1-1-2010

Natural Language Learning

Michal Andrzej Krezolek
Ryerson University

NATURAL LANGUAGE LEARNING

by

Michal Andrzej Krezolek
Bachelors of Arts and Sciences,
Computer Science and Mathematics
Felician College, Lodi, New Jersey 2006

A thesis
presented to Ryerson University

in partial fulfillment of the requirements for the degree of
Master of Science
in the Program of
Computer Science

Toronto, Ontario, Canada, 2010

Author's Declaration

I hereby declare that I am the sole author of this thesis or dissertation.

I authorize Ryerson University to lend this thesis or dissertation to other institutions or individuals for the purpose of scholarly research.

Michal Andrzej Krezolek

_____

I further authorize Ryerson University to reproduce this thesis or dissertation by photo-copying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Michal Andrzej Krezolek

_____

NATURAL LANGUAGE LEARNING

A thesis by Michal Andrzej Krezolek, Master of Science in Computer Science, 2010
Ryerson University

Abstract

This thesis is a small step towards automated learning of natural languages. With the use of a parser that incorporates machine-learning algorithms, our algorithm is able to learn meanings of words representing relations in simple sentences, that describe relative positions of two points on a 2D plane. Our SentenceLearner program can create simple sentences describing relations between two points on another 2D plane using data, collected by a statistical parser from sentences given for training, based on n-grams of five words.

In this thesis I show that association of simple relations expressed in training sentences with the positional relations of a corresponding pair of points on a 2D plane is possible without the use of any machine-learning algorithm in some circumstances.

Acknowledgements

I would like to thank my father, Andrzej Krezolek, for moral and financial support, and Dr. Eric Harley for the help with the corrections of my English sentences in this thesis and for ideas that are included in this thesis. I would like to thank also those people who made this research possible by doing their research.

Table of Contents

List of Figures

List of Equations

List of Appendices

Chapter 1: Introduction

For a long time people have wanted computers to learn facts and even language itself without human-machine interaction [1]. One such attempt was by Zellig Harris, who in 1951 published a paper on automated discovery of language structure [2]. The automated language structure discovery is done through the use of a parser. The parsers can either have knowledge of grammar encoded into them or try to discover the grammar rules automatically.

Grammar dependent parsers have trouble distinguishing relations for languages for which no grammar exists as a computer algorithm or code. For this type of parser, people would need to input the grammar rules so that learning of relations would begin. If we say in Polish language "`Kot nie jest na lewo od psa.`" we mean to say "`A/The cat is not to the left of a/the dog.`" However, when we translate the sentence word for word, the sentence will be missing some descriptive words such as "`the`" for specific object, and "`a`" for an unknown or a general object, and will not be grammatical: "`Cat not is left of dog.`" Because of that, a parser with encoded rules for English language will fail when processing sentences translated word for word from another language such as Polish or German. Some languages called head-last languages [3], like Japanese, also add another twist since the subject of the sentence is always at the very end of a sentence. In this thesis, we do not use grammar rules to design the parser, but rather let the machine learn from the situation given. This is because we might not be able to distinguish between two languages if they have the same grammar structures. In addition, our approach is not interested in the sentence structures, but meaning of words and associating those meanings through learning to the pictured situation.

Advanced versions of our algorithm will find uses in user interfaces, automatic translators for which translation is not easily achieved by current grammar or relation based translators, automated intelligent customer service, robotics, and many others.

1.1. Approach

In each of the experiments discussed in this thesis, the algorithm has access to a matrix of cells, each of which is labeled with the name of one or more points that it contains such as `A` or `K`. Associated with the matrix are sentences in a given language describing the Cartesian relationships between pairs of points. The object of the experiment is to induce the structure and meaning of each sentence based on the assumption that the sentences correctly and completely de-

scribe the relations inherent in the matrix in the sense of having exactly one sentence per relation and ordered pair of points. Our algorithm is a statistical parser with limited determination of word functions. While some word functions can be determined automatically without using complicated algorithms, some of them need to be determined by the use of the algorithms discussed below. Finally, the data about words and sentence structure gleaned by the proposed parser during training is subsequently used to build sentences that describe new matrix arrangements. Any program attempting building sentences should stop building a sentence when one of the following occurs: there are no words that could fit on the ends of the sentence that is being created, or the words on the ends of the sentence that is being created are known to be terminating words of a sentence. The program that attempts to build a sentence will be successful if the sentence is grammatical and carries a meaning that is appropriate to a given situation.

1.2. Contributions

This research is a step towards making it possible for the computer to learn how to express in a natural language the physical relations between objects in two or three dimensional space. In particular, we work with a two dimensional grid of readily identified points. We then associate the meanings of words from training sentences to the appropriate relations between two points on a given grid. During this process, we collect the statistics about the possible word combinations, and the words, in the given training sentences. We then use those statistics together with the associated meanings to describe another grid. Possible real-world applications of this research are discussed below.

Devices are needed that describe in the user's own language the relative position of objects in viewing direction of a camera. For this approach, the system developed here would need to be extended to include training that associates names such as "`cup`" with the pictures of the corresponding object. After the most common objects have been learned by the visual part of the system, the algorithms proposed in this paper would learn the relations between objects based on situations and a set of sentences. The entire system would then be able to determine what it is looking at and describe the situation with sentences expressed in user's own language. This could be helpful for a partially or totally blind person.

This research is also a small step towards machine learning of natural language. Eventually it will be possible to build robots that are able to follow commands expressed in natural language and to respond with sentences.

1.3. Overview of Thesis

In this thesis, we will demonstrate that by intelligent algorithm design, there is the possibility of creating sentences automatically. The learning of how to describe situations in the simple grid environment starts with examples that the algorithm analyzes word by word. The algorithm compares what it "knows" about the grid with natural language sentences expressing relationships of points in the grid. Later the algorithm constructs sentences based on the situation shown in a grid. The source code is included with this thesis as a separate download since the implementation of the algorithm proposed in this thesis is too long to be included even in appendixes.

In Chapter 2 we discuss the work that was done before that is related to the proposed algorithms. In Chapter 3 we propose our algorithms and we discuss how we approach the experiments. In Chapter 4 we show the results of the experiments and discuss them. In Chapter 5, we conclude the thesis and propose directions for the future work.

Chapter 2: Background and Related Work

Without the work described below, there would be no possibility for the proposed algorithm to work.

2.1. Natural Language Processing

People always wanted to ask or say something to computers in their own language, also known as natural language, and have the computer "understand" ([2], [3]). Evidence of "understanding" would be that the computers intelligently respond to such sentences perhaps by performing an assigned task [4], updating stored information [5] or discussing a situation. "Understanding" typically involves four steps. The first is lexical – breaking the sentence of input stream into tokens called words. The second step is syntactical parsing, attaching grammatical function to the words or phrases. Parsers have been developed. They can tell the structure of a given sentence ([3], [6], [7], [8], [9]). The third step is semantics – attaching meaning to words, phrases, and sentences as a whole. The fourth step, which is not addressed in this thesis, concerns semantics in context, where we decipher word meanings based on surrounding text ([3], [6], [9]). Parsers cannot create sentences by themselves.

While it might be easy for people to communicate with other people, computers always had problems with understanding the meaning of words and phrases, or differentiating their functions ([9], [10]). The ambiguities of words and phrases can confuse even the most carefully designed parser. The classic example of this is the sentence "`I saw a boy with a telescope.`" which might mean either "`A boy was in possession of a telescope when I saw him.`" or "`I used a telescope to see a boy.`" The word "`with`" has different possible functions in the given example. The information for disambiguation can be a few sentences away from the currently processed sentence or not present at all.

2.1.1. N-Grams

N-grams were first used by Chomsky for prediction probability of sentences that can occur in a text [9]. N-grams can be created by parsers or tokenizers. An n-gram has n elements of the same type, called tokens, and those tokens must be any sequential pieces of data of the same type, usually words, a user would find useful. For example the elements can be single words, or letters. In this thesis, the elements are words. The main use of n-grams is to predict the likelihood

of some data occurring before or after the data that the user provided. For example, given a trigram (w_1, w_2, w_3) and knowing what the first two words are, we may be able to select the third word based on probability, as shown in Equation 1.

$$w_3 = \underset{w_i \in V}{\operatorname{argmax}} P(w_i | w_1, w_2)$$

Equation 1. Most probable third word based on the two previous words.

In Equation 1 the V is vocalbruary, and w_1, w_2 are the first two words in the trigram. There are many practical applications for n-grams, such as machine learning, spelling and grammar correction, speech recognition ([9], [12], [13], [14]), character and text recognition ([9], [15]), machine translation ([9], [16], [17]), quality control of collected data [18], improvements in data compression searching for similar data sets, text creation, and even determination of encoding schemes or language [19]. N-grams with 2 tokens are usually called bigrams, or less commonly 2-grams. Similarly, the n-grams with 3 tokens are usually called trigrams, or less commonly 3-gram [9]. Figure 1 shows some n-grams from sentences in our experiments. N-grams often do not contain enough data to accomplish a given purpose, so researchers tend to supplement n-grams with other data, such as frequencies of occurrences of phrases or build probabilistic models for later use [9].

A)

```
is above A
is below B
A is in
A is not
not above B
```

B)

```
jest na lewo
jest na prawo
nie jest powyżej
A jest w
powyżej od B
```

Figure 1. Example of word level trigrams extracted from (A) English and (B) Polish language sentences.

2.1.2. Grammar-Based Natural Language Processing

Grammar based parsers have the knowledge of grammar structure of one of the natural languages encoded into them. These parsers usually use dictionaries of words and the possible functions which were discovered by linguists. There has been lots of work in implementing grammar based parsers, yet those parsers have downfalls such as no ability to dynamically change for new grammar ([6], [8], [9]). These parsers are language specific, that is if a parser worked well for English, it may not work at all for Polish, German, or Arabic, for example, producing errors. Consider the sentence in Polish: "`Kot nie jest na lewo od psa.`" and the word-for-word translation of that sentence to English: "`Cat not is on left of dog.`" As you

can see, the grammar forces a person to place the words in proper order for a given language. Thus, for each language a new parser has to be encoded. Even when a parser is written for a particular language, the parser must be updated to be able to parse a new grammatical structure of a language. A famous example is a work of William Shakespeare, which cannot be easily read right now since some of the grammar rules as well as word meanings have changed and/or disappeared from the English language that we speak and write today.

Some grammatical structures are very complex and encoding them correctly takes a long time. Linguists must discover the grammar rules first, and convey the rules in easy to understand detailed description. From those descriptions, programmers can then create grammar dependent parsers ([7] , [8], [9], [20]).

2.1.3. Natural Language Processing

Natural language processing has two inter-related processes. Sentences need to be parsed, and also often at the same time meaning of words needs to be determined automatically. Both of those processes benefit from each other because determination of a word meaning can indicate the word's function in a sentence, and the sentence structure can also indicate a word's meaning ([8], [9]).

Probabilistic Parser

These inconveniences are removed when a grammarless parser is used. The grammarless parser, also called a probabilistic parser, uses a probabilistic approach to learn the structure of sentences from a large body of text ([3], [6], [8], [9]). Probabilistic parser does so by attaching probabilities to syntactic trees and selecting the most likely one. This requires solving ambiguities of word functions in sentences. A probabilistic parser will not associate the meaning of the words to situations since it works only on text.

Statistical natural language processing does not have grammar rules encoded at the beginning of each run. It is designed to discover those rules automatically by learning from sentences parsed previously. This fact enables the scientists to use the same code, such as statistical CKY algorithm [9], for finding grammar of languages that do not have a language-specific parser implemented. The probabilistic parser collects data about each word and where in the sentence the word is, and what is the frequency of a phrase in the text. The parser then uses those statistics to determine what words should have what functions. The discovery process can be dif-

ficult, since many phrases such as "`in same column as`" should be treated as single words, and some phrases can be divided by a word or multiple words, such as in the sentence "`A touches B on the corner`," the phrase is "`touches on the corner`" and the word "`B`" is inserted into the phrase to satisfy grammar rules.

Also, the knowledge of where a sentence or possibly a multi word syntactic unit begins, and where it ends, can be difficult for an algorithm to determine because of characters such as periods, commas, hyphens, and quotation marks ([3], [6], [8], [9]). Often a corpora of text translated into another language provides clues.

Discovery of Word's Meanings

The process of discovery of word meanings is hampered by the ambiguities of word meanings and functions ([3], [6], [8], [9], [14], [21]), such as "`left`" can mean that something has a particular position relative to another thing, that someone has exited a room or vehicle, or that something remains. In addition, there are idioms that are specific to the geographical region of the author of the text, and those idioms dictate the word functions in sentences. The accuracy of the parsing process can be greatly improved by the use of regional and language dependent dictionaries, thesaurus, and even corpora of text translated to another language.

Many probabilistic parsers benefit greatly from machine-learning algorithms, since machine-learning algorithms discover some hidden rules of grammar and word meanings ([9], [22], [23]). This benefit is especially useful in discovering the meaning of an unknown word or phrase. However, some probabilistic parsers, such as Cooke-Kasami-Younger (CKY) algorithm, can do that without the use of a machine-learning algorithm [9].

2.2. Machine-Learning

Machine-learning is a collection of algorithms that would allow a computer to discover automatically or semi-automatically the relation between input data and expected output. A machine-learning algorithm can be used to predict a value of output based on a given input.

Machine-learning allows for learning of obvious and subtle relations among elements of given data [24]. The learned relations are called models, some of which are shown in Figure 9.D and E, and can be used to predict missing data, to group data in clusters, or to identify the class of an instance. A model has the ability to represent objects, properties of objects, and relations between objects [9]. This is especially useful in data-mining where people want to have models

of complex relations shown in easy to understand diagrams or sentence-like output of rules. These models can be updated automatically whenever there is need for such updates [25].

This model creation is useful for our approach since in the algorithm proposed in this thesis, models are created during the process of learning of word meanings and are later used in the process of sentence creation. A similar process is used in `GridLearner` [26], discussed in 2.3, but in a more rudimentary way.

Machine-learning algorithms are hard to optimize for speed while preserving the accuracy of the model creation [27]. To benchmark machine-learning algorithms, often a shallow parser is used [22]. A shallow parser is a form of probabilistic parser, which only tags a limited set of word functions as well as limited number of meanings.

