0.38
Damenspiel	0.29	0.25	0.29	0.00	0.00	0.29	0.00	0.29	0.13		0.00	0.29	0.29	0.29	0.57	0.29	0.29	0.29	0.43	0.29	0.29	0.00	0.00	0.00	0.00	0.14	0.29
Damspel	0.29	0.25	0.29	0.00	0.00	0.29	0.00	0.29	0.13	0.00		0.29	0.29	0.29	0.57	0.29	0.29	0.29	0.43	0.29	0.29	0.00	0.00	0.00	0.00	0.14	0.29
Diagonal Draughts	0.00	0.50	0.00	0.29	0.29	0.00	0.29	0.00	0.38	0.29	0.29		0.00	0.00	0.40	0.00	0.00	0.00	0.20	0.00	0.57	0.29	0.29	0.29	0.29	0.43	0.00
Doov	0.00	0.50	0.00	0.29	0.29	0.00	0.29	0.00	0.38	0.29	0.29	0.00		0.00	0.40	0.00	0.00	0.00	0.20	0.00	0.57	0.29	0.29	0.29	0.29	0.43	0.00
English Draughts	0.00	0.50	0.00	0.29	0.29	0.00	0.29	0.00	0.38	0.29	0.29	0.00	0.00		0.40	0.00	0.00	0.00	0.20	0.00	0.57	0.29	0.29	0.29	0.29	0.43	0.00
Game of Solomon	0.40	0.75	0.40	0.57	0.57	0.40	0.57	0.40	0.63	0.57	0.57	0.40	0.40	0.40		0.40	0.40	0.40	0.60	0.40	0.40	0.57	0.57	0.57	0.57	0.71	0.40
HexDame	0.00	0.50	0.00	0.29	0.29	0.00	0.29	0.00	0.38	0.29	0.29	0.00	0.00	0.00	0.40		0.00	0.00	0.20	0.00	0.57	0.29	0.29	0.29	0.29	0.43	0.00
International Draughts	0.00	0.50	0.00	0.29	0.29	0.00	0.29	0.00	0.38	0.29	0.29	0.00	0.00	0.00	0.40	0.00		0.00	0.20	0.00	0.57	0.29	0.29	0.29	0.29	0.43	0.00
Jekab	0.00	0.50	0.00	0.29	0.29	0.00	0.29	0.00	0.38	0.29	0.29	0.00	0.00	0.00	0.40	0.00	0.00		0.20	0.00	0.57	0.29	0.29	0.29	0.29	0.43	0.00
Lasca	0.20	0.63	0.20	0.43	0.43	0.20	0.43	0.20	0.50	0.43	0.43	0.20	0.20	0.20	0.60	0.20	0.20	0.20		0.20	0.71	0.43	0.43	0.43	0.43	0.57	0.20
Main Dam	0.00	0.50	0.00	0.29	0.29	0.00	0.29	0.00	0.38	0.29	0.29	0.00	0.00	0.00	0.40	0.00	0.00	0.00	0.20		0.57	0.29	0.29	0.29	0.29	0.43	0.00
Makvoer	0.57	0.50	0.57	0.29	0.29	0.57	0.29	0.57	0.38	0.29	0.29	0.57	0.57	0.57	0.40	0.57	0.57	0.57	0.71	0.57		0.29	0.29	0.29	0.29	0.43	0.57
Maleys	0.29	0.25	0.29	0.00	0.00	0.29	0.00	0.29	0.13	0.00	0.00	0.29	0.29	0.29	0.57	0.29	0.29	0.29	0.43	0.29	0.29		0.00	0.00	0.00	0.14	0.29
Moo	0.29	0.25	0.29	0.00	0.00	0.29	0.00	0.29	0.13	0.00	0.00	0.29	0.29	0.29	0.57	0.29	0.29	0.29	0.43	0.29	0.29	0.00		0.00	0.00	0.14	0.29
Pleasant Draughts	0.29	0.25	0.29	0.00	0.00	0.29	0.00	0.29	0.13	0.00	0.00	0.29	0.29	0.29	0.57	0.29	0.29	0.29	0.43	0.29	0.29	0.00	0.00		0.00	0.14	0.29
Shashki	0.29	0.25	0.29	0.00	0.00	0.29	0.00	0.29	0.13	0.00	0.00	0.29	0.29	0.29	0.57	0.29	0.29	0.29	0.43	0.29	0.29	0.00	0.00	0.00		0.14	0.29
The Babylonian	0.43	0.14	0.43	0.14	0.14	0.43	0.14	0.43	0.25	0.14	0.14	0.43	0.43	0.43	0.71	0.43	0.43	0.43	0.57	0.43	0.43	0.14	0.14	0.14	0.14		0.43
Unnamed Dutch	0.00	0.50	0.00	0.29	0.29	0.00	0.29	0.00	0.38	0.29	0.29	0.00	0.00	0.00	0.40	0.00	0.00	0.00	0.20	0.00	0.57	0.29	0.29	0.29	0.29	0.43	