2.2.1. Weka

The algorithms proposed in this thesis use machine-learning algorithms from `Weka`. `Weka` is open-source software written in Java created for data mining applications [28]. The software is a collection of machine-learning algorithms that are presented with a graphical user interface (GUI) of the program according to the functions and machine-learning algorithm types. The `Weka` GUI allows for the visualization of the data and the output of a selected machine-learning algorithm.

In our approach, we use a shallow parser with a machine-learning algorithm from `Weka` software to learn the meanings of words expressing relations, for later use in sentence creation algorithm.

2.3. GridLearner

Some of the ideas in this thesis extend relations in a program called `GridLearner` [26], that allows for teaching the computer positional relationships for two points in a grid. It does so by learning a model of a relation with the help of a machine-learning algorithm and then associating the model with a relation string that user inputs interactively using a GUI. The `GridLearner` program incorporates the `Weka` algorithms, that user is able to choose interactively, to learn relations directly from the grid. `GridLearner` does so by using the values similar to the simple relations, as discussed in 3.1.3, and the truth values entered by a user. Those values are then feed to a machine-learning algorithm, which user selects interactively, and the model is then associated with the relation string that a user enters. If the model classifies a relation string to be

applicable to the given point configuration, the string is then displayed in another part of a GUI. `GridLearner`, however, does not parse sentences describing relations. The `GridLearner` approach was investigated in our experiments and was found to be inadequate for sentence creation and similarity reasoning to the approach described in 3.3.2.

2.4. Automatic Natural Language Learning

There have been a few approaches to the problem of natural language learning without the help of humans [29]. This task is extremely difficult because of ambiguities, and grammar rules discussed above.

Algorithms that learn a natural language must resolve these ambiguities and select the most correct meaning for a word or phrase in question in order to achieve at least some understanding of the language ([3], [6], [23]). This is just a first step, since language learning also involves the creation of a sentence as a next step.

2.4.1. Sentence Creation

One of the constraints on sentence creation is not imposed by grammar itself, but by an ability of a human user to understand the relations conveyed in a sentence, and the attention span of the human user, because grammar of most if not all languages allows for creation of sentence of infinite size ([3], [6], [7]). In addition, the sentence creation process can go into an infinite loop where creation of infinite length sentences is attempted. Therefore, it is crucial to design an algorithm that avoids infinite loops, either limiting the number of words used or the overall time spent building a sentence.

The sentence creation process can be supported either by using n-grams and statistics of frequency of occurrences of phrases extracted from corpora of text, or a dictionary of words and their functions and a set of grammar rules [30]. The grammar based approach has similar drawbacks to the grammar based parsers: the grammar rules and the dictionary of words and their functions need to be adjusted for each language. They will not work for a different language without recoding the grammar rules and changing a language-specific dictionary. With statistical sentence generation, as in [30], the generated sentence quality depends on the data extracted by a probabilistic parser. In our approach, we opted to use a statistical sentence generation algorithm.

2.5. Counting by Insects

Researchers have discovered that bees can count to four [31]. The experiments involved training bees to locate food in a tunnel with movable markers and identical feeders, only one of which had a food at any given time. The number of markers as well as the number of feeders remained the same, but the location of food and/or the markers varied.

If the researchers changed the shapes of the markers, the bees trained to go past a specific number of markers ignored the shape of a marker and generalized the markers in order to count them. In addition, the bees were presented with a tunnel with the markers and feeders in different locations than they had been trained. The bees always skipped a specific number of markers without regarding the distance flown. The only problem for bees was when the researchers trained bees to go to look for food near the fifth marker. The bees were confused during the experiment phase as they checked all feeders for food.

I deduced that if bees have the ability to count embedded in their primitive brains or nervous systems, it might also be that humans have the ability to count, innate concepts for simple shapes, basic colors, and simple directional relations embedded into their brains. In the thesis, we make use of innate concepts to determine the relational position for any two chosen points in a given grid. In the case of the computer, the "innate concepts," called later simple relations, that we grant it are basic comparison and arithmetic operators that it has in the central processing unit. When relating the position of point `A` relative to point `B` in a grid, we program the computer to compare row and column positions and to set the corresponding values of simple relations. For example, if the column of point `A` is less than the column of point `B`, then we set the `HorizontalRelation` variable to the value `LEFT`. These simple relations are then used in conjunction with a machine-learning algorithm to learn relations expressed in sentences. This is explained in more detail in later sections.

Chapter 3: Methodology and Implementation

Humans have the ability to almost intuitively find subject and object of any sentence after many months of learning. By making certain assumptions in the implementation, the machine learning will determine the necessary relations from given data. These assumptions are discussed below. First we will discuss the grid, then we will describe the acceptable format of sentences, and finally we will discuss the simple relations. After discussing those, we will discuss the `SentenceLearner` approach, and why integration with `Weka` is important. Finally, we will describe the proposed algorithms and discuss the experiment set up.

Figure 2 shows the schematic of information flow in the `SentenceLearner` program as it appears after the proposed parser and the proposed sentence creation algorithms. The program takes a grid, discussed in 3.1.1, and the training sentences, discussed in 3.1.2, produces augmented n-grams and models for relation words, discussed in 3.3.3, waits for the user for either a command to generate a grid, or to load a grid from a file. The user then selects a point name, and the program uses augmented n-grams and appropriate model for a relation to generate sentence, as shown in 3.3.4, describing the selected point.

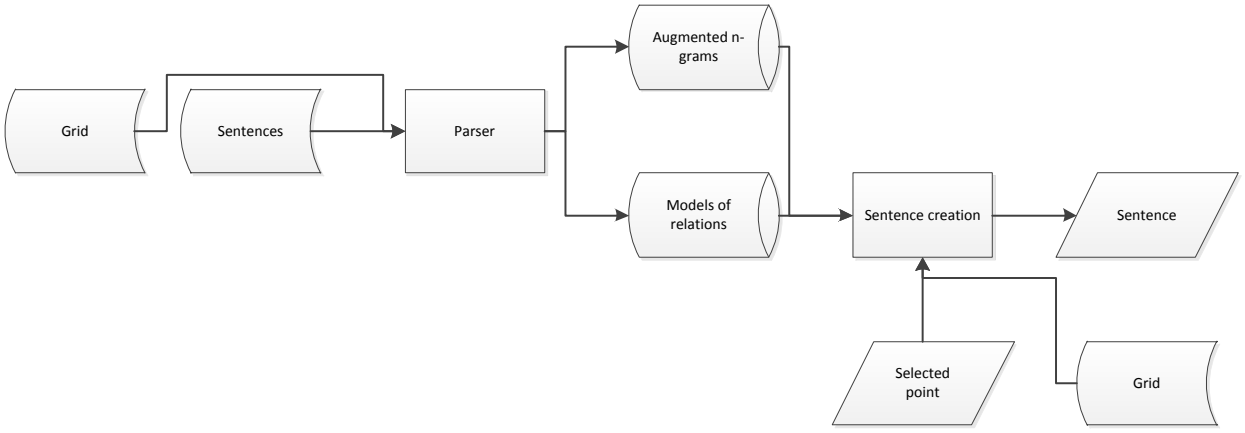


Figure 2. Schematic of information flow in the `SentenceLearner` System.

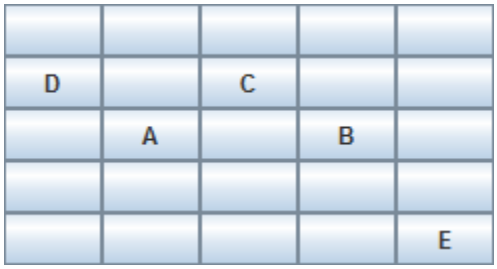
3.1. Assumptions and Necessary Data

The algorithms developed in this thesis are based on a few assumptions. We assume that each training sentence has a subject, an object, and one relation. The sentences also might include a negation of a relation. To get the learning started, the algorithms must be able to distinguish subject and object words, have a set of simple relations described below, and be able to

reference the matrix of points called the grid. Although the program we have developed is interactive, allowing continuous change of the input, the input at any given time is comprised of a grid and a set of training sentences describing positional relations of points in the grid. These components of the input are described in detail below.

3.1.1. The Grid (Input)

The grid, shown as an external data source in Figure 2, is a collection of points in a matrix that can be changed at any time by the human user. The points can be randomly distributed in the matrix. The dimensions of the matrix and the number of points can vary, but the matrix must be at least two by two in size and the number of points must be equal or greater than two.

A) 

B)

```
5,5
2,1
2,3
1,2
1,0
4,4
```

Figure 3. Example five by five grid (A) with five points: A, B, C, D, and E, and a file content (B) associated with the grid.

The grid in Figure 3.A can be set up interactively using a mouse and a graphical user interface (GUI), but it could also be specified by a simple text file shown in Figure 3.B as follows: the first number is the number of columns, then there is a comma, and then there is the number of rows. On the following lines, there are point coordinates such that the first number is the column number and the second number is the row number for the point. These numbers are separated by a comma. We start numbering rows from top to bottom and columns from left to right. The starting number for rows and columns is 0. Each point is described on a separate line. Thus given the grid input, the computer has the row and column coordinates of each point in the grid. For example, the point C in the grid in Figure 3.A is at row 1 and column 2 if we number rows from top to bottom and the columns from left to right starting at 0. We will refer to the row and column coordinates as Cartesian coordinates of a point, so a point C has the Cartesian coordinates (1, 2).

3.1.2. Training Sentences (Input)

Each sentence in our experimental system states a positional relationship or the negation of positional relationship between two points, and the sentences are context-independent. That is the sentences do not depend for meaning on the other sentences. We will call the points the subject and the object where the subject is the main topic of a sentence and the object is the point needed to describe the position of the subject, for example in the English sentence "`A is above B`," which applies to the grid in Figure 4, we identify `A` as a subject, and `B` as the object. In a head-last language [3], the sentence might take the form "`Is above B A`," as shown in Figure 5, and again, `A` is the subject and `B` is the object. The detection of the subject and object could be difficult if we allow for either structure, and therefore, this complication will be a part of future work. Here we assume the language is head-first, as in Figure 4.

A) `A is above B.`

B) 

Figure 4. A) English sentence describing where the point A is in relation to point B. B) The grid to which the sentence applies.

A) `Is above B A.`

B) 

Figure 5. A) Example of a head-last sentence structure. B) The grid to which the sentence structure applies.

We assume each sentence has a verb, which is either a "`verb of being`" or a "`relational verb`." A "`verb of being`" is a verb that introduces the predicate relation. For example, in the English sentence shown in Figure 4, the "`verb of being`" is the word "`is`," which introduces the relation "`above`." In the English sentence "`A touches B`," shown in Figure 6, the verb is a "`relational verb`," "`touches`," which in itself states the positional relation between subject and object.

A) `A touches B.`

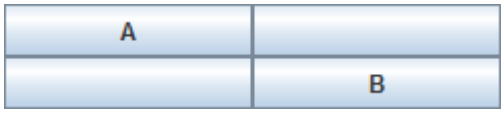
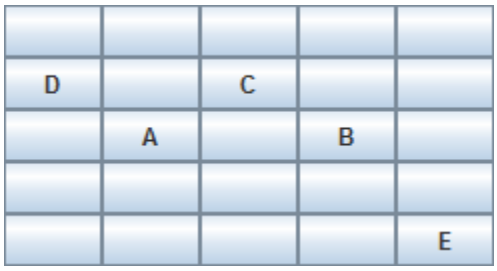
B) 

Figure 6. A) Example of a "`relational verb`" used in English sentence. B) The grid to which the sentence applies.

There are languages, such as ancient Greek, Chinese, Japanese, and Persian [3], where the separation between words is not simply white spaces as in many modern languages. The de-

tection of word boundaries in such languages is a difficult task, even if the grammar rules and a dictionary for those languages exist. This word boundary detection will therefore be part of the future work, and the words in the input to the proposed algorithm will have to be separated by white space characters.

A)



B)

```
A is not above B.
B is not above C.
C is in same row as D.
D is above E.
E is not above A.
A is not below B.
B is below C.
C is not below D.
D is not below E.
E is below A.
```

Figure 7. A) Grid. B) Some sentences describing the grid.

The number of training sentences is typically $n(n-1)m$, where there are n points and m relations. For example, if the training sentences for the grid in Figure 7.A cover the relations "`above`" and "`below`," there would be $5(5-1)2 = 40$ sentences, some of which are shown in Figure 7.B above. For the proposed parser, there must be $m \geq 2$ relations.

The training sentences, shown as an external data source in Figure 2, include those sentences that have logical consequences of the previous sentences since we assume the proposed algorithms have no previous knowledge of the language and word meanings. Knowledge is either associated with or learned, with the help of a machine learning algorithm, from relations between ordered pairs of points. This is discussed further in this chapter.

3.1.3. Simple Relations

We assume the human brain has basic directional relations that are associated with words or symbols through learning process. We also assume the human brain has the ability to count embedded into it from birth, and then through learning, the brain associates words or symbols for numbers with the numbers themselves. We define the ability to count and the basic directional relations as the simple relations.

Because the computer does not have those simple relations, we define `RelationalData` objects (described in this paragraph and in Figure 8 below) to represent the simple relations, and call the values of the fields of `RelationalData` objects as simple relations. Consider two points,

A and B, in a grid, such that the coordinates for A are (a_r, a_c) and the coordinates for B are (b_r, b_c). The number of rows is r_g, and the number of columns is c_g. The SentenceLearner algorithm receives these coordinates as input and constructs a RelationalData object for A relative to B as follows. There are seven fields to RelationalData object. The possible values for each field are shown in Figure 8 below. The reason for including the grid dimensions is that the distance relation from training sentences relies on distance measured between points and the dimensions of a grid, as shown in Equation 2 and Equation 3 below. These two equations define near and far. The value in the denominator is chosen arbitrarily.

Field	Value	When
HorizontalRelation	LEFT	$a_c < b_c$
	RIGHT	$a_c > b_c$
	SAME_COLUMN	$a_c = b_c$
HorizontalDistance	n	$\lvert a_c - b_c \rvert = n$
VerticalRelation	ABOVE	$a_r < b_r$
	BELOW	$a_r > b_r$
	SAME_ROW	$a_r = b_r$
VerticalDistance	m	$\lvert a_r - b_r \rvert = m$
IsTouching	TRUE	$\lvert a_c - b_c \rvert \leq 1$ and $\lvert a_r - b_r \rvert \leq 1$
	FALSE	$\lvert a_c - b_c \rvert > 1$ or $\lvert a_r - b_r \rvert > 1$
HorizontalGridDimention	c_g	
VerticalGridDimention	r_g	

Figure 8. Fields of RelationalData object and possible values for points A and B.

$$\sqrt{\text{HorizontalDistance}(A,B)^2 + \text{VerticalDistance}(A,B)^2} \leq \frac{\sqrt{\text{HorizontalGridDimention}^2 + \text{VerticalGridDimention}^2}}{5}$$

Equation 2. The inequality defining the relation "near to" between a pair of points A and B. If the inequality is true, the relation applies to the pair of points A and B.

The simple relations' values are: `LEFT`, `RIGHT`, `SAME_COLUMN`, `ABOVE`, `BELOW`, `SAME_ROW`, `IS_TOUCHING`, as well as non-negative numeric values, such as `0`, `1`, `2`. Those values are then used as attributes together with `FALSE` or `TRUE` class value to train a machine-learning algorithm. From there, the model is then associated with words or phrases that express ideas understandable to the user.

$$\sqrt{\text{HorizontalDistance}(A,B)^2 + \text{VerticalDistance}(A,B)^2} \geq \frac{4\sqrt{\text{HorizontalGridDimention}^2 + \text{VerticalGridDimention}^2}}{5}$$

Equation 3. The inequality defining the relation "`far from`" between a pair of points `A` and `B`. If the inequality is true, the relation applies to the pair of points `A` and `B`.

Horizontal relations between two points in the grid as observed by the algorithm are `LEFT`, `RIGHT`, and `SAME_COLUMN`. Vertical relation between two points in the grid as observed by the algorithm are `ABOVE`, `BELOW`, and `SAME_ROW`. The simple relation `IS_TOUCHING` indicates if two points on a grid are touching each other on the sides, corners, or when the points occupy the same cell in the grid. The numerical simple relations are used for determining distance measured from one point to a second point or to determine the size of the grid.

Our algorithms have a simple way to determine the truth values for relations corresponding to simple relations based on data contained in a given grid. The algorithm just compares Cartesian coordinates of the points and produces the truth values for the simple relations. These relations are meant to simulate the human brain function on how the objects are located in the field of vision as well as touch senses. These simple relations are used to determine the positional relationships in a given grid, and must be associated with a word or phrase to describe the positional relation between a pair of points in a grid. For this purpose, a machine-learning algorithm will determine whether a relation expressed in words applies to the observations from the grid.

3.1.4. Subject and Object Words in a Sentence

We assume the first point name is the subject and the second point name is the object. The point names must fall in the range of capital letters from `A` to `Z`.

3.2. SentenceLearner

`SentenceLearner` is an algorithm able to learn the meaning of words by parsing the sentences and extracting information from sentences associated with a given matrix of points. It is

also an algorithm to build sentences based on the knowledge extracted during the parsing process and comparison of sentences with the situations between an ordered pair of points. The parts of the SentenceLearner program in the final form that appears in this thesis are described in sub-sections 3.3.3 to 3.3.4.

3.2.1. Integration with Weka

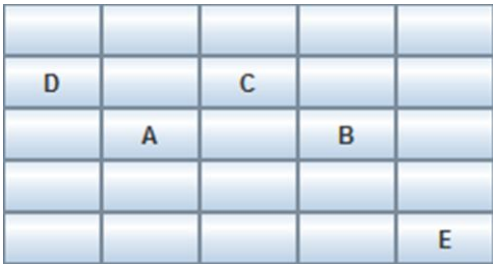
The algorithms of the SentenceLearner program are integrated smoothly with the Java classes of the Weka software, so that machine-learning algorithms can be called directly. Those machine-learning algorithms can be selected interactively using GUI of the SentenceLearner program. In contrast, if the Weka software were used externally and interactively, one would manually set up input files called ARFF files (Attribute-Relation File Format) which hold the definitions of the attributes and their values and the class value for each training instance [28]. Although the SentenceLearner program does not need to use ARFF files, it can. Figure 9.A shows an ARFF file for building a simple model for the relation "is above," relational data for the grid shown in Figure 9.B, and sentences that convey the "above" relation are shown in Figure 9.C.

The ARFF file starts with the header @relation after which the name of the relation is introduced. In case of the name of relation containing more than one word separated by a white space character, quotation marks must be used around the name. The next lines starting with @attribute introduce attributes. Each attribute name is then followed by curly brackets in which the possible values of the attributes are separated by commas, or by the key word numeric which means the allowed values for the attribute are real numbers. Because the class is defined later by the user of Weka software as one of the attributes, the ARFF file format does not have a special key word to mark an attribute as a class. The @data key word tells that the following lines are the actual values, separated by commas, for the attributes in the order that they were declared. For example, the line LEFT,2,SAME_ROW,0,false,5,5,false, which corresponds to the sentence "A is not above B." can be read from left as the point A is left of point B with 2 spaces away in the horizontal direction, the point A is in same row as point B with no vertical separation, the points do not touch, the grid is 5x5 in size, and the class value of "is above" is false. Missing value for an attribute is represented by a question mark, but for this feature we do not have use in our algorithm as everything is determined from a grid and assigned class value. The values for each of these attributes are defined in section 3.1.3.

A)

```
@relation "is above"
@attribute horizontalConceptAttribute {LEFT,RIGHT,SAME_COLUMN}
@attribute horizontalDistanceAttribute numeric
@attribute verticalConceptAttribute {ABOVE,BELOW,SAME_ROW}
@attribute verticalDistanceAttribute numeric
@attribute isTouchingAttribute {true,false}
@attribute horizontalGridDimentionAttribute numeric
@attribute verticalGridDimentionAttribute numeric
@attribute internalPhraseOrKeyWordIDAttribute numeric
@attribute classAttribute {true,false}
@data
LEFT,2,SAME_ROW,0,false,5,5,false
LEFT,1,BELOW,1,true,5,5,false
RIGHT,1,BELOW,1,true,5,5,false
RIGHT,3,ABOVE,2,false,5,5,true
RIGHT,1,SAME_ROW,0,false,5,5,false
RIGHT,1,BELOW,1,true,5,5,false
RIGHT,3,BELOW,1,false,5,5,false
LEFT,1,ABOVE,2,false,5,5,true
RIGHT,1,ABOVE,1,true,5,5,true
LEFT,1,ABOVE,1,true,5,5,true
RIGHT,2,SAME_ROW,0,false,5,5,false
LEFT,2,ABOVE,2,false,5,5,true
LEFT,1,ABOVE,1,true,5,5,true
LEFT,3,ABOVE,1,false,5,5,true
LEFT,2, SAME_ROW,0,false,5,5,false
LEFT,4,ABOVE,3,false,5,5,true
RIGHT,3,BELOW,2,false,5,5,false
RIGHT,1,BELOW,2,false,5,5,false
RIGHT,2,BELOW,3,false,5,5,false
RIGHT,4,BELOW,3,false,5,5,false
```

B)



C)

```
A is not above B.
A is not above C.
A is not above D.
A is above E.
B is not above A.
B is not above C.
B is not above D.
B is above E.
C is above A.
C is above B.
C is not above D.
C is above E.
D is above A.
D is above B.
D is not above C.
D is above E.
E is not above A.
E is not above B.
E is not above C.
E is not above D.
```

```
------------------
J48 pruned tree
------------------

verticalConceptAttribute = ABOVE: true (8.0)
verticalConceptAttribute = BELOW: false (8.0)
verticalConceptAttribute = SAME_ROW: false (4.0)

Number of Leaves  :         3

Size of the tree :         4
```

D)

```
------------------

NNGE classifier

Rules generated :
        class    false    IF    :    horizontalConceptAttribute    in    {LEFT,RIGHT}    ^
0.0<=horizontalDistanceAttribute<=3.0  ^  verticalConceptAttribute  in  {BELOW,SAME_ROW}  ^
1.0<=verticalDistanceAttribute<=4.0    ^    isTouchingAttribute    in    {false,true}    ^
horizontalGridDimentionAttribute=5.0 ^ verticalGridDimentionAttribute=5.0  (12)
        class    true    IF    :    horizontalConceptAttribute    in    {LEFT,RIGHT}    ^
1.0<=horizontalDistanceAttribute<=3.0    ^    verticalConceptAttribute    in    {ABOVE}    ^
1.0<=verticalDistanceAttribute<=4.0    ^    isTouchingAttribute    in    {false,true}    ^
horizontalGridDimentionAttribute=5.0 ^ verticalGridDimentionAttribute=5.0  (8)

Stat :
        class false : 1 exemplar(s) including 1 Hyperrectangle(s) and 0 Single(s).
        class true : 1 exemplar(s) including 1 Hyperrectangle(s) and 0 Single(s).

        Total : 2 exemplars(s) including 2 Hyperrectangle(s) and 0 Single(s).

        Feature  weights  :  [0.2958073480446817  0.17095059445466854  0.9709505944546686
0.07594386796016228 0.012750423385226917 0.0 0.0]
```

E)

Figure 9. Example 5x5 grid (B) from which the `ARFF` file (A) was generated. C) Sentences that describe the relation "`above`." D) Model[1] created by `J48` algorithm. E) Model[1] created by `Nearest Neighbour` algorithm. Models in `SentenceLearner` program are stored in an internal database, as shown in Figure 2.

The machine-learning algorithms also generate models that can be used to predict class values based on the values of simple relations. Figure 9.D and E show example models generated by the chosen machine-learning algorithms.

3.3. Algorithm Design

The general design of the algorithms in this thesis is described in this section. Since the program `SentenceLearner` program failed expectations in the initial experiments, the program was redesigned for improved results. In this section we describe algorithm designs at each stage

[1] In addition to a model, `Weka` also outputs statistics about class details, results, and confusion matrix.
[2] "`X --> Y`" means position of `X` relative to `Y`.
[3] Some of the words listed are just different version of the other words, as Polish language requires the form of the word to be

of the design process and in Chapter 4 we conduct experiments for each stage of the design process to prove the proposed algorithms work as intended.

3.3.1. Association between Simple Relations and Sentences

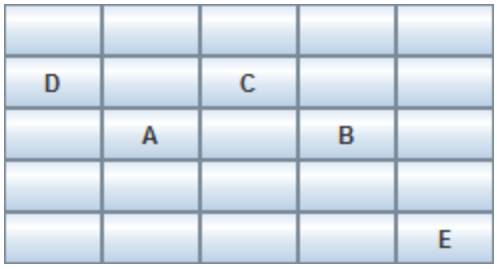
The association between simple relations and some sentences can be done easily without the use of any machine-learning algorithm as discussed below. We need a structure called `SentenceData`, which will hold everything of interest from a given sentence, that is the unaltered sentence text, the relation from sentence, and the point names. The `RelationText` element in `SentenceData` holds the sentence text that remains after removing the point names. Each sentence is processed into a corresponding `SentenceData` object. This is illustrated in Figure 10 below.

Sentence Text	RelationText	Left Point Name	Right Point Name
A is left of B.	is left of	A	B

Figure 10. Tabular representation of an example `SentenceData` data structure.

The process of association of sentences to simple relations can now begin. First, the `SentenceData` objects need to be sorted according to `RelationText`. For each different `RelationText`, a set that will hold `SentenceData` objects is created. Each `SentenceData` object is then moved to the appropriate set that corresponds to `RelationText`. For example, if we have the sentences: "`A is left of B`," "`A is right of C`," "`A is not right of B`," "`A is not left of C`," "`B is left of C`," and "`B is not right of C`," then we have the sets: one for the `RelationText` "`is left of`" holding object representing sentences {"`A is left of B.`", "`B is left of C.`"}, another for `RelationText` "`is not right of`" {"`A is not right of B.`", "`B is not right of C.`"}, yet another for `RelationText` "`is not left of`" {"`A is not left of C.`"}, and a similar set for `RelationText` "`is right of`" holding sentence {"`A is right of C.`"}. Even if the sentences are seemingly the logical consequences of each other, they need to be included, since we assume the program has no prior knowledge of the languages, and if the sentences given for training include those expressing "`in same column`," the logic becomes more complicated.

A)



B)

```
A is not above B.
B is not above C.
C is not above D.
D is above E.
E is not above A.
A is not below B.
B is below C.
C is not below D.
D is not below E.
E is below A.
```

C)

Line number	Sentence or state	Value for Boolean variable associated with `LEFT`	Value for Boolean variable associated with `NOT LEFT`
1	Initial state for "`is not above`"	TRUE	TRUE
2	`A is not above B.`	TRUE	FALSE
3	`B is not above C.`	FALSE	FALSE
4	Final state for "`is not above`"	FALSE	FALSE
1	Initial state for "`is above`"	TRUE	TRUE
2	`D is above E.`	TRUE	FALSE
3	Final state for "`is above`"	TRUE	FALSE
1	Initial state for "`is not below`"	TRUE	TRUE
2	`A is not below B.`	TRUE	FALSE
3	`C is not below D.`	FALSE	FALSE
4	Final state for "`is not below`"	FALSE	FALSE
1	Initial state for "is below"	TRUE	TRUE
2	B is below C.	FALSE	TRUE
3	E is below A.	FALSE	TRUE
4	Final state for "is below"	FALSE	TRUE

Figure 11. Example grid (A) with associated sentences (B) and the state of association of `RelationText` to simple relations (C). See text for detailed explanation.

After partitioning `SentenceData` objects into sets, we can check for consistency of each set corresponding to a `RelationText` with each non-numeric simple relation and its negation. To do so we use a Boolean variable for each non-numeric simple relation and their negations. We start the consistency check by marking the chosen `RelationText` as consistent with all simple

relations and their negations. For each SentenceData object in a selected partition, we then check if the values from the corresponding RelationalData object have at least one non-numeric simple relation value in common. If they do not match, we change appropriate Boolean variable's value to FALSE for the simple relation whose value did not match. We repeat the checking of the values for each SentenceData object from the selected partition as shown in example from Figure 11. If the value of a Boolean variable stays TRUE, there is no conflict, and we can say the set corresponding to a RelationText is at 100% consistent with a chosen simple relation and associate the RelationText with that simple relation. We do this consistency check to distinguish between the meanings of the RelationTexts and whether we can say the same thing using different words.