Figure 14: Full Benchmark Games Frequency Moves

Benchmark Games	Tic-Tac-Toe	Breakthrough	Yavalath	Hex	Mu Torere	Round Merels	Chess	English Draughts	Taikyoku Shogi	Shogi	Go	Tai Shogi	Hahne	L Game	Konane	Backgammon	Plakoto	Cloister
Tic-Tac-Toe	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9969	0.9908	0.0046	0.9967	0.9778	0.9955	0.9968	0.9990	0.9999	1.0000
Breakthrough	1.0000	1.0000	0.0011	0.0011	0.0064	0.0064	0.5339	0.0134	0.0112	0.0981	0.9892	0.0373	0.8851	0.9777	0.0085	0.8828	0.8408	0.0427
Yavalath	1.0000	1.0000	1.0000	0.0000	1.0000	1.0000	1.0000	0.9999	0.9999	0.9997	0.0054	0.9999	0.9991	0.9990	0.9999	0.9995	0.9995	1.0000
Hex	1.0000	1.0000	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0054	1.0000	0.9999	1.0000	1.0000	1.0000	1.0000	1.0000
Mu Torere	1.0000	0.0064	1.0000	1.0000	0.0000	0.0000	0.5337	0.0172	0.0125	0.0923	1.0000	0.0412	0.8844	0.9775	0.0146	0.8834	0.8404	0.0814
Round Merels	1.0000	0.0064	1.0000	1.0000	0.0000	0.0000	0.5337	0.0172	0.0125	0.0923	1.0000	0.0412	0.8844	0.9775	0.0146	0.8834	0.8404	0.0814
Chess	1.0000	0.5339	1.0000	1.0000	0.5337	0.5337	0.0000	0.5312	0.5279	0.5571	0.9977	0.5766	0.9108	0.9496	0.5334	0.9370	0.9063	0.5621
English Draughts	1.0000	0.0134	0.9999	1.0000	0.0172	0.0172	0.5312	0.0000	0.0071	0.0527	0.9914	0.0167	0.8834	0.8829	0.0112	0.8408	0.7969	0.0618
Taikyoku Shogi	0.9969	0.0112	0.9999	1.0000	0.0125	0.0125	0.5279	0.0071	0.0000	0.0477	0.9909	0.0135	0.7611	0.8655	0.0032	0.8340	0.7909	0.0663
Shogi	0.9908	0.0981	0.9997	1.0000	0.0923	0.0923	0.5571	0.0527	0.0477	0.0000	0.9922	0.0438	0.5769	0.6496	0.0527	0.7468	0.7046	0.1662
Go	0.0046	0.9892	0.0054	0.0054	1.0000	1.0000	0.9977	0.9914	0.9909	0.9922	0.0000	0.9887	0.9812	0.9962	0.9868	0.9980	0.9993	0.9672
Tai Shogi	0.9967	0.0373	0.9999	1.0000	0.0412	0.0412	0.5766	0.0167	0.0135	0.0438	0.9887	0.0000	0.7424	0.8928	0.0263	0.8265	0.7843	0.0840
Hahne	0.9778	0.8851	0.9991	0.9999	0.8844	0.8844	0.9108	0.8834	0.7611	0.5769	0.9812	0.7424	0.0000	0.0865	0.7570	0.5797	0.5644	0.8938
L Game	0.9955	0.9777	0.9990	0.9999	0.9775	0.9775	0.9496	0.8829	0.8655	0.6496	0.9962	0.8928	0.0865	0.0000	0.8820	0.5415	0.5199	0.9794
Konane	0.9968	0.0085	0.9999	1.0000	0.0146	0.0146	0.5334	0.0112	0.0032	0.0527	0.9868	0.0263	0.7570	0.8820	0.0000	0.8321	0.7894	0.0517
Backgammon	0.9990	0.8828	0.9995	1.0000	0.8834	0.8834	0.9370	0.8408	0.8340	0.7468	0.9980	0.8265	0.5797	0.5415	0.8321	0.0000	0.0019	0.8882
Plakoto	0.9999	0.8408	0.9995	1.0000	0.8404	0.8404	0.9063	0.7969	0.7909	0.7046	0.9993	0.7843	0.5644	0.5199	0.7894	0.0019	0.0000	0.8511
Cloister	1.0000	0.0427	1.0000	1.0000	0.0814	0.0814	0.5621	0.0618	0.0663	0.1662	0.9672	0.0840	0.8938	0.9794	0.0517	0.8882	0.8511	0.0000