As you can see from the example in Figure 11 above, the sentences must describe the grid completely to avoid false associations, such as "is above" with LEFT and "is below" with negation of LEFT. We also can stop checking a partition the moment all values of Boolean variables become FALSE. For example, on line 1 of "is not above" we start with an assumed value TRUE for LEFT and NOT LEFT. On line 2, the value for NOT LEFT changes to FALSE because the point A is left of point B. Similarly, on line 3, the value for LEFT changes to FALSE because the point B is not left of point C. The final state of "is not above," shown on line 4, means no association of "is not above" with LEFT or NOT LEFT has been made.

Conversely, the initial state for "is above," shown on line 1, starts with the assumed value TRUE for LEFT and NOT LEFT. The line 2 for the state "is above" has the value for NOT LEFT changed to FALSE since the point D is left of the point E on the grid. Line 3 shows the final state of "is above." Since there were no more sentences for "is above" and the value for LEFT remained TRUE, association of the phrase "is above" to LEFT is made.

```
Algorithm Simple Association
Inputs:  Grid, Sentences describing the grid
Outputs: Truth values and associated RelationText values
Steps:
   1. Create SentenceData objects, and partition them according to
      RelationText values.
   2. For each partition.
         a. Sort SentenceData objects according to subject and object.
         b. Set values of Boolean variables corresponding to non-numeric
            simple relations and their negations to TRUE.
         c. For all ordered point name combinations.
```

i. If SentenceData object exist, proceed to step 2.c.i.1. Otherwise, if tere are SentenceData objects remaining in the selected partition, skip all steps before step 2.c.i.2.b.
    1. Create RelationalData object for the selected SentenceData object.
    2. For each value of non-numeric simple relation and its negation.
        a. If the value of a corresponding simple relation from the RelationalData object differs from the selected simple relation value, change the value of the corresponding Boolean variable to FALSE.
        b. Proceed to step 2.c.i.2 if there are remaining values for simple relations and their negations. Otherwise, output the RelationText values for the simple relations or their negations whose Boolean variables remained TRUE.

Figure 12. Simple association algorithm.

3.3.2. Association of Machine-Learning Model with RelationText

The algorithm discussed in the previous subsection had trouble distinguishing the concepts "touches" and "touches on the corner" since the second concept is a special case of the first one. This is because the mentioned relation is associated with horizontal distance and vertical distance simple relations which are numeric and both have to be equal to 1. The numeric relations and the combinations of relations would need to be encoded into the algorithm from 3.3.1, thus it is not possible to pre-code all possible relations, for example, that require using natural numbers. Instead, we can use an existing machine-learning algorithm to induce a model that represents the positional relationship described by a word or some phrase. For example, if the positional phrase is "is left of," then the class name is also "is left of." For each sentence of the type "X is left of Y," the algorithm receives as input the numeric and non-numeric values of simple relations shown in Figure 8 along with the class value TRUE. For each pair of points X, Y for which there is not a sentence "X is left of Y," the algorithm receives the simple relation values appropriate for the relative positions of X and Y, along with the class value FALSE. The algorithm discussed here is supposed to learn all the relations the previous algorithm failed to distinguish through association.

This approach is similar to the approach in 3.3.1, but it has a slight advantage since now all complex relations can be considered. The same algorithm to partition `SentenceData` objects into sets mentioned in 3.3.1 is used.

To build a model of a class, whose name is `RelationText`, using a machine-learning algorithm we need to either associate `SentenceData` objects with observed simple relations as mentioned in 3.3.1 or build `RelationalData` objects that store all the simple relation values for the appropriate ordered point name combinations as described in 3.1.3.

We select one of the partitions of the set of training sentences and define a class with the name of that `RelationText`. Now we move the `RelationalData` objects corresponding to the `SentenceData` objects in the selected partition to a set which is associated with `TRUE` value of the class, and move the remaining `RelationalData` objects to a set which is associated with `FALSE` value of the class as we need a counter-example for the machine-learning to work. Otherwise, if there would not be any counter-example, the model produced during training would always choose `TRUE` as the result of later classification. We feed the values of elements of the `RelationalData` objects from both sets together with appropriate class value to a machine-learning algorithm in order to get a model, as in example shown in Figure 9.D and E, corresponding to the `RelationText` associated with the selected group.

Now we can check if the model is 100% consistent with the data. To do so, we do the following with all partitions: we get the model and feed through it the `RelationalData` objects corresponding to a single partition. A partition of `SentenceData` objects is said to be 100% consistent with the model created by the machine-learning algorithm if all the `RelationalData` objects from the tested partition are classified by the model as `TRUE`. Only then an association between sets of `SentenceData` objects that were deemed consistent with the model to the model can be made. The model can then be stored in memory and must not be used again in this algorithm as to not affect the other `RelationText` consistency tests and associations as the process repeats for the remaining partitions associated with respective `RelationTexts`.

A)

Horizontal Relation	Horizontal Distance	Vertical Relation	Vertical Distance	Is_Touching Relation	Horizontal Grid Dimension	Vertical Grid Dimension
LEFT	1	ABOVE	2	FALSE	5	10

B)

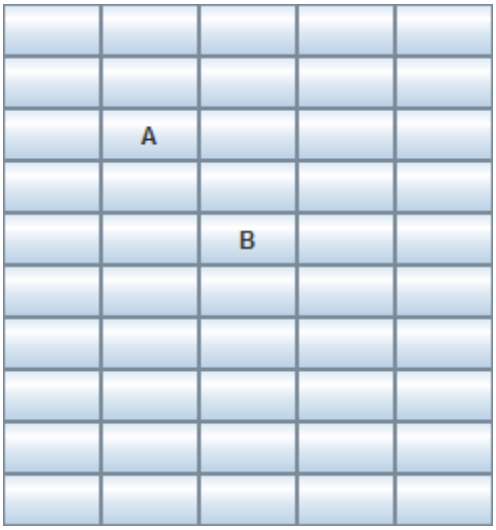


Figure 13. A tabular representation of an example `RelationalData` (A) where the subject, `A`, is located according to object, `B`, as shown in (B). Points `A` and `B` can be in any place in the above grid as long as they are placed in the proper distances from each other and on the proper sides of each other.

```
Algorithm Model Association
Inputs:  Grid, Sentences describing the grid
Outputs: Models for relations and associated RelationText values
Steps:
   1. Create SentenceData objects, and partition them according to
      RelationText values.
   2. For each partition.
         a. Sort SentenceData objects according to subject and object.
         b. For all ordered point name combinations.
               i. Create RelationalData object.
              ii. If SentenceData object exist, proceed to step 2.b.ii.1.
                  Otherwise skip all steps before step 2.b.iii.
                     1. Move the RelationalData object to the set associated
                        with class value TRUE.
                     2. If there are remaining ordered point name
                        combinations, proceed to step 2.b. Otherwise skip all
                        steps before step 2.c.
             iii. Move RelationalData object to the set associated with class
                  value FALSE.
              iv. If there are remaining ordered point name combinations,
                  proceed to step 2.b. Otherwise skip all steps before step
                  2.c.
```

```
c. Build machine-learning model using the values of RelationalData
   objects and appropriate class values.
d. For each partition of SentenceData objects.
     i.  For each SentenceData object.
           1. Create RelationalData object.
           2. Classify the RelationalData object using the model.
           3. If the RelationalData object is classified as FALSE,
              proceed to step 2.d.
           4. If there are SentenceData objects remaining, proceed
              to step 2.d.i. Otherwise, remember RelationText and
              associate it with the model.
           5. If there are more SentenceData partitions remaining
              after step 2.d, proceed to step 2.d. Otherwise
              proceed to step 2.e.
e. If there are more partitions remaining after step 2, proceed to
   step 2.
```

Figure 14. Model association algorithm.

3.3.3. Parser

We developed a simplistic parser algorithm defined below. Although complete parsers for various languages are available in the literature ([3], [6], [7] , [8], [9], [20], [21], [23]), we chose not to use them because they cannot associate a meaning of a word from sentences to a situation between objects in a given picture, or points in a given grid. The parser takes sentences as input together with the associated grid. The parser splits the sentences into words, determines subject, object, verb, negation, and relation in a given set of sentences. It does so by a statistical approach which extracts word combinations, and counts how many times they were used. This is discussed in detail in the next subsection. Each of the augmented n-grams discussed below needs to be stored in the memory for the sentence creation algorithm which is discussed further below. The reason for developing a new simplistic parser is the other parser algorithms cannot learn from a grid and a set of training sentences. The other simplistic parsers are usually context specific and require a large corpus of text for training, whereas the proposed parser learns from context-independent sentences. There are a few key differences in the two versions of the proposed parser algorithm. Those differences are discussed below.

Augmented N-Gram Based Approach

Each augmented n-gram data structure in the proposed algorithm consists of one or more word level n-grams, numbers of occurrences of each word of the n-gram, and the types of the words in the n-gram. Word types are defined below.

Our augmented n-gram has either 3 or 5 spaces for words. The sliding window spaces which are empty are marked as NOTHING. This means, if we have a sliding window, pictured in Figure 15 below, we simply include an empty space instead of a word from previous or next sentence. The main word is the word in the middle spot of the sliding window, and the sliding window must contain a word in the middle space. The sliding window is defined as a limit of the number of tokens from the input that can be processed at any given time even though the input might have more than n tokens. For example, if we have a sliding window of size n we can select up to n tokens from the input of size m.

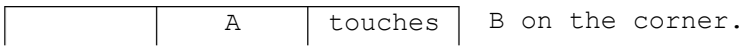


Figure 15. An example of three word sliding window in which the main word in augmented n-gram will be A.

The main word is classified as one of the following types: RELATION, RELATION_VERB, NEGATION, POINT_NAME, VERB, or UNKNOWN. The UNKNOWN type is assigned by default to all words before they are classified as other types in the parser.

As described further in this subsection, the type RELATION is assigned to words, such as "left," that are determined to be relations. The type RELATION_VERB is assigned to words, such as "touches," that are both relations and verbs. The type NEGATION is assigned to words, such as "not," that negate the relation words. The type VERB is assigned to words, such as "is," which are determined to be verbs as described further in this subsection.

A special case of word type is POINT_NAME, since the parser just discards the word meaning the point name, such as "B," and just assigns a placeholder for the actual point name. Later the placeholder is replaced with a given point name when building sentences as described further in this subsection.

The main word also needs to have the number of uses associated with it. The number of uses is the number of times a word occurs in the set of sentences given for training.

There might or might not be words preceding or following the main word. The allowed word types for the words adjacent to the main word in the same sentence are NOTHING, POINT_NAME, or UNKNOWN, and in case of NOTHING, there is no word. We do not need to mark the adjacent words in the augmented n-gram with the remaining types since we can extract the information about those words from the augmented n-grams whose main word equals the adjacent word in question.

The words surrounding the main word also have to have the number of uses associated with them. For example, in Figure 16, the number in column 4 is the number of co-occurrences of "is" and "left" in the training sentences that the program has seen so far. The number in column 2 is the number of co-occurrences of POINT_NAME, "is," and "left." All point names are replaced by a placeholder called POINT_NAME, so they are all treated the same. Conversely, the same applies for the words on the right side of main word in an augmented n-gram. These numbers are not saved by the first parser algorithm.

We give the words that are one position away from the main words the name inner words, and the words that are two spots away from the main word are outer words.

Previous Outer Word	# of Uses	Previous Inner Word	# of Uses	Main Word	Main Word Type	# of Uses	Following Inner Word	# of Uses	Following Outer Word	# of Uses
POINT_NAME	10	is	10	left	NEGATION	10	of	10	POINT_NAME	10

Figure 16. Example of an augmented n-gram whose main word was incorrectly classified as NEGATION.

We define two augmented n-grams to be compatible if they share the same main word or if the type of the main word in both of the augmented n-grams is POINT_NAME. Each compatible augmented n-gram pair can be merged component-wise to form a new augmented n-gram. The merging does not affect the main word type if and only if both augmented n-grams have the same main word type. Otherwise, the main word type is set to the word type of the augmented n-gram whose number of uses of main word is greatest in both augmented n-grams that were merged. For example, if we have augmented n-grams pictured in Figure 16 and Figure 17 below, we have the word type as shown in Figure 18 below. The reason for merging of the augmented n-grams is to avoid repeating the augmented n-grams for each identical augmented n-gram that is extracted from a sentence or assigning counters for each non-repeating augmented n-gram. Examples of such merged augmented n-grams are in Figure 18 and Figure 19.

We define an augmented n-gram to be compatible with a word if the main word from the augmented n-gram is the same as the word. For example, if we have augmented n-gram shown in Figure 16 above, the augmented n-gram is compatible with the word "left".

All the augmented n-grams are then stored in internal database, as shown in Figure 2.

Previous Outer Word	# of Uses	Previous Inner Word	# of Uses	Main Word	Main Word Type	# of Uses	Following Inner Word	# of Uses	Following Outer Word	# of Uses
is	20	not	20	left	RELATION	20	of	20	POINT_NAME	20

Figure 17. Example of an augmented n-gram whose main word was classified as RELATION**.**

Previous Outer Word	# of Uses	Previous Inner Word	# of Uses	Main Word	Main Word Type	# of Uses	Following Inner Word	# of Uses	Following Outer Word	# of Uses
POINT_NAME	10	is	10	left	RELATION	30	of	30	POINT_NAME	30
is	20	not	20							

Figure 18. A tabular representation of an example five word augmented n-gram collected for the word "left**" that was classified in this example as** RELATION**. There were 30 sentences containing the word "**left**." In this example there are 10 sentences saying "**X is left of Y**" and 20 sentences saying "**X is not left of Y**," where** X **and** Y **are distinct point name pairs.**

Previous Outer Word	# of Uses	Previous Inner Word	# of Uses	Main Word	Main Word Type	# of Uses	Follow-ing Inner Word	# of Uses	Following Outer Word	# of Uses
NOTHING	85	POINT_NAME	85	is	VERB	85	left	30	of	30
							not	40	left	20
									right	5
									above	5
							above	5	POINT_NAME	5
							right	10	of	10

Figure 19. A tabular representation of an example five word augmented n-gram collected for the word "is**" that was classified in this example as** VERB**. Note the words following the word "**is**" as shown in columns Following Inner Word and the words in Following Outer Word following the words in Following Inner Words column.**

Determination of Subject and Object

Determination of subject and object is relatively easy since we assume the point names have a strict format, are subject or object, and the subject comes near or on the beginning of a sentence whereas the object comes near or at the end of a sentence. The point names are one letter words, made with capital letters from A to Z. Since of the assumption in 3.1.4, we do not need to mark the subject and object words as subject and object, respectively. The algorithm, described in this section below, is given subject and object point names. This greatly simplifies the parsing of sentences but will be changed in future work as discussed in 5.1 to support more complex sentences, such as head-last sentences.