Figure 15: Full Draught Games Frequency moves

Draught Games	Brazilian Draughts	Cage	Canadian Draughts	Coc-Inbert	Dam	Dama (Italy)	Dama (Philippines)	Dama	Damas	Damenspiel	Damspel	Diagonal Draughts	Doov	English Draughts	Game of Solomon	HexDame	International Draughts	Jekab	Lasca	Main Dam	Makvoer	Moo	Pleasant Draughts	Shashki	The Babylonian	Unnamed Dutch
Brazilian Draughts		0.22	0.00	0.00	0.02	0.00	0.03	0.01	0.22	0.00	0.02	0.00	0.08	0.00	0.01	0.01	0.00	0.00	0.03	0.00	0.02	0.02	0.07	0.00	0.05	0.01
Cage	0.22		0.21	0.23	0.22	0.23	0.25	0.21	0.40	0.22	0.24	0.21	0.30	0.23	0.24	0.21	0.21	0.22	0.23	0.22	0.25	0.24	0.28	0.22	0.23	0.21
Canadian Draughts	0.00	0.21		0.01	0.01	0.01	0.04	0.00	0.23	0.01	0.03	0.00	0.10	0.01	0.03	0.00	0.00	0.00	0.03	0.01	0.03	0.03	0.08	0.00	0.04	0.00
Coc-Inbert	0.00	0.23	0.01		0.02	0.00	0.01	0.01	0.21	0.00	0.01	0.01	0.05	0.00	0.01	0.02	0.01	0.00	0.03	0.00	0.01	0.01	0.05	0.00	0.06	0.02
Dam	0.02	0.22	0.01	0.02		0.02	0.04	0.01	0.23	0.02	0.01	0.01	0.10	0.02	0.04	0.01	0.01	0.02	0.03	0.02	0.04	0.02	0.05	0.01	0.01	0.01
Dama (Italy)	0.00	0.23	0.01	0.00	0.02		0.02	0.01	0.22	0.00	0.02	0.00	0.05	0.00	0.01	0.02	0.01	0.00	0.02	0.00	0.01	0.02	0.05	0.00	0.07	0.02
Dama (Philippines)	0.03	0.25	0.04	0.01	0.04	0.02		0.04	0.22	0.03	0.01	0.03	0.02	0.02	0.02	0.05	0.04	0.03	0.02	0.02	0.02	0.01	0.02	0.03	0.09	0.05
Dama	0.01	0.21	0.00	0.01	0.01	0.01	0.04		0.23	0.01	0.02	0.00	0.09	0.01	0.03	0.00	0.00	0.01	0.01	0.01	0.04	0.03	0.07	0.01	0.04	0.00
Damas	0.22	0.40	0.23	0.21	0.23	0.22	0.22	0.23		0.22	0.23	0.22	0.25	0.21	0.21	0.24	0.23	0.22	0.24	0.22	0.21	0.22	0.25	0.22	0.27	0.24
Damenspiel	0.00	0.22	0.01	0.00	0.02	0.00	0.03	0.01	0.22		0.02	0.00	0.08	0.00	0.01	0.01	0.00	0.00	0.03	0.00	0.01	0.02	0.06	0.00	0.05	0.02
Damspel	0.02	0.24	0.03	0.01	0.01	0.02	0.01	0.02	0.23	0.02		0.02	0.04	0.02	0.04	0.03	0.03	0.02	0.01	0.02	0.03	0.00	0.02	0.02	0.04	0.03
Diagonal Draughts	0.00	0.21	0.00	0.01	0.01	0.00	0.03	0.00	0.22	0.00	0.02		0.08	0.01	0.02	0.01	0.00	0.00	0.03	0.00	0.02	0.02	0.07	0.00	0.05	0.01
Doov	0.08	0.30	0.10	0.05	0.10	0.05	0.02	0.09	0.25	0.08	0.04	0.08		0.05	0.06	0.10	0.09	0.08	0.04	0.07	0.05	0.04	0.03	0.08	0.16	0.11
English Draughts	0.00	0.23	0.01	0.00	0.02	0.00	0.02	0.01	0.21	0.00	0.02	0.01	0.05		0.01	0.02	0.01	0.00	0.03	0.00	0.01	0.02	0.05	0.00	0.07	0.03
Game of Solomon	0.01	0.24	0.03	0.01	0.04	0.01	0.02	0.03	0.21	0.01	0.04	0.02	0.06	0.01		0.04	0.02	0.01	0.05	0.00	0.00	0.03	0.07	0.01	0.10	0.05
HexDame	0.01	0.21	0.00	0.02	0.01	0.02	0.05	0.00	0.24	0.01	0.03	0.01	0.10	0.02	0.04		0.01	0.01	0.02	0.02	0.05	0.03	0.08	0.01	0.03	0.00
International Draughts	0.00	0.21	0.00	0.01	0.01	0.01	0.04	0.00	0.23	0.00	0.03	0.00	0.09	0.01	0.02	0.01		0.00	0.03	0.01	0.03	0.03	0.08	0.00	0.04	0.01
Jekab	0.00	0.22	0.00	0.00	0.02	0.00	0.03	0.01	0.22	0.00	0.02	0.00	0.08	0.00	0.01	0.01	0.00		0.03	0.00	0.01	0.02	0.07	0.00	0.06	0.01
Lasca	0.03	0.23	0.03	0.03	0.03	0.02	0.02	0.01	0.24	0.03	0.01	0.03	0.04	0.03	0.05	0.02	0.03	0.03		0.03	0.04	0.02	0.04	0.03	0.05	0.02
Main Dam	0.00	0.22	0.01	0.00	0.02	0.00	0.02	0.01	0.22	0.00	0.02	0.00	0.07	0.00	0.00	0.02	0.01	0.00	0.03		0.01	0.02	0.07	0.00	0.07	0.02
Makvoer	0.02	0.25	0.03	0.01	0.04	0.01	0.02	0.04	0.21	0.01	0.03	0.02	0.05	0.01	0.00	0.05	0.03	0.01	0.04	0.01		0.02	0.06	0.01	0.10	0.05
Moo	0.02	0.24	0.03	0.01	0.02	0.02	0.01	0.03	0.22	0.02	0.00	0.02	0.04	0.02	0.03	0.03	0.03	0.02	0.02	0.02	0.02		0.01	0.02	0.05	0.04
Pleasant Draughts	0.07	0.28	0.08	0.05	0.05	0.05	0.02	0.07	0.25	0.06	0.02	0.07	0.03	0.05	0.07	0.08	0.08	0.07	0.04	0.07	0.06	0.01		0.07	0.09	0.09
Shashki	0.00	0.22	0.00	0.00	0.01	0.00	0.03	0.01	0.22	0.00	0.02	0.00	0.08	0.00	0.01	0.01	0.00	0.00	0.03	0.00	0.01	0.02	0.07		0.05	0.01
The Babylonian	0.05	0.23	0.04	0.06	0.01	0.07	0.09	0.04	0.27	0.05	0.04	0.05	0.16	0.07	0.10	0.03	0.04	0.06	0.05	0.07	0.10	0.05	0.09	0.05		0.03
Unnamed Dutch	0.01	0.21	0.00	0.02	0.01	0.02	0.05	0.00	0.24	0.02	0.03	0.01	0.11	0.03	0.05	0.00	0.01	0.01	0.02	0.02	0.05	0.04	0.09	0.01	0.03	

Figure 16: Full Knight Moves Frequency moves

Knight Games	Knight Moves 1	Knight Moves 2	Knight Moves 3	Knight Moves 4	Knight Moves 5	Knight Moves 6	Knight Moves 7	Knight Moves 8	Knight Moves 9	Knight Moves 10	Knight Moves 11	Knight Moves 12	Knight Moves 13	Knight Moves 14	Knight Moves 15	Knight Moves 16	Knight Moves 17	Knight Moves 18	Knight Moves 19	Knight Moves 20
Knight Moves 1		0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004
Knight Moves 2	0.000		0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000
Knight Moves 3	0.004	0.000		0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004
Knight Moves 4	0.000	0.004	0.000		0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004
Knight Moves 5	0.004	0.000	0.004	0.000		0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000
Knight Moves 6	0.000	0.004	0.000	0.004	0.000		0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004
Knight Moves 7	0.004	0.000	0.004	0.000	0.004	0.000		0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000
Knight Moves 8	0.000	0.004	0.000	0.004	0.000	0.004	0.000		0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004
Knight Moves 9	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000		0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000
Knight Moves 10	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000		0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000
Knight Moves 11	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000		0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004
Knight Moves 12	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004		0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004
Knight Moves 13	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000		0.000	0.004	0.000	0.004	0.000	0.004	0.000
Knight Moves 14	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004		0.000	0.004	0.000	0.004	0.000	0.004
Knight Moves 15	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000		0.000	0.004	0.000	0.004	0.000
Knight Moves 16	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004		0.000	0.004	0.000	0.004
Knight Moves 17	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000		0.004	0.000	0.004
Knight Moves 18	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000		0.004	0.000
Knight Moves 19	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000		0.004
Knight Moves 20	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	0.000	0.004	