Determination of Verb of Being

The verb of being can be determined by using the ratio of the number of occurrences of a selected word in the bag of words obtained by tokenizing all the sentences given for training to the number of sentences that were given for training. Given the structure of sentences assumed in our experiments (see section 3.1.2), this ratio for verb of being must be less than or equal to 1, since we assume at most one verb of being per sentence. This is because training sentences containing exactly three words consist of subject, object, and relation verb words. The ratio is also greater than certain value, which was found in the experimental phrase. Because we do not want to falsely classify a negation as a verb of being, an educated guess for the lowest value of the ratio should be at least 0.5.

Determination of Relations and Negations

The algorithm needs to identify correctly words or phrases that represent relations and the negations in order to later build the sentences that accurately describe a situation. The template approach, which extracts predetermined key words or phrases and reacts accordingly by executing associated rules [7], would also work but it might produce more errors and will not be as flexible in learning the relations and negations accurately. This is because for each key word or phrase we would need to encode set of rules, and if a user enters a phrase or a key word that does not have associated rule, the template approach would ignore the input. We also avoid the template approach since we assume the computer should not have any prior knowledge about a language, and using the template approach would indirectly imply the knowledge about a language.

In our experiments we have had two versions of the algorithm for determining the relations and negations. Since some of the relation words were not found using the simpler approach, we had to improve the algorithm to detect those relations. The results are discussed in detail in the experiments section. The algorithm for determination of relations and negations is executed after finding the verb of being, subject, and object words. The input is the grid and sentences describing it. The output is classification of words and the models associated with `RELATION` words. These models are set of rules that would tell whether a particular relation applies to a chosen relation. Those models are later used in the sentence creation algorithm.

The First Algorithm

When we have identified all the subjects, objects, and possible verbs of being as discussed previously in this subsection, the next step is to identify relations and negations. Relations in three word sentences, such as "`A touches B`," in our experiment are easily spotted, since the relation is also a verb, and is the word that is left after deletion of subject and object words.

Relations and negations in sentences longer than three words have to be determined using the parser that incorporates a machine-learning algorithm. First we tokenize the training sentences and create augmented n-grams for each of them. We merge all the compatible extracted augmented n-grams and keep them in a computer memory.

In this subsection, we define a word to be the main word and a word type to be the main word type from the same augmented n-gram.

For each ordered pair of augmented n-grams, we select two different main words initially classified as `UNKNOWN` from the list of augmented n-grams. Then we assume one word is a `NEGATION`, and the other word is a `RELATION`, and because we do not know what types the words really are, we are going to test the assumption later in the parser algorithm. For example, we have words "`left`" and "`not`" we can assume that "`left`" is a `RELATION` and "`not`" is a `NEGATION`. Since the program has no prior knowledge about a language in the initial state, the assumed word functions might be incorrect. If there are wrong assumptions, the parser algorithm will correct the assumptions, as discussed below.

Now we can partition the `SentenceData` objects into two sets. The first set contains all `SentenceData` objects with the assumed `RELATION` word and the assumed `NEGATION` word, and the second set contains `SentenceData` objects with the assumed `RELATION` word and not the assumed `NEGATION` word. If we have two sets that are not empty, we can proceed, otherwise, we forget the assumptions and repeat word selections, and partitioning the `SentenceData` objects as described above.

Now we can create a class whose name is the assumed `RELATION` word. We assign the class value `FALSE` with the set that has the assumed `RELATION` word and the assumed `NEGATION` word, and associate the class value to `TRUE` with the set that has the assumed `RELATION` word and not the assumed `NEGATION` word. For example we have sentence "`A is left of B`," so we associate the corresponding `RelationalData` object to the class value `TRUE`, and for the sentence

"A is not left of C," we associate the corresponding RelationalData object to the class value FALSE for the assumed relation "left."

We then generate RelationalData objects for the appropriate SentenceData objects and feed the RelationalData objects together with corresponding class value to a machine-learning algorithm that is selected by the user prior to parsing. The user is allowed to select which machine-learning algorithm to use.

Then a model for the assumed relation and negation words is built using a machine-learning algorithm. After that, all the RelationalData objects in the FALSE set are checked if they are all classified using machine-learning algorithm as FALSE, and all the RelationalData objects in the TRUE set are checked to see if they are all classified as TRUE. If both of those conditions are met, we change the main word types in appropriate augmented n-grams associated with selected words to NEGATION and RELATION respectively, and repeat the process for the remaining pairs of UNKNOWN words and assumed roles. This is because there might be more than one form of NEGATION word in a language, such as the phrase "is not" can be contracted to "isn't" in English.

The next step is to test our hypothesis of the assumed word types by checking if we can find more RELATIONS by using an assumed NEGATION word. We then select a word not assumed to be RELATION or NEGATION from the list of words classified as UNKNOWN. We then partition the SentenceData objects, build a model, test the assumptions, and mark the words as discussed above in the previous paragraph. We repeat this step for all the remaining words classified as UNKNOWN.

If, after performing the step described in the previous paragraph we have more than one assumed RELATION the assumed RELATIONS are called actual RELATIONS and the assumed NEGATION is called actual NEGATION. We remember the models associated with RELATIONS. Otherwise, the assumed NEGATIONS are not NEGATIONS at all, so we set them to UNKNOWN, and the assumed RELATION is actually a NEGATION.

Now we can find the remaining RELATIONS by choosing a NEGATION word, constructing sets, building models, and testing as above if in fact the assumed RELATION is actually 100% relation as described two paragraphs above.

For all the actual RELATIONS we remember the models. The first parser is shown in Figure 20 below.

The proposed first parser algorithm is of the polynomial complexity shown by the function in Equation 4 where k is the average number of words in a sentence, m is the number of relations for learning, n is the number of points in the associated grid, o is the number of occurrences of a negation word in the set of training sentences, $l(x)$ is the time required for learning x instances, and $c(x)$ is the time required for classifying one instance based on learned x instances. The first proposed parser algorithm's memory usage is of polynomial complexity, as shown in Equation 5, where $\acute{l}(x)$ is the memory required to learn from x instances, and $\acute{c}(x)$ is the memory required to classify an instance using a model created from x instances. The time of classification ($c(x)$) and learning ($l(x)$), and the memory required for learning ($\acute{l}(x)$) and classification ($\acute{c}(x)$), are all dependent on the machine-learning algorithm chosen to be used during parsing. Equation 6 shows maximum number of augmented n-grams returned by the proposed parser algorithms.

$$f(k,m,n,o) = O\left(m^2n^4 + k^2m^2 + kmo + mn \times l(n^2) + (mn + n^2) \times c(n^2)\right)$$

Equation 4. Function depicting the complexity of the first proposed parser algorithm.

$$g(m,n,k) = O\left(mn^2k + \acute{l}(mn^2k) + \acute{c}(n^2)\right)$$

Equation 5. Memory requirement for the proposed parser algorithms.

$$h(m,k) = 2mk - m$$

Equation 6. Maximum number of augmented n-grams returned by the proposed parser algorithms.

```
Algorithm Parser I
Input:  Sentences and associated grid
Output: Augmented trigram, word classifications and models associated with
RELATIONS
Steps:
   1. Convert all sentences into SentenceData objects.
         a. Detect point names.
         b. Create RelationText by removing point names from sentence text.
         c. Store unaltered sentence text, RelationText and point names in
            SentenceData object.
   2. Create augmented n-gram objects from words in the sentences.
   3. Merge compatible augmented n-gram objects from all sentences.
   4. For all the augmented n-gram objects determine the ratio of use of the
      main word from the augmented n-gram object to the number of sentences.
         a. If the ratio is between 0.5 and 1 inclusively, the middle word
            from augmented n-gram is VERB.
```

        i. Modify main word type to VERB in corresponding augmented n-gram.
5. For all SentenceData objects:
    a. If the sentence from a SentenceData object has 3 words, the word that is not POINT_NAME is RELATION_VERB.
        i. Modify main word type to RELATION_VERB in corresponding augmented n-gram object.
6. For all the augmented n-grams which were previously not classified.
    a. Choose an augmented n-gram and assume it is a RELATION.
        i. Create a class whose name is the main word from chosen augmented n-gram.
    b. Choose a different augmented n-gram and assume it is a NEGATION.
    c. Pick SentenceData objects which was created from a sentence which contains assumed RELATION and the assumed NEGATION and generate RelationalData objects and move them to set associated with class value FALSE
    d. Pick SentenceData objects which correspond to sentence which contains assumed RELATION and does not contain the assumed NEGATION and generate RelationalData objects and move them to set associated with class value TRUE.
    e. Pick SentenceData objects which correspond to sentence which contains assumed RELATION and the assumed NEGATION and generate RelationalData objects and move them to set associated with class value FALSE.
    f. Train a machine-learning algorithm with the fields of the RelationalData objects and the associated class values.
    g. Use the model created by a machine-learning algorithm to check if all the RelationalData objects in the set associated with FALSE and TRUE values are classified as FALSE and TRUE values respectively. If so, remember the assumptions and models associated with assumed RELATIONS. Otherwise, forget the model associated with the chosen augmented n-gram, and the assumptions for the chosen augmented n-grams.
7. For all the augmented n-grams:
    a. Pick one of the assumed NEGATION augmented n-gram.
    b. For all the n-grams which were previously not classified.
        i. Select augmented n-gram and assume the selected augmented n-gram to be a RELATION.
        ii. Create class whose name is the value of main word from the selected augmented n-gram.
        iii. Pick SentenceData objects which correspond to sentence which contains assumed RELATION and does not contain the assumed NEGATION, and generate RelationalData objects and move them to set associated with class value TRUE.
        iv. Pick SentenceData objects which correspond to sentence which contains assumed RELATION and the assumed NEGATION, and generate RelationalData objects and move them to set associated with class value FALSE.

v. Train a machine-learning algorithm with the fields of the RelationalData objects and the associated class values.
vi. Use the model created by a machine-learning algorithm to check if all the RelationalData objects in the set associated with FALSE and TRUE values are classified as FALSE and TRUE values respectively. If so, remember the assumptions and models associated with assumed RELATIONS. Otherwise, forget the model associated with the assumed RELATION, and forget the assumption.

8. If we have one assumed RELATION remembered after steps 6 and 7, set the assumed RELATION to NEGATION, and forget the assumed NEGATION and the model associated with assumed RELATION, and repeat step 7 once and then skip step 8.

Figure 20. The first parser algorithm.

The Improved Algorithm

The first algorithm misses relations, such as "touching" in the sentences "X is touching Y." and "X is not touching Y on the corner." The failure occurs because the word "touching" occurs for the same pair of points in two different structures. In the first example the word "touching" should be treated as a single word, whereas in the second example the word "touching" should be treated as a part of a phrase that expresses a more complex relation. For that reason, there are two more steps to make after the execution of the first algorithm described above.

The first step is to remove sentences from the set of sentences given for learning containing relations that have been already classified.

The second step is the same as finding the remaining relations after finding true negation as described in the previous subsection but this time we use the reduced set of sentences. That means, we remove sentences containing known relations from the set of sentences given for training.

For all the determined RELATIONS we need to remember the models or the RelationalData objects and appropriate class values used for finding the determined RELATIONS, as we will be constructing sentences based on a situation between two points as shown in a grid. The details of the improved parser algorithm are shown in Figure 21 below.

The proposed improved parser algorithm is of the polynomial complexity shown by the function in Equation 7 where k is the average number of words in a sentence, m is the number of relations for learning, n is the number of points in the associated grid, o is the number of occur-

rences of a negation word, $l(x)$ is the time required for learning x instances, and $c(x)$ is the time required for classifying one instance based on learning from x instances. The equations showing memory usage of the first proposed parser and the number of returned augmented n-grams by the first proposed parser apply as well to the proposed improved parser algorithm.

$$f(k,m,n,o) = O\left(m^2n^4 + k^2m^2 + 7kmn^2 + kmo + km \times l(n^2) + (km + n^2) \times c(n^2)\right)$$

Equation 7. Function depicting the complexity of the improved proposed parser algorithm.

```
Algorithm Parser II
Input:  Sentences and associated grid
Output: Five word augmented n-grams, word classifications and models
associated with RELATIONS
Steps:
   1. Convert all sentences into SentenceData objects.
         a. Detect point names.
         b. Create RelationText by removing point names from sentence text.
         c. Store unaltered sentence text, RelationText and point names in
            SentenceData object.
   2. Create augmented n-gram objects from words in the sentences.
   3. Merge compatible augmented n-gram objects from all sentences.
   4. For all the augmented n-gram objects determine the ratio of use of the
      main word from the augmented n-gram object to the number of sentences.
         a. If the ratio is between 0.5 and 1 inclusively, the middle word
            from augmented n-gram is VERB.
               i. Modify main word type to VERB in corresponding augmented n-
                  gram.
   5. For all SentenceData objects:
         a. If the sentence from a SentenceData object has 3 words, the word
            that is not POINT_NAME is RELATION_VERB.
               i. Modify main word type to RELATION_VERB in corresponding
                  augmented n-gram object.
   6. For all the augmented n-grams which were previously not classified.
         a. Choose an augmented n-gram and assume it is a RELATION.
               i. Create a class whose name is the main word from chosen
                  augmented n-gram.
         b. Choose a different augmented n-gram and assume it is a NEGATION.
         c. Pick SentenceData objects which was created from a sentence which
            contains assumed RELATION and the assumed NEGATION and generate
            RelationalData objects and move them to set associated with class
            value FALSE
         d. Pick SentenceData objects which correspond to sentence which
            contains assumed RELATION and does not contain the assumed
            NEGATION and generate RelationalData objects and move them to set
            associated with class value TRUE
         e. Train a machine-learning algorithm with the fields of the
            RelationalData objects and the associated class values.
```

f. Use the model created by a machine-learning algorithm to check if all the RelationalData objects in the set associated with FALSE and TRUE values are classified as FALSE and TRUE values respectively. If so, remember the assumptions and models associated with assumed RELATIONS. Otherwise, forget the model associated with the chosen augmented n-gram, and the assumptions for the chosen augmented n-grams.

7. For all the augmented n-grams:
   a. Pick one of the assumed NEGATION augmented n-gram.
   b. For all the n-grams which were previously not classified.
      i. Select augmented n-gram and assume the selected augmented n-gram to be a RELATION.
      ii. Create class whose name is the value of main word from the selected augmented n-gram.
      iii. Pick SentenceData objects which correspond to sentence which contains assumed RELATION and does not contain the assumed NEGATION, and generate RelationalData objects and move them to set associated with class value TRUE
      iv. Train a machine-learning algorithm with the fields of the RelationalData objects and the associated class values.
      v. Use the model created by a machine-learning algorithm to check if all the RelationalData objects in the set associated with FALSE and TRUE values are classified as FALSE and TRUE values respectively. If so, remember the assumptions and models associated with assumed RELATIONS. Otherwise, forget the model associated with the assumed RELATION, and forget the assumption.
8. If we have one assumed RELATION remembered after steps 6 and 7, set the assumed RELATION to NEGATION, and forget the assumed NEGATION and the model associated with assumed RELATION, and repeat step 7 once and then skip step 8.
9. Remove SentenceData for sentences that contain known RELATIONS from steps 6 and 7.
10. Repeat step 7 once with the rest of SentenceData objects skipping steps 8, 9, and 10.

Figure 21. The improved parser algorithm.

3.3.4. Sentence Creation

The sentence creation algorithm is used in our program to describe the learned relations. Sentences can at least in theory be created by reversing the function of a grammar dependent parser by using the grammar rules, a set of words and their functions in sentences. That would create sentences that are grammatically correct but might make no sense [7], for example "Green nails seldom make explanations." With a statistical parser or approach such as ours using augmented n-grams, however, in place of grammar rules we have statistics regarding frequent usage patterns, and, in our case, information we have induced concerning word function and the

meaning of the expression for positional relations. Our proposed sentence generation algorithm creates sentences based on the data collected by the proposed parser algorithm. Those generated sentences are not just copies of the sentences from a training set.

Both of the proposed sentence generation algorithms complete their tasks if they reach terminating tokens on both sides of the sentence being generated, and the sentence being generated includes given point names. The proposed sentence generation algorithms are successful if they create a grammatically correct sentence, and fail if the generated sentence is not grammatical. These proposed sentence generation algorithms abort the process of generating a sentence if the sentence that is generated is not growing in size and the terminating words have not been included, or one of the point names has not been included.

After the sentences are created and deemed valid, the program outputs the sentences to the user via GUI, as shown in schematic from Figure 2.

First Fits Augmented Trigram Approach

We repeat the following process for the point name chosen by user and the other point names that are present in the grid.

Since we need to describe a situation represented in a grid, we pick a `RELATION` word corresponding to an augmented n-gram as it was determined by the first parser algorithm. We test if the relation applies to the selected ordered pair of points with the help of a corresponding `RelationalData` object and the corresponding model associated with the selected relation word. This shows whether the relation word is true for the selected ordered pair of points.

From the list of augmented n-grams we then pick the augmented n-gram which corresponds to the `NEGATION` word. If the relation word was classified as `FALSE` for the situation between two selected points with the help of machine-learning algorithm, we need to include the negation in the sentence but in appropriate spot in sentence that is going to be created as described below.

Then we begin creating a sentence starting from the `RELATION` word. We need to have variables that would store a partial state of the sentence, and we call them left and right current augmented n-grams for the left and right, respectively, side of the sentence being created. The side of sentence is defined to be everything that is in sentence on the left or the right of a `RELATION` word. We then set the current left and the current right augmented n-gram to the aug-

mented n-gram whose middle word is the RELATION word. Now we will discuss how to create the part of the sentence which is on the left side of the RELATION word. The creation of the right side of the sentence follows the same process as for the left side.

We repeat the following steps until we encounter the current left augmented n-gram in which the middle word is preceded by NOTHING, which means we need to terminate the sentence on the left side. Otherwise, we proceed as described below.

We choose augmented n-gram from the list of augmented n-grams that is compatible with the word on the left of the current left augmented n-gram as described in 3.3.3. If the word is a NEGATION, and we need to include a NEGATION, we do so by choosing a NEGATION instead of the selected word and remember that we already included a NEGATION, otherwise, we select an augmented n-gram whose main word is not a NEGATION and is compatible with one of the words on the left of the main words of the current left augmented n-gram. We then include the main word from the selected augmented n-gram on the left side of the sentence and set the current left augmented n-gram to the selected augmented n-gram.

We do same thing, respectively, for the right side of the sentence being created as we did for the left side of the sentence.

Since we do not know from the beginning where will be the left and right point names, we replace the right point name placeholder with the point name that represents the object, and then the left point name placeholder with the point name that represents the subject.

Sometimes we create a sentence that could not be created, such as in case of the NEGATION was needed and was not included in sentence the sentence is invalid for output. In addition, the inclusion of only one point name makes the sentence invalid. These invalid sentences are discarded.

Because we attempt to create sentences for all augmented n-grams whose main word type is RELATION, we need to remove some of the repeated sentences because the RELATION words might be a part of a phrase that would result in a duplicate sentence. Therefore, we include only one of such duplicating sentences in the answers.

Without this algorithm, the SentenceLearner would not be expressing the learned language, as it would only repeat the phrases given to it, therefore, this step is crucial in expressing observed relations. The entire first fits augmented trigram sentence creation algorithm is shown in Figure 22 below.

The algorithm is a greedy algorithm which tries a random word from a bag of words and takes the first word that fits at a given spot in the sentence. The algorithm has polynomial complexity shown in function in Equation 8. On average, the algorithm's average memory requirements are shown in Equation 9, where k is the average number of words in sentences given for training, r is the average size in bytes for an augmented n-gram, and p is the average size of a word in bytes from the training sentences.

$$j(m,k) = O\big(mk + c(1)\big)$$

Equation 8. Function showing the time needed for completion of a sentence by the proposed sentence creation algorithms.

$$s(k,p,r) = O(kp + r)$$

Equation 9. Function showing the memory requirements of the proposed sentence creation algorithms.

Algorithm SentenceCreation I
Input: Augmented n-grams, point names, grid, models associated with RELATIONS, augmented n-gram whose main word type is RELATION.
Output: Sentence and a flag for valid sentence
Steps:

1. Initialize variable isSentenceValid to TRUE.
2. Store selected augmented n-gram whose main word type is RELATION to relationAugmentedNGram.
3. For all augmented n-grams.
    a. If augmented n-gram main word type is NEGATION store value of main word into negationWord variable and proceed to step 4. Otherwise proceed to step 3.a. with another augmented n-gram that was not tested.
4. From models for RELATIONS pick the one that corresponds to the value of main word from relationAugmentedNGram.
5. Build RelationalData object for given point names.
6. Classify the RelationalData object with the model corresponding to RELATION.
7. Initialize variable isNegationNeeded to the class value that was obtained from the model after classifying RelationalData object.
8. Initialize variables isNegationIncludedInSentence, isSentenceTerminatedOnLeft, and isSentenceTerminatedOnRight to FALSE.
9. Initialize variable sentence to the value of main word from relationAugmentedNGram
10. Initialize variables currentLeftAugmentedNGram and currentRightAugmentedNGram to the value of relationAugmentedNGram.
11. For all augmented n-grams.
    a. Chose an augmented n-gram.
    b. If isSentenceTerminatedOnLeft = FALSE go to step 11.b.i. Otherwise skip all steps before step 11.c.

i. If main word from chosen augmented n-gram precedes main word from currentLeftAugmentedNGram proceed to step 11.b.i.1. Otherwise skip all steps before step 11.c.
    1. If isNegationNeeded = TRUE and isNegationIncludedInSentence = FALSE proceed to step 11.b.i.1.a. Otherwise skip all steps before step 11.c.
        a. If main word from chosen augmented n-gram = negationWord proceed to step 11.b.i.1.a.i. Otherwise skip all steps before step 11.b.i.1.b.
            i. Change value of sentence to negationWord + " " + sentence.
            ii. Change the value of isNegationIncludedInSentence to TRUE.
            iii. Change the value of currentLeftAugmentedNGram to the value of the chosen augmented n-gram.
            iv. If main word from the chosen augmented n-gram is preceded by NOTHING, change the value of isSentenceTerminatedOnLeft to TRUE.
            v. Skip all steps before step 11.c.
        b. If main word from currentLeftAugmentedNGram is not preceded by negationWord skip all steps before step 11.b.i.2.a. O therwise skip all steps before step 11.c.
    2. If isNegationNeeded = FALSE or isNegationIncludedInSentence = TRUE proceed to step 11.b.i.2.a. Otherwise skip all steps before step 11.c.
        a. Change value of sentence to value of main word from selected augmented n-gram + " " + sentence.
        b. Change the value of currentLeftAugmentedNGram to the value of the chosen augmented n-gram.
        c. If main word from the chosen augmented n-gram is preceded by NOTHING, change the value of isSentenceTerminatedOnLeft to TRUE.

c. If isSentenceTerminatedOnRight = FALSE go to step 11.c.i. Otherwise skip all steps before step 11.d.
    i. If main word from chosen augmented n-gram follows main word from currentRightAugmentedNGram proceed to step 11.c.i.1. Otherwise skip all steps before step 11.d.
        1. If isNegationNeeded = TRUE and isNegationIncludedInSentence = FALSE proceed to step 11.c.i.1.a. Otherwise skip all steps before step 11.c.i.2.

a. If main word from chosen augmented n-gram = negationWord proceed to step 11.c.i.1.a.i. Otherwise skip all steps before step 11.c.i.1.b.
   i. Change value of sentence to sentence + " " + negationWord.
   ii. Change the value of isNegationIncludedInSentence to TRUE.
   iii. Change the value of currentRightAugmentedNGram to the value of the chosen augmented n-gram.
   iv. If main word from the chosen augmented n-gram is followed by NOTHING, change the value of isSentenceTerminatedOnRight to TRUE.
   v. Skip all steps before step 11.d.
b. If main word from currentRightAugmentedNGram is not preceded by negationWord skip all steps before step 11.c.i.2.a. Otherwise skip all steps before step 11.d.

2. If isNegationNeeded = FALSE or isNegationIncludedInSentence = TRUE.
   a. Change value of sentence to sentence + " " + value of main word from selected augmented n-gram.
   b. Change the value of currentRightAugmentedNGram to the value of the chosen augmented n-gram.
   c. If main word from the chosen augmented n-gram is followed by NOTHING, change the value of isSentenceTerminatedOnRight to TRUE.
   d. Proceed to step 11.d.

d. If there are more n-grams to choose from from step 11.a skip all steps before step 11.e. Otherwise proceed to step 11.d.i.
   i. If the sentence from step 11 has not been expanded by words from steps 11.b and 11.c set the value of isSentenceValid to FALSE and exit the algorithm. Otherwise jump to step 11.

e. If isSentenceTerminatedOnLeft = TRUE and isSentenceTerminatedOnRight = TRUE go to step 12. Otherwise jump to step 11.a.

12. If sentence has 2 placeholders for point names, replace them with respective point names, otherwise set the value of isSentenceValid to FALSE and skip all remaining steps.

13. If IsNegationNeeded = TRUE and isNegationIncludedInSentence = FALSE, set the value of isSentenceValid to FALSE.

Figure 22. First fits augmented n-gram algorithm for creating sentences. This algorithm is called inside two loops that choose point names, and augmented n-grams whose main word type is `RELATION`**. If the value of the isSentenceValid variable remains** `TRUE` **after the run of the above algorithm, the sentence created by this algorithm is given to further processing in those loops.**

Most Probable Five Word Augmented N-Gram Approach

The most probable five word augmented n-gram approach was created since the previous approach created sentences that were incorrect grammatically, as shown in 4.3.1. Also the algorithm discussed above would not build sentences for the relation "`touches on the corner`" as in the sentence "`X touches Y on the corner`," since after building the part "`Y on the corner`," the algorithm would mark the created sentence as terminated on both sides. We do not check if the algorithm has included a verb since some verbs in the training sentences are not detected by the proposed parser. In addition, the algorithm discussed above would probably create more incorrect sentences if the training sentences contained grammar or spelling errors.

We will describe how to create the left side of the sentence as the right side follows the same process. We proceed as discussed previously in this subsection above except we change three details. First, we find which word on the left side of the current left augmented n-gram is the most likely word. We then select the augmented n-gram that is compatible with the selected word and proceed as above with the `NEGATION`. For the second change, before the sentence validation if we have exactly one word on the left side of the sentence being created, we initialize a variable called previous left augmented n-gram to hold the state of the current left augmented n-gram variable before it is replaced with the selected augmented n-gram value. For the third change, we choose the most probable word taking care of the fact that we need to look if the two leftmost words are preceded by the chosen word in the previous left augmented n-gram, and we need to include a `NEGATION` as previously in this subsection.

The changes apply, similarly to the right side of the sentence as well. After we have reached end of the sentence creation process, we validate the sentence by the method described in previously in this subsection. The entire most probable five word augmented sentence creation algorithm is shown in Appendix A Figure 39.

This algorithm also has a polynomial complexity shown in function in Equation 8 and Equation 9.

3.4. Experimental Design

The experiments were divided into four groups. The first group called the early stage experiments were the starting point for our research, and involved association of the simple relations with the sentences partitioned by `RelationText`. The second group of experiments was

performed with the help of the parser algorithms described above. The object of those experiments was to determine whether it is possible to associate a meaning to a word that is a relation or is a part of a relational phrase. The third part of the experiments was the attempts to create sentences using the data collected by one of the parser algorithms. The fourth phase of experiments was to determine which machine-learning algorithms are most suitable for learning those meanings of relation words.