Figure 17: Full Knight Moves 2-grams

Knight Moves	Knight Moves 1	Knight Moves 1	Knight Moves 3	Knight Moves 4	Knight Moves 5	Knight Moves 6	Knight Moves 7	Knight Moves 8	Knight Moves 9	Knight Moves 10	Knight Moves 11	Knight Moves 12	Knight Moves 13	Knight Moves 14	Knight Moves 15	Knight Moves 16	Knight Moves 17	Knight Moves 18	Knight Moves 19	Knight Moves 20
Knight Moves 1	0.000	0.000	0.007	0.007	0.007	0.007	0.000	0.001	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.000	0.000	0.007
Knight Moves 1	0.000	0.000	0.007	0.007	0.007	0.007	0.000	0.001	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.000	0.000	0.007
Knight Moves 3	0.007	0.007	0.000	0.000	0.000	0.000	0.007	0.011	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.000
Knight Moves 4	0.007	0.007	0.000	0.000	0.000	0.000	0.007	0.011	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.000
Knight Moves 5	0.007	0.007	0.000	0.000	0.000	0.000	0.007	0.011	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.000
Knight Moves 6	0.000	0.000	0.007	0.007	0.007	0.007	0.000	0.001	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.000	0.000	0.007
Knight Moves 7	0.000	0.000	0.007	0.007	0.007	0.007	0.000	0.001	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.000	0.000	0.007
Knight Moves 8	0.000	0.000	0.007	0.007	0.007	0.007	0.000	0.001	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.000	0.000	0.007
Knight Moves 9	0.001	0.001	0.011	0.011	0.011	0.011	0.001	0.001	0.001	0.001	0.011	0.011	0.011	0.011	0.011	0.011	0.011	0.001	0.001	0.011
Knight Moves 10	0.007	0.007	0.000	0.000	0.000	0.000	0.007	0.011	0.007	0.007	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.007	0.007	0.000
Knight Moves 11	0.007	0.007	0.000	0.000	0.000	0.000	0.007	0.011	0.007	0.007	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.007	0.007	0.000
Knight Moves 12	0.007	0.007	0.000	0.000	0.000	0.000	0.007	0.011	0.007	0.007	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.007	0.007	0.000
Knight Moves 13	0.007	0.007	0.000	0.000	0.000	0.000	0.007	0.011	0.007	0.007	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.007	0.007	0.000
Knight Moves 14	0.007	0.007	0.000	0.000	0.000	0.000	0.007	0.011	0.007	0.007	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.007	0.007	0.000
Knight Moves 15	0.007	0.007	0.000	0.000	0.000	0.000	0.007	0.011	0.007	0.007	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.007	0.007	0.000
Knight Moves 16	0.007	0.007	0.000	0.000	0.000	0.000	0.007	0.011	0.007	0.007	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.007	0.007	0.000
Knight Moves 17	0.007	0.007	0.000	0.000	0.000	0.000	0.007	0.011	0.007	0.007	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.007	0.007	0.000
Knight Moves 18	0.000	0.000	0.007	0.007	0.007	0.007	0.000	0.001	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.000	0.000	0.007
Knight Moves 19	0.000	0.000	0.007	0.007	0.007	0.007	0.000	0.001	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.007	0.000	0.000	0.007
Knight Moves 20	0.007	0.007	0.000	0.000	0.000	0.000	0.007	0.011	0.007	0.007	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.007	0.007	0.000

Figure 18: Full Knight Moves 3-grams

Knight Moves	Knight Moves 1	Knight Moves 1	Knight Moves 3	Knight Moves 4	Knight Moves 5	Knight Moves 6	Knight Moves 7	Knight Moves 8	Knight Moves 9	Knight Moves 10	Knight Moves 11	Knight Moves 12	Knight Moves 13	Knight Moves 14	Knight Moves 15	Knight Moves 16	Knight Moves 17	Knight Moves 18	Knight Moves 19	Knight Moves 20
Knight Moves 1		0.000	0.009	0.009	0.009	0.000	0.000	0.000	0.001	0.009	0.009	0.009	0.009	0.009	0.009	0.009	0.009	0.000	0.000	0.009
Knight Moves 1	0.000		0.009	0.009	0.009	0.000	0.000	0.000	0.001	0.009	0.009	0.009	0.009	0.009	0.009	0.009	0.009	0.000	0.000	0.009
Knight Moves 3	0.009	0.009		0.000	0.000	0.000	0.008	0.009	0.015	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.009	0.009	0.000
Knight Moves 4	0.009	0.009	0.000		0.000	0.008	0.009	0.009	0.015	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.009	0.009	0.000
Knight Moves 5	0.009	0.009	0.000	0.000		0.008	0.009	0.009	0.015	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.009	0.009	0.000
Knight Moves 6	0.009	0.009	0.000	0.000	0.000		0.008	0.009	0.015	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.009	0.009	0.000
Knight Moves 7	0.000	0.000	0.008	0.008	0.008	0.008		0.000	0.001	0.008	0.008	0.008	0.008	0.008	0.008	0.008	0.008	0.000	0.000	0.008
Knight Moves 8	0.000	0.000	0.009	0.009	0.009	0.009	0.000		0.001	0.009	0.009	0.009	0.009	0.009	0.009	0.009	0.009	0.000	0.000	0.009
Knight Moves 9	0.001	0.001	0.015	0.015	0.015	0.015	0.001	0.001		0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.001	0.001	0.015
Knight Moves 10	0.009	0.009	0.000	0.000	0.000	0.000	0.008	0.009	0.015		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.009	0.009	0.000
Knight Moves 11	0.009	0.009	0.000	0.000	0.000	0.000	0.008	0.009	0.015	0.000		0.000	0.000	0.000	0.000	0.000	0.000	0.009	0.009	0.000
Knight Moves 12	0.009	0.009	0.000	0.000	0.000	0.000	0.008	0.009	0.015	0.000	0.000		0.000	0.000	0.000	0.000	0.000	0.009	0.009	0.000
Knight Moves 13	0.009	0.009	0.000	0.000	0.000	0.000	0.008	0.009	0.015	0.000	0.000	0.000		0.000	0.000	0.000	0.000	0.009	0.009	0.000
Knight Moves 14	0.009	0.009	0.000	0.000	0.000	0.000	0.008	0.009	0.015	0.000	0.000	0.000	0.000		0.000	0.000	0.000	0.009	0.009	0.000
Knight Moves 15	0.009	0.009	0.000	0.000	0.000	0.000	0.008	0.009	0.015	0.000	0.000	0.000	0.000	0.000		0.000	0.000	0.009	0.009	0.000
Knight Moves 16	0.009	0.009	0.000	0.000	0.000	0.000	0.008	0.009	0.015	0.000	0.000	0.000	0.000	0.000	0.000		0.000	0.009	0.009	0.000
Knight Moves 17	0.009	0.009	0.000	0.000	0.000	0.000	0.008	0.009	0.015	0.000	0.000	0.000	0.000	0.000	0.000	0.000		0.009	0.009	0.000
Knight Moves 18	0.000	0.000	0.009	0.009	0.009	0.009	0.000	0.000	0.001	0.009	0.009	0.009	0.009	0.009	0.009	0.009	0.009		0.000	0.009
Knight Moves 19	0.000	0.000	0.009	0.009	0.009	0.009	0.000	0.000	0.001	0.009	0.009	0.009	0.009	0.009	0.009	0.009	0.009	0.000		0.009
Knight Moves 20	0.009	0.009	0.000	0.000	0.000	0.000	0.008	0.009	0.015	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.009	0.009	

Figure 19: Full Knight Moves 5-grams

Knight Moves	Knight Moves 1	Knight Moves 2	Knight Moves 3	Knight Moves 4	Knight Moves 5	Knight Moves 6	Knight Moves 7	Knight Moves 8	Knight Moves 9	Knight Moves 10	Knight Moves 11	Knight Moves 12	Knight Moves 13	Knight Moves 14	Knight Moves 15	Knight Moves 16	Knight Moves 17	Knight Moves 18	Knight Moves 19	Knight Moves 20
Knight Moves 1	0.000	0.000	0.016	0.016	0.016	0.000	0.000	0.000	0.002	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.000	0.000	0.016
Knight Moves 1	0.000	0.000	0.016	0.016	0.016	0.000	0.000	0.000	0.002	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.000	0.000	0.016
Knight Moves 3	0.016	0.016	0.000	0.000	0.000	0.000	0.000	0.000	0.002	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.000	0.000	0.016
Knight Moves 4	0.016	0.016	0.000	0.000	0.000	0.000	0.000	0.000	0.002	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.000	0.000	0.016
Knight Moves 5	0.016	0.016	0.000	0.000	0.000	0.000	0.000	0.000	0.002	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.000	0.000	0.016
Knight Moves 6	0.000	0.000	0.016	0.016	0.016	0.000	0.000	0.000	0.002	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.000	0.000	0.016
Knight Moves 7	0.000	0.000	0.016	0.016	0.016	0.000	0.000	0.000	0.002	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.000	0.000	0.016
Knight Moves 8	0.000	0.000	0.017	0.017	0.017	0.000	0.000	0.000	0.002	0.017	0.017	0.017	0.017	0.017	0.017	0.017	0.017	0.000	0.000	0.017
Knight Moves 9	0.002	0.002	0.029	0.029	0.029	0.002	0.002	0.002	0.002	0.029	0.029	0.029	0.029	0.029	0.029	0.029	0.029	0.002	0.002	0.029
Knight Moves 10	0.016	0.016	0.000	0.000	0.000	0.000	0.016	0.016	0.029	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.016	0.016	0.000
Knight Moves 11	0.016	0.016	0.000	0.000	0.000	0.000	0.016	0.017	0.029	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.016	0.016	0.000
Knight Moves 12	0.016	0.016	0.000	0.000	0.000	0.000	0.016	0.017	0.029	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.016	0.016	0.000
Knight Moves 13	0.016	0.016	0.000	0.000	0.000	0.000	0.016	0.017	0.029	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.016	0.016	0.000
Knight Moves 14	0.016	0.016	0.000	0.000	0.000	0.000	0.016	0.017	0.029	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.016	0.016	0.000
Knight Moves 15	0.016	0.016	0.000	0.000	0.000	0.000	0.016	0.017	0.029	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.016	0.016	0.000
Knight Moves 16	0.016	0.016	0.000	0.000	0.000	0.000	0.016	0.017	0.029	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.016	0.016	0.000
Knight Moves 17	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.017	0.029	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.016	0.016	0.000
Knight Moves 18	0.000	0.000	0.016	0.016	0.016	0.000	0.000	0.000	0.002	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.000	0.000	0.016
Knight Moves 19	0.000	0.000	0.016	0.016	0.016	0.000	0.000	0.000	0.002	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.000	0.000	0.016
Knight Moves 20	0.016	0.016	0.000	0.000	0.000	0.000	0.016	0.017	0.029	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.016	0.016	0.000