3.4.1. Early stage experiments

The first two experiments were conducted using grid shown in Figure 23 and a list of sentences shown in Appendix A Figure 40. The sentences are automatically generated by an auxiliary function in `SentenceLearner`, and they accurately describe relationships in Figure 23.

The first experiment included association of one of the simple relations to the sets of sentences. The second experiment involved sorting the sentences into sets that expressed the same relation, learning of relations expressed in sentences with the help of the `J48` machine-learning algorithm, and then checking with the help of the same machine-learning algorithm whether the sets are consistent with the learned relations.

As you will note, however, the set of sentences used for training could not contain the relation "`in same column`" as in sentence "`X is in same column as Y`," because there are no points that are aligned in the same column, which in turn causes problems that are discussed in 4.1.

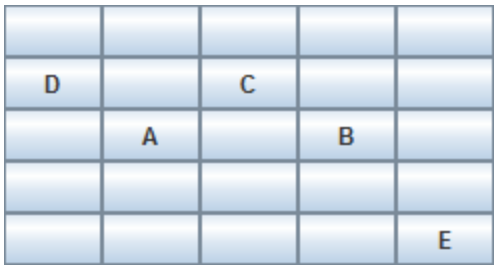


Figure 23. The 5x5 grid for which sentences (Appendix A Figure 40) describing it were used in the first two experiments.

3.4.2. Parser

The parser was tested in the two phases. In both phases, the parser categorized the words in the given set of sentences shown in Appendix A Figure 41 describing a grid, which is shown in Figure 24. This grid has one or more points in every cell so, that includes all the possible rela-

tions. All of the parser algorithms were tested on sentences shown in Appendix A Figure 41 and Figure 43. The first and the improved parser algorithms are described in 3.3.3.

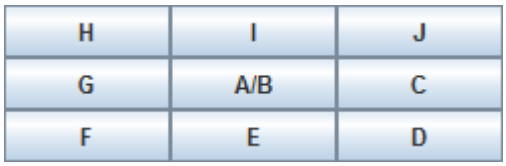


Figure 24. The 3x3 grid used for training of the parser.

Each experiment of the parser performance started with the initial state in which the `SentenceLearner` had no prior knowledge of words and their functions in sentences, and the program used the `J48` machine-learning algorithm. In each of the experiments, only one of the mentioned sets of sentences was used. The parser had to distinguish between `RELATION`, `NEGATION`, `POINT_NAME`, and `VERB`.

The first parser algorithm did not recognize all possible relations, so the second test was conducted using the improved parser, which recognized all the possible relations correctly.

3.4.3. Sentence Building

This part of experiments is divided into two parts. Once SentenceLearner learned all of the interesting word functions from the English sentence set, the program had to output Sentences associated with the grid shown in Figure 25.

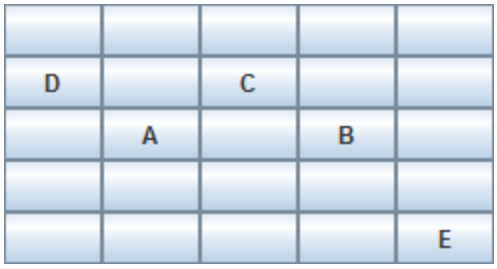


Figure 25. Grid that had to be described for a selectected point to all other point combinations.

Since the first proposed sentence creation algorithm, described in 3.3.4, did not work correctly for the data extracted from grammatically correct English sentences, the test for the second approach, described in 3.3.4, was performed using the English and Polish language and the improved algorithm. The program again used the `J48` machine-learning algorithm to determine whether relations apply to selected pair of points.

3.4.4. Machine-Learning Algorithm and Its Role In Learning Relationships

Since not all machine-learning algorithms are created equal in the terms of the accuracy and applicability of the model, the machine-learning algorithms had to be tested for learning various relations expressed in the set of sentences that were used in the experiments mentioned above.

Each run of the test started with the initial state of the algorithm, that has no prior knowledge of the words and their functions in sentences. The `SentenceLearner` had to learn from sentences and then describe a selected point from grid shown in Figure 25 above. Some interesting results are included in the experiments section.

The list of sentences for training was generated, by the sentence generating module, that does not participate in the learning process. The structure of the sentences produced by the sentence generating module was given by the programmer. The rules of sentence generation included the words that need to go to each of the training sentences, and where the point names need to be inserted. The sentence generating module saved the grid information together with the set of sentences.

With that information, the program was set to automatically read the lists of sentences, sort them out according to the wording, and then sort them out in accordance to the internal classification system, like simple relations, and their negations.

The program then removes the sets of sentences which were not really meant to represent the simple relation, and then remove duplicates for all the sentence relations that are found to be included for the negations of the simple relations, for example, "`A is above B.`" represents the same information in different words as the combined two sentences: "`A is not below B.`" "`A is not in the same column as B.`" This would cause problems with repeated sets of sentences as a machine-learning algorithm would over train for an answer.

Another approach is to use the list of sentences using the same relation and pass the sentences to a machine-learning algorithm thus the program will learn `true` for `left` for the ordered pair of points `X` and `Y`. The program will then use other sentences with different relation so that the point names will be substituted from the other sentences giving `false` for `left` for the ordered pair of points `Y` and `X`. For a machine-learning algorithm, the set of simple attributes also consist of numeric values because of discovery concerning counting by bees by M. Dacke [31].

In addition, different machine-learning algorithms will be used to learn the meaning of words and their functions, so that the learning algorithm with the most accurate results will be the preferred one for future research.

Chapter 4: Results and Discussion

The program is able to learn meaning of a set of sentences, and is able to learn correctly the relation words for simple relations, such as `ABOVE` vs. `RIGHT`, or RelationalDatas.

4.1. Early Stage Experiments

Since the sentences (see Appendix A Figure 40) did not contain the relation for "`in same column as`" as in the sentence "`X is in same column as Y`," the algorithm did not associate the sentences with the right relation, whether it was a simple relation or whether it was a relation extracted from sentences as discussed in 3.3.1 and 0.

4.1.1. Experiment 1: Association of Simple Relations with Relations Expressed in Sentences without Using Machine-Learning Algorithms

The goal was to associate the sentences given for training with simple relations without the use of a machine-learning algorithm using `Simple Association` algorithm. First, the sentences were sorted into sets according to `RelationText`, and then the associations were made. The inputs were the sentences from Appendix A Figure 41 and grid from Figure 26.B. The output is sets of sentences that were associated with simple relations.

The figure below shows the result of the quick association of sentences with simple relations. Some of the sentences were not associated with simple relations correctly because there was no representation of the simple relation `SAME_COLUMN` in the sentences from which the program associated meanings. Out of 16 groups of sentences grouped by `RelationText`, 8 were associated correctly with simple relations as shown in Figure 27. However, some sentence groups were associated incorrectly with more than one simple relation or its negation because of poor quality of the training data. However, this should self-correct when there is a better quality training data provided.

A)

```
Sentences associated with simple relation SAME_ROW:
[[A is in same row as B., B is in same row as A., C is in same row as
D., D is in same row as C.]]

Sentences associated with negation of simple relation SAME_ROW:
[[A is below C., A is below D., B is below C., B is below D., E is below
A., E is below B., E is below C., E is below D.]]
```

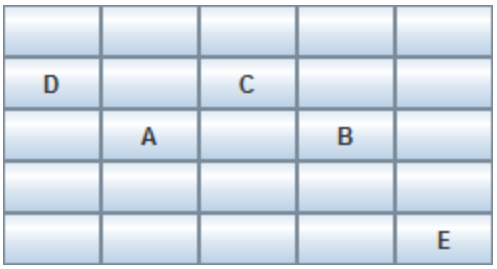
B) 

Figure 26. A) Some of the sentences associated with simple relations without the use of a machine-learning algorithm. For the full list of sentences see Appendix A Figure 44. B) Grid used for associations of `RelationTexts` to simple relations.

RelationText	Associated Simple Relation	Correctness
is left of	LEFT	Correct
is not left of	Negation of LEFT	Correct
is right of	RIGHT	Correct
is not right of	Negation of RIGHT	Correct
is not in same column as	Negation of SAME_COLUMN	Correct
is not above	Negation of SAME_COLUMN	Incorrect
is not below	Negation of SAME_COLUMN	Incorrect
does not touch	Negation of SAME_COLUMN	Incorrect
is not in same row as	Negation of SAME_COLUMN	Incorrect
touches	Negation of SAME_COLUMN	Incorrect
touches on the corner	Negation of SAME_COLUMN	Incorrect
is above	ABOVE	Correct
is below	Negation of ABOVE	Incorrect
is below	BELOW	Correct
is above	Negation of BELOW	Incorrect
is in same row as	SAME_ROW	Correct
is below	Negation of SAME_ROW	Incorrect
is near to	TOUCHING	Incorrect
is in same row as	Negation of TOUCHING	Incorrect

Figure 27. Result of association of RelationText with simple relations.

Since relations expressed in the sentences might be more complex than one of simple relations, the approach to associate sentences with simple relations directly was abandoned. In-

stead, the machine-learning algorithms were used, and the results of this approach are described in 4.1.2.

4.1.2. Experiment 2: Association of Relations with Sentences

The next logical step was to use a machine-learning algorithm to associate the relations expressed in sentences with models of the relations with the use of the `Model Association` algorithm. This allowed for learning more complex relations that would have to be associated with more than one simple relation such as "`touches on the corner`," which requires horizontal and vertical distances to be equal to 1. While this feature was working correctly, the training sentences, shown in Appendix A Figure 40, that were used in the earlier experiment discussed in 4.1.1 had caused some mix-ups as shown in Figure 28 below. It was eliminated when the grid with complete set of relations was used shown in Figure 26.B.

As previously, the input was the same as in 4.1.1. This time, the algorithm had to learn to distinguish between the text in the `RelationText` field that could be expressed in different words. The algorithm had to eliminate the sentences without parsing them. The output was the result of association of the models created for the text in `RelationText` field with sets of sentences with the help of a machine-learning algorithm. Out of 16 groups of sentences, 8 were correctly associated with model corresponding to `RelationText` as shown in Figure 29. When comparing the results of this experiment with the results of the experiment discussed in 4.1.1, there was no improvement in the quality of the recognized meanings. This is because of the poor quality of the training data given to the program.

This experiment demonstrates that even without parsing, the meaning of sentences can be learned. However, in this experiment, we noticed that the quality of training data has influence on the understanding of a natural language. Due to this fact, the algorithm was not able to distinguish meanings of some sentences, however this should self-correct when a higher quality training data is supplied.

```
Sets of sentences true for machine-learning model whose class name is "is
left of:"
[[A is left of B., A is left of C., A is left of E., B is left of E., C is
left of B., C is left of E., D is left of A., D is left of B., D is left of
C., D is left of E.]]

Sets of sentences true for machine-learning model whose class name is "is not
left of:"
```

[[A is right of D., B is right of A., B is right of C., B is right of D., C is right of A., C is right of D., E is right of A., E is right of B., E is right of C., E is right of D.]]

Figure 28. The result of association of simple relations with some of the relations expressed in sentences with the use of a machine-learning algorithm. See Appendix A Figure 45 for full list of sentences.

RelationText provided as a class name to the J48 machine-learning algorithm	RelationText of sentences true for the model generated by the J48 machine-learning algorithm	Correctness
is left of	is left of	Correct
is not left of	is right of	Incorrect
is not right of	is not right of	Correct
is not in same column as	is not in same column as	Correct
	is not above	Incorrect
	is not in same row as	Incorrect
	does not touch	Incorrect
	is not left of	Incorrect
	touches	Incorrect
	touches on the corner	Incorrect
	is not below	Incorrect
is not above	is in same row as	Incorrect
is below	is below	Correct
is not in same row as	is below	Incorrect
does not touch	is in same row as	Incorrect
is not left of	is right of	Incorrect
is right of	is right of	Correct
touches	is near to	Incorrect
touches on the corner	is near to	Incorrect
is near to	is near to	Correct
is above	is above	Correct
is not below	is above	Incorrect
is in same row as	is in same row as	Correct

Figure 29. Result of association of RelationText with models created for RelationTexts.

4.1.3. Experiment 3: Check of Learned Knowledge

The goal of this experiment was to test the `Model Association` algorithm described in 0. The input was the set of sentences in Figure 40 along with the corresponding grid shown in Figure 30.B below. The algorithm then produced answers for the grid in Figure 30.C below. The result was 28 out of 28 correct answers.

Each of the answers is grouped by the point to point relation expressed by an arrow that tells the first point is described in relative position to the second point. Then there are phrases captured in `RelationText` fields that describe those relations between the point names.

The program was then queried with pair of points, such as `C` and `A`, and the program had to output the `RelationText` describing the relationships of these pair of points.

The approach tested here was not good for building sentences associated with simple relations since the parser did not save enough information about the sentence structure.

The replacement of point names in each of the sentences used for training would not be considered language learning since if the sentences contained grammar or spelling errors, the algorithm would not be able to correct those errors.

This approach also suffers from another shortcoming. If there are not enough sentences describing a relation in a grid such as when number of sentences describing the grid is less than $n(n-1)m$ sentences with m concepts for a grid with n points, a machine-learning algorithm might create a model that would classify everything as `FALSE`, thus preventing learning and association of a `RelationText` to the relational position of points. The parser approach does not have this problem as it will deduce the meaning of words given in training sentences.

A)

C --> A
is not left of
is right of
is not in same column as
is above
is not below
is not in same row as
touches
C --> B
is left of
is not right of
is not in same column as
is above
is not below
is not in same row as
touches
C --> D
is not left of
is right of
is not in same column as
is not above
is not below
is in same row as
does not touch
C --> E
is left of
is not right of
is not in same column as
is above
is not below
is not in same row as
does not touch

B)

H	I	J
G	A/B	C
F	E	D

C)

D		C		
	A		B	
				E

Figure 30. Program's answers[2] (A) for grid (C) where answers were correct. B) The grid used for learning.

[2] "X --> Y" means position of X relative to Y.