Figure 20: Full Draught games 2-grams

Figure 21: Full Draught games 3-grams

Figure 22: Full Draught games 5-grams

Figure 23: Full Knight Moves Game Concept Overlap

Knight Moves	Knight Moves 1	Knight Moves 1	Knight Moves 3	Knight Moves 4	Knight Moves 5	Knight Moves 6	Knight Moves 7	Knight Moves 8	Knight Moves 9	Knight Moves 10	Knight Moves 11	Knight Moves 12	Knight Moves 13	Knight Moves 14	Knight Moves 15	Knight Moves 16	Knight Moves 17	Knight Moves 18	Knight Moves 19	Knight Moves 20
Knight Moves 1		0.20	0.23	0.24	0.25	0.29	0.20	0.23	0.27	0.23	0.24	0.24	0.30	0.27	0.25	0.23	0.27	0.21	0.15	0.19
Knight Moves 1	0.20		0.18	0.19	0.21	0.24	0.09	0.13	0.09	0.24	0.24	0.17	0.25	0.27	0.25	0.18	0.27	0.11	0.13	0.11
Knight Moves 3	0.23	0.18		0.11	0.13	0.08	0.13	0.22	0.18	0.19	0.19	0.14	0.18	0.23	0.20	0.10	0.20	0.14	0.16	0.08
Knight Moves 4	0.24	0.19	0.11		0.05	0.09	0.14	0.15	0.16	0.08	0.20	0.10	0.11	0.16	0.10	0.08	0.21	0.15	0.20	0.10
Knight Moves 5	0.25	0.21	0.13	0.05		0.05	0.15	0.11	0.18	0.07	0.22	0.05	0.09	0.14	0.11	0.07	0.20	0.16	0.21	0.11
Knight Moves 6	0.29	0.24	0.08	0.09	0.05		0.16	0.14	0.21	0.11	0.25	0.09	0.13	0.18	0.15	0.11	0.21	0.17	0.22	0.16
Knight Moves 7	0.20	0.09	0.13	0.14	0.15	0.16		0.10	0.06	0.21	0.24	0.17	0.20	0.25	0.22	0.13	0.27	0.02	0.13	0.11
Knight Moves 8	0.23	0.13	0.22	0.15	0.11	0.14	0.10		0.10	0.14	0.28	0.09	0.19	0.20	0.18	0.16	0.25	0.09	0.16	0.15
Knight Moves 9	0.22	0.09	0.18	0.16	0.18	0.21	0.06	0.10		0.23	0.27	0.19	0.23	0.27	0.25	0.15	0.29	0.05	0.15	0.14
Knight Moves 10	0.23	0.24	0.19	0.08	0.07	0.11	0.21	0.14	0.23		0.19	0.08	0.15	0.11	0.13	0.17	0.17	0.22	0.19	0.15
Knight Moves 11	0.24	0.24	0.14	0.10	0.05	0.09	0.17	0.09	0.19	0.19		0.20	0.24	0.20	0.18	0.13	0.20	0.25	0.18	0.07
Knight Moves 12	0.24	0.17	0.14	0.10	0.05	0.09	0.17	0.09	0.19	0.08	0.20		0.14	0.16	0.10	0.11	0.18	0.18	0.20	0.07
Knight Moves 13	0.30	0.25	0.18	0.11	0.09	0.13	0.20	0.19	0.23	0.15	0.24	0.14		0.05	0.14	0.13	0.17	0.21	0.26	0.17
Knight Moves 14	0.27	0.27	0.23	0.16	0.14	0.18	0.25	0.20	0.27	0.11	0.20	0.16	0.05		0.10	0.17	0.13	0.26	0.23	0.19
Knight Moves 15	0.25	0.25	0.20	0.10	0.11	0.15	0.22	0.18	0.25	0.05	0.18	0.13	0.14	0.10		0.11	0.13	0.24	0.20	0.16
Knight Moves 16	0.23	0.18	0.10	0.08	0.07	0.11	0.13	0.16	0.15	0.13	0.19	0.11	0.13	0.17	0.11		0.17	0.14	0.19	0.08
Knight Moves 17	0.27	0.27	0.14	0.21	0.20	0.21	0.27	0.25	0.29	0.17	0.20	0.18	0.17	0.13	0.13	0.17		0.28	0.14	0.19
Knight Moves 18	0.21	0.11	0.16	0.15	0.16	0.17	0.27	0.25	0.05	0.22	0.25	0.18	0.21	0.26	0.24	0.14	0.28		0.14	0.13
Knight Moves 19	0.15	0.13	0.14	0.20	0.21	0.22	0.13	0.16	0.15	0.19	0.20	0.20	0.23	0.25	0.24	0.19	0.26	0.14		0.15
Knight Moves 20	0.19	0.11	0.08	0.10	0.11	0.16	0.11	0.15	0.14	0.15	0.15	0.07	0.17	0.19	0.16	0.08	0.19	0.13	0.15	

Figure 24: Full Benchmark Games with balance 0.3 3-grams and 0.7 game concept

Benchmark Games	Tic-Tac-Toe	Breakthrough	Yavalath	Hex	Mu Torere	Round Merels	Chess	English Draughts	Taikyoku Shogi	Shogi	Go	Tai Shogi	Hnefatafl	Hnefatafl	Konane	Backgammon	Pikoko	Clobber
Tic-Tac-Toe		0.72	0.2002	0.2559	0.7239	0.6652	0.8189	0.7534	0.8277	0.7804	0.3916	0.8236	0.7591	0.7436	0.7182	0.8124	0.7989	0.6667
Breakthrough	0.72		0.7519	0.743	0.355	0.6567	0.4894	0.2634	0.3585	0.5122	0.7109	0.3777	0.5858	0.7055	0.3354	0.5927	0.5855	0.2498
Yavalath	0.2002	0.7519		0.1043	0.7181	0.66	0.832	0.7793	0.8433	0.8019	0.3998	0.8397	0.7835	0.7764	0.7375	0.7467	0.8267	0.6924
Hex	0.2559	0.743	0.1043		0.7219	0.6932	0.8264	0.7717	0.8381	0.7953	0.4037	0.8909	0.7763	0.7795	0.7467	0.8267	0.8143	0.697
Mu Torere	0.7239	0.355	0.7181	0.7219		0.4239	0.572	0.3168	0.4539	0.5508	0.7023	0.46	0.5743	0.6826	0.3546	0.5572	0.5304	0.2852
Round Merels	0.6652	0.6567	0.66	0.6932	0.4239		0.7527	0.657	0.756	0.5779	0.6898	0.7496	0.6725	0.747	0.6774	0.7186	0.703	0.6298
Chess	0.8189	0.4894	0.832	0.8264	0.572	0.7527		0.8531	0.3959	0.4984	0.7632	0.3876	0.6145	0.7252	0.5502	0.6419	0.6574	0.5544
English Draughts	0.7534	0.2634	0.7793	0.7717	0.3168	0.657	0.8531		0.3415	0.4563	0.7118	0.3454	0.4362	0.6425	0.2375	0.5303	0.5323	0.3048
Taikyoku Shogi	0.8277	0.3585	0.8433	0.8381	0.4539	0.756	0.3959	0.3415		0.4717	0.7882	0.0555	0.589	0.7341	0.4038	0.5984	0.5872	0.4493
Shogi	0.7804	0.5122	0.8019	0.7953	0.5508	0.5779	0.4984	0.4563	0.4717		0.7405	0.4533	0.5018	0.6601	0.5318	0.669	0.6658	0.5719
Go	0.3916	0.7109	0.3998	0.4037	0.7023	0.6898	0.7632	0.7118	0.7882	0.7405		0.7599	0.7	0.7305	0.6656	0.7318	0.7173	0.6675
Tai Shogi	0.8236	0.3777	0.8397	0.8409	0.46	0.7496	0.3876	0.3454	0.0555	0.4533	0.7599		0.5823	0.7196	0.4016	0.5883	0.7173	0.4482
Hnefatafl	0.7591	0.5858	0.7835	0.7763	0.5743	0.6725	0.6145	0.4362	0.589	0.5018	0.7	0.5823		0.6136	0.5013	0.6362	0.6447	0.6199
Hnefatafl	0.7436	0.7055	0.7764	0.7795	0.6826	0.747	0.7252	0.6425	0.7341	0.6601	0.7305	0.7196	0.6136		0.6086	0.6785	0.6711	0.6538
Konane	0.7182	0.3354	0.7375	0.7467	0.3546	0.6774	0.5502	0.2375	0.4038	0.5318	0.6656	0.4016	0.5013	0.6086		0.5619	0.564	0.2523
Backgammon	0.8124	0.5927	0.8337	0.8267	0.5572	0.7186	0.6419	0.5303	0.5984	0.669	0.7318	0.5883	0.6362	0.6785	0.5619		0.6685	0.5811
Pikoko	0.7989	0.5855	0.8214	0.8143	0.5304	0.703	0.6574	0.5323	0.5872	0.6658	0.7173	0.5777	0.6447	0.6711	0.564	0.6685		0.5859
Clobber	0.6667	0.2498	0.6924	0.697	0.2852	0.6298	0.5544	0.3048	0.4493	0.5719	0.6675	0.4482	0.6199	0.6538	0.2523	0.5811	0.5859	

Figure 25: Full Benchmark Games with balance 0.5 3-grams and 0.5 game concept

Benchmark Games	Tic-Tac-Toe	Breakthrough	Yavalath	Hex	Mu Torere	Round Merels	Chess	English Draughts	Taikyoku Shogi	Shogi	Go	Tai Shogi	Hahna	L Game	Konane	Backgammon	Plakoto	Clobber
Tic-Tac-Toe		0.88	0.0861	0.11	0.8817	0.8565	0.9207	0.8943	0.9262	0.9059	0.1749	0.9244	0.8968	0.8897	0.8792	0.9196	0.9138	0.8571
Breakthrough	0.88		0.8937	0.8899	0.17	0.8263	0.5337	0.1243	0.1592	0.6011	0.8761	0.1758	0.7137	0.8657	0.147	0.5702	0.5281	0.1257
Yavalath	0.0861	0.8937		0.0447	0.8792	0.8543	0.928	0.9054	0.9328	0.9151	0.1784	0.9313	0.9072	0.9042	0.8875	0.9287	0.9235	0.8682
Hex	0.11	0.8899	0.0447		0.8808	0.8685	0.9256	0.9022	0.9306	0.9123	0.1801	0.9318	0.9041	0.9055	0.8889	0.9257	0.9204	0.8701
Mu Torere	0.8817	0.17	0.8792	0.8808		0.622	0.5727	0.1569	0.2144	0.5733	0.8724	0.2757	0.7092	0.856	0.1688	0.5579	0.5078	0.1557
Round Merels	0.8565	0.8263	0.8543	0.8685	0.622		0.8916	0.8172	0.895	0.5656	0.867	0.8904	0.8553	0.8916	0.8533	0.8794	0.8717	0.8389
Chess	0.9207	0.5337	0.928	0.9256	0.5727	0.8916		0.5194	0.492	0.7005	0.8985	0.4927	0.7774	0.8786	0.5575	0.7331	0.7224	0.5619
English Draughts	0.8943	0.1243	0.9054	0.9022	0.1569	0.8172	0.5194		0.4912	0.549	0.8765	0.1652	0.5778	0.8249	0.1097	0.546	0.5082	0.1583
Taikyoku Shogi	0.9262	0.1592	0.9328	0.9306	0.2144	0.895	0.492	0.1557		0.5794	0.9099	0.0337	0.674	0.8769	0.1746	0.5708	0.5267	0.2124
Shogi	0.9059	0.6011	0.9151	0.9123	0.5733	0.5656	0.7005	0.549	0.5794		0.8888	0.5718	0.5744	0.8964	0.606	0.7761	0.7621	0.6405
Go	0.1749	0.8761	0.1784	0.1801	0.8724	0.867	0.8985	0.8765	0.9099	0.8888		0.8971	0.8714	0.8884	0.8567	0.885	0.8788	0.8575
Tai Shogi	0.9244	0.1758	0.9313	0.9318	0.2757	0.8904	0.4927	0.1652	0.0337	0.5718	0.8971		0.658	0.8553	0.1826	0.5707	0.5275	0.722
Hahna	0.8968	0.7137	0.9072	0.9041	0.7092	0.8553	0.7774	0.5778	0.674	0.5744	0.8714	0.658		0.7578	0.6449	0.794	0.79	0.7296
L Game	0.8897	0.8657	0.9042	0.9055	0.856	0.8916	0.8786	0.8249	0.8769	0.8964	0.8884	0.8553	0.7578		0.8141	0.8585	0.8547	0.8446
Konane	0.8792	0.147	0.8875	0.8889	0.1688	0.8533	0.5575	0.1097	0.1746	0.606	0.8567	0.1826	0.6449	0.8141		0.5546	0.5161	0.1257
Backgammon	0.9196	0.5702	0.9287	0.9257	0.5579	0.8794	0.7331	0.546	0.5708	0.7761	0.885	0.5707	0.794	0.8585	0.5546		0.0344	0.57
Plakoto	0.9138	0.5281	0.9235	0.9204	0.5078	0.8717	0.7224	0.5082	0.5267	0.7621	0.8788	0.5275	0.79	0.8547	0.5161	0.0344		0.5337
Clobber	0.8571	0.1257	0.8682	0.8701	0.1557	0.8389	0.5619	0.1583	0.2124	0.6405	0.8575	0.722	0.7296	0.8446	0.1257	0.57	0.5337	

Figure 26: Full Benchmark Games with balance 0.7 3-grams and 0.3 game concept

Benchmark Games	Tic-Tac-Toe	Breakthrough	Yavalath	Hex	Mu Torere	Round Mercels	Chess	English Draughts	Taikyoku Shogi	Shogi	Go	Tai Shogi	Halma	L Game	Konane	Backgammon	Plakoto	Ckobber
Tic-Tac-Toe	0.8	0.8	0.1431	0.183	0.8028	0.7609	0.8678	0.8239	0.8769	0.8431	0.2832	0.874	0.828	0.8162	0.7987	0.866	0.8564	0.7619
Breakthrough	0.8	0.8728	0.8728	0.8165	0.2625	0.7415	0.5116	0.1939	0.2588	0.5567	0.7935	0.2743	0.6497	0.7856	0.2412	0.5814	0.5568	0.1877
Yavalath	0.1431	0.8728	0.0745	0.0745	0.7986	0.7571	0.88	0.8474	0.8881	0.8585	0.2891	0.8855	0.8454	0.8403	0.8125	0.8812	0.8724	0.7803
Hex	0.183	0.8165	0.0745	0.8014	0.8014	0.7808	0.876	0.837	0.8843	0.8538	0.2919	0.8864	0.8402	0.8425	0.8148	0.8762	0.8673	0.7836
Mu Torere	0.8028	0.2625	0.7986	0.8014	0.523	0.5723	0.7723	0.2369	0.3341	0.5631	0.7874	0.3429	0.6418	0.7693	0.2617	0.5575	0.5191	0.2205
Round Mercels	0.7609	0.7415	0.7571	0.7808	0.523	0.8221	0.8221	0.7371	0.8255	0.5718	0.7784	0.82	0.7639	0.8193	0.7653	0.799	0.7879	0.7343
Chess	0.8678	0.5116	0.88	0.876	0.5723	0.8221	0.8863	0.8863	0.4439	0.5994	0.8308	0.4401	0.6959	0.8019	0.5539	0.6875	0.6899	0.5582
English Draughts	0.8239	0.1939	0.8474	0.837	0.2369	0.7371	0.8255	0.4439	0.2486	0.5906	0.7941	0.2553	0.507	0.7337	0.1736	0.5381	0.5203	0.3315
Taikyoku Shogi	0.8769	0.2588	0.8881	0.8843	0.3341	0.8255	0.4439	0.2486	0.5256	0.8147	0.8415	0.0441	0.6315	0.8055	0.2892	0.5846	0.5569	0.3309
Shogi	0.8431	0.5567	0.8585	0.8538	0.5631	0.5718	0.5994	0.5906	0.5256	0.8147	0.8415	0.5125	0.5381	0.7533	0.5689	0.7225	0.7139	0.6062
Go	0.2832	0.7935	0.2891	0.2919	0.7874	0.7784	0.8308	0.7941	0.8415	0.8147	0.8285	0.7857	0.814	0.7611	0.8804	0.7981	0.7625	
Tai Shogi	0.874	0.2743	0.8855	0.8864	0.3429	0.82	0.4401	0.2553	0.0441	0.5125	0.8285	0.6202	0.7875	0.2921	0.5795	0.5526	0.3351	
Halma	0.828	0.6497	0.8454	0.8402	0.6418	0.7639	0.6959	0.507	0.6315	0.5381	0.7857	0.6202	0.6857	0.7113	0.5731	0.7151	0.7174	0.6748
L Game	0.8162	0.7856	0.8403	0.8425	0.7693	0.8193	0.8019	0.7337	0.8055	0.7533	0.814	0.7875	0.6857	0.7113	0.5582	0.7685	0.7679	0.7502
Konane	0.7987	0.2412	0.8125	0.8148	0.2617	0.7653	0.5539	0.1736	0.2892	0.5689	0.7611	0.2921	0.5731	0.7113	0.5582	0.5401	0.189	
Backgammon	0.866	0.5814	0.8812	0.8762	0.5575	0.799	0.6875	0.5381	0.5846	0.7225	0.8804	0.5795	0.7151	0.7685	0.5582	0.0515	0.5756	
Plakoto	0.8564	0.5568	0.8724	0.8673	0.5191	0.7879	0.6899	0.5703	0.5669	0.7139	0.7981	0.5526	0.7174	0.7679	0.5401	0.0515	0.5598	
Ckobber	0.7619	0.1877	0.7803	0.7836	0.2205	0.7343	0.5582	0.3315	0.3309	0.6062	0.7625	0.3351	0.6748	0.7502	0.189	0.5756	0.5598	

Figure 27: Full Knight Moves Games with balance 0.3 3-grams and 0.7 game concept

Figure 28: Full Draught Games with balance 0.3 3-grams and 0.7 game concept