4.2. Parser

The program was not able to learn successfully if a negation of relation is missing from the list of sentences as is in case of sentences in Figure 40 associated with simple relations describing grid in Figure 26.B.

With the machine learning approach to designing a relational model for a given phrase held in `RelationText` field, the program learned correctly every time, and did not discard relations that are more complex from the learning point of view.

Yet with all of the mentioned methods for classification, the computer will not be able to learn successfully if there are not enough sentences that say the same thing for different points. With some sentences missing from the set, the computer will not discard the set, but the result of machine learning may not be perfect.

4.2.1. Experiments 4, and 5: First Parser

Experiment 4: English Sentences

The goal was to classify word functions for sentences (see Appendix A Figure 41) that describe grid in Figure 31.B below. The output is the word classification by functions: `NEGATION`, `RELATION`, `RELATION_VERB`, `UNKNOWN`, or `VERB`. Out of 20 word classifications, 16 were correct as shown in Figure 31.

The algorithm would miss the relation "`touching`" if the training sentences were introducing noise for the word "`touching`" as in sentences "`X is touching Y.`" and "`X is not touching Y on the corner.`" for the same pair of points `X` and `Y`. This is due to the fact that the parser would not distinguish the `RelationText` "`is touching`" from "`is not touching on the corner`" because the parser only looks at individual words in all of the sentences. If the sentences used for training included "is touching" instead of "touches" and "is not touching" instead of "does not touch," the results of the word classifications would have been 15 correct of 18. The improved parser would have classified 16 words correctly out of 18.

Because there was no time for comparing the first proposed parser algorithm with a parser that incorporates semantics, the results of such comparison will be included in a future work.

A)

Word	Classification	Expected
is	VERB	VERB
not	NEGATION	NEGATION
left	RELATION	RELATION
right	RELATION	RELATION
column	RELATION	RELATION
above	RELATION	RELATION
below	RELATION	RELATION
row	RELATION	RELATION
near	UNKNOWN	RELATION
on	RELATION	UNKNOWN
the	RELATION	UNKNOWN
corner	RELATION	RELATION
touches	RELATION_VERB	RELATION_VERB
of	UNKNOWN	UNKNOWN
in	UNKNOWN	UNKNOWN
same	UNKNOWN	UNKNOWN
as	UNKNOWN	UNKNOWN
does	UNKNOWN	UNKNOWN
touch	UNKNOWN	RELATION
to	UNKNOWN	UNKNOWN

B)

H	I	J
G	A/B	C
F	E	D

Figure 31. Quick classification of relation against negation (A) with rules from the first proposed parser algorithm, described in 3.3.3. B) The grid used in this experiment.

The only expected but not correct relation words are "on" and "the." This was because training sentences did not include sentences saying, "X is on the left of Y" and "X is on the right of Y," but only "X is left of Y" and "X is not left of Y" where these words do not exist. Thus the word "the" had no noise in the form of inconsistency of meaning in the training set. However, when the training sentences will include the word "the" as in the example given above, the word "the" will not be considered a RELATION.

Another incorrect classification is the phrase "near to" since the learning algorithm had not enough data to learn this relation correctly, because we defined the relation as shown in Chapter 3. Equation 2, and such equation gave us only two sentences for TRUE for the ordered pairs of points (A, B) and (B, A), and as much as 87 sentences for FALSE class value. Therefore, the model created for that relation had always predicted FALSE value for the class, and such false negative values have stopped the parser from detecting the relation.

This issue of detection can be addressed in such a way as to have a machine-learning algorithm classify correctly at least 90% of the training instances so the parser can detect a relation, but this is a topic for future work. The issue of false negatives or false positives can be addressed by adding more examples that would make it easier for a learning algorithm to build a more accurate model for a relation.

Experiment 5: Polish Sentences

The object of this experiment was to check if the parser can process sentences in a language other than English. The reason for this experiment is because the Polish language grammar dictates different adjective forms for different noun genders. In addition, the adjectives that describe the subject can have different form than those that describe the object of a sentence. The inputs to the algorithm are sentences from Appendix A Figure 43 and the grid from Figure 32.B.

The output is the word classification by functions: NEGATION, RELATION, RELATION_VERB, UNKNOWN, or VERB. Out of 19 word classifications, 12 were correct as shown in Figure 32.A. Please note that since the parser does not recognize different word forms, such as the word "tej" and "tym," and the word "samej" and "samym," the result ratio of correct to total number of words is different than would be expected when the parser would recognize these word functions and forms. We plan to address the problem of word forms by comparing the different words letter by letter to see if they have similar form. If they are very similar, there is high likelihood that those word forms are the same word.

Since the program has to learn from the set of sentences, it treats "X nie styka się z Y na rogu," meaning "X does not touch Y on the corner," as noise for the relation "styka się z," meaning "touching," in sentence "X styka się z Y," meaning "X touches Y." Because of that fact, the relation "styka się z" had too much noise in the input and thus model describing the phrase was discarded by the first parser algorithm.

A)

Word	Classification	Expected
nie	NEGATION	NEGATION
jest	VERB	VERB
lewo	RELATION	RELATION
prawo	RELATION	RELATION
tej	RELATION	UNKNOWN
samej	RELATION	UNKNOWN
kolumnie	RELATION	RELATION
powyżej	RELATION	RELATION
poniżej	RELATION	RELATION
tym	RELATION	UNKNOWN
samym	RELATION	UNKNOWN
rzędzie	RELATION	RELATION
rogu	RELATION	RELATION
na	UNKNOWN	UNKNOWN
w	UNKNOWN	UNKNOWN
co	UNKNOWN	UNKNOWN
styka	UNKNOWN	RELATION
się	UNKNOWN	RELATION
z	UNKNOWN	RELATION

B)

H	I	J
G	A/B	C
F	E	D

Figure 32. A) Results of classification of word functions[3] in Polish sentences using the first parser. B) The grid used in this experiment.

4.2.2. Experiment 6: Improved Parser

The object of this experiment was to check if the improved parser can classify more of the word functions from sentences in a language other than English. The inputs to the algorithm are sentences from Appendix A Figure 43 and the grid from Figure 33.B. The output is the word classification by functions: `NEGATION`, `RELATION`, `RELATION_VERB`, `UNKNOWN`, or `VERB`. Out of 19 word classifications, 15 were correct.

[3] Some of the words listed are just different version of the other words, as Polish language requires the form of the word to be changed according to which words follow or precede a word and what gender is the object and subject of the sentence.

Because of the drawbacks discovered in 4.2.1, the next logical step is to eliminate the sentences containing the known relations that were learned to facilitate recognition of other relations that would otherwise be missed. The improvement of the proposed improved parser over the proposed first parser is 20%.

This improved parser also suffers from the same drawback for marking some words incorrectly as relations as the first parser, but this should be corrected when more sentences will be introduced that clarify the word meanings or their functions in sentences.

A)

Word	Classification	Expected
nie	NEGATION	NEGATION
jest	VERB	VERB
lewo	RELATION	RELATION
prawo	RELATION	RELATION
tej	RELATION	UNKNOWN
samej	RELATION	UNKNOWN
kolumnie	RELATION	RELATION
powyżej	RELATION	RELATION
poniżej	RELATION	RELATION
tym	RELATION	UNKNOWN
samym	RELATION	UNKNOWN
rzędzie	RELATION	RELATION
rogu	RELATION	RELATION
na	UNKNOWN	UNKNOWN
w	UNKNOWN	UNKNOWN
co	UNKNOWN	UNKNOWN
styka	RELATION	RELATION
się	RELATION	RELATION
z	RELATION	RELATION

B)

H	I	J
G	A/B	C
F	E	D

Figure 33. A) Note the improvement of recognition of relation words or words belonging to the relational phrase. B) Grid used in this experiment.

4.3. Building Sentences Using Augmented N-grams

Reversing the parser to build a sentence proved to be a challenge. It was, however, done successfully for simple sentences that cannot be broken into smaller sentences. If we want to create more advanced sentences, we need to improve the parser as well as the sentence building algorithm, which will be a good topic for a future work.

While it might seem that the sentences cannot be easily built just starting from the relation word to the outwards of the sentence, all of the sentences that were only one not breakable sentence were built correctly from the given starting point, which was the relation word. However, we cannot assume that is the case with the language composition process in humans. If the sentence composition mechanisms will be discovered and will differ, for example, sentence will be built from the beginning and not the relation word, the sentence-building algorithm can be easily modified to incorporate such discovery.

4.3.1. Experiment 7: Augmented Trigrams – Not a Good Sentence Building Material

The object of this experiment was to determine whether it is possible to make sentences by just looking one word ahead or backwards from the rightmost or leftmost word, respectively, in currently created sentence. The augmented trigrams were created by the first parser algorithm from sentences in Appendix A Figure 41 that describe grid in Figure 34.A above. In addition, the sentence creation algorithm had to describe grid shown in Figure 34.C. The output is shown in Figure 34.B. Out of 26 sentences only 10 were built correctly for the English grammar rules.

In addition, the first fits algorithm described in 3.3.4 would not build sentences for the relation "`touches on the corner`" as in the sentence "`X touches Y on the corner`," since after building the part "`Y on the corner`," the algorithm would mark the created sentence as terminated on both sides. This is because after the algorithm encountered the augmented trigram whose main word type was `POINT_NAME` had the space for preceding word marked as `NOTHING` which caused the algorithm to terminate the left side of the sentence prematurely. In addition, the first sentence creation algorithm does not consider probabilities of a word occurring in a sentence thus leading to incorrect sentences.

Therefore, five word n-grams are needed for correct grammar observations. In addition, the probabilistic word choice is needed for correcting the errors introduced by grammar errors from sentences that might show up in the training set.

A)

H	I	J
G	A/B	C
F	E	D

C)

D		C		
	A		B	
				E

B)

C does not left of A
C is right of A
C does not in same column as A
C is above A
C does not below A
C does not in same row as A
C touches A
C is left of B
C does not right of B
C does not in same column as B
C is above B
C does not below B
C does not in same row as B
C touches B
C does not left of D
C is right of D
C does not in same column as D
C does not above D
C does not below D
C is in same row as D
C is left of E
C does not right of E
C does not in same column as E
C is above E
C does not below E
C does not in same row as E

Figure 34. A) Grid used for training. Sentences (B) created by program using augmented tri-grams for point C to other points' relations in grid (C).

4.3.2. Experiments 8, and 9: Probabilistic Sentence Building Using Augmented Five Word N-grams

The goal of these two following experiments was to see if there are improvements over the first fit augmented trigram approach in building sentences.

Experiment 8: Probabilistic Sentence Building Using Augmented Five Word N-grams Extracted from Grammatically Correct Sentences in English Language

The object of this experiment was to determine whether it is possible to make sentences by probabilistically choosing a word ahead or backwards from the rightmost or leftmost word, respectively, in currently created sentence. The five word augmented n-grams were created by

the improved parser algorithm from sentences in Appendix A Figure 41 that describe grid in Figure 35.A below.

A)

H	I	J
G	A/B	C
F	E	D

C)

D		C		
	A		B	
				E

B)

C is not left of A
C is right of A
C is not in same column as A
C is above A
C is not below A
C is not in same row as A
C touches A
C does not touch A on the corner
C is left of B
C is not right of B
C is not in same column as B
C is above B
C is not below B
C is not in same row as B
C touches B
C does not touch B on the corner
C is not left of D
C is right of D
C is not in same column as D
C is not above D
C is not below D
C is in same row as D
C does not touch D on the corner
C is left of E
C is not right of E
C is not in same column as E
C is above E
C is not below E
C is not in same row as E
C does not touch E on the corner

Figure 35. A) Grid used for training. Sentences (B) created by program using five word augmented n-grams for point C to other points' relations in grid (C).

In addition, the sentence creation algorithm had to describe grid shown in Figure 35.C. The output is shown in Figure 35.B. Out of 30 sentences, all were built correctly for the English grammar rules. The improved sentence creation algorithm shows improvement of 61.5% over the first sentence generation algorithm when both sentence generation algorithms use the same data extracted from grammatically correct English sentences.

Experiment 9: Probabilistic Sentence Building Using Augmented Five Word N-grams Extracted from Grammatically Correct and Incorrect Sentences in English Language

The object of this experiment was to determine whether it is possible to make grammatically correct sentences from data learned from correct and incorrect sentences. The five word augmented n-grams were created by the improved parser algorithm from sentences in Appendix A Figure 41 and Figure 42 that describe grid in Figure 36.A below. In addition, the sentence creation algorithm had to describe grid shown in Figure 36.C. The output is shown in Figure 36.B. Out of 29 sentences, 23 were built correctly for the English grammar rules. That gives the improvement of 48.5% of the proposed improved sentence creation algorithm that created sentences from data extracted from grammatically correct and incorrect English sentences over the proposed first sentence creation algorithm that created sentences from data extracted from the incorrect and correct English sentences.

If errors in grammar structure are introduced in less than half of the sentences describing a relation, we predict the sentences will be built correctly. There is one exception to this rule as the sentence building algorithm is designed to include a negation word instead of a more likely word if the negation is needed in the sentence.

However, more complex sentences made up from two or more sentences are not built correctly because there is possibility of infinite loop that would make grammatically correct sentence of infinite size, which is of course not desirable side effect of the sentence-building algorithm. The possible solution for this problem is to use a word counter in the algorithm's loop that would stop the algorithm from including more words after the counter reaches a certain value. If the sentence is then terminated, then the sentence that is being created will be passed to the validation part of the algorithm, otherwise, the sentence will not be valid. Another way is to limit the overall time the algorithm spends building a sentence. If the algorithm exceeds the time limit, and the sentence being created is not valid, the sentence will be discarded.

Even though the improved sentence creation algorithm uses probabilities of occurrences, the improved sentence creation algorithm did make mistakes, such as "`C does not touches E.`" even though there was no sentence saying "`X does not touches Y.`" The sentence creation algorithm had the word "`touches`" as a chosen relation, which is the starting point of a sentence creation process. The second the program had seen the point name placeholder, and it was

the most probable word to the right of the relation word, the program included it. The next was check if the two words had been indeed observed as a pair that terminates a sentence. Since there was such observation in sentences, such as "`A touches B`," the algorithm stopped the sentence creation process on the right side of the relation word. This is unavoidable mistake if we have similar observations for all other words.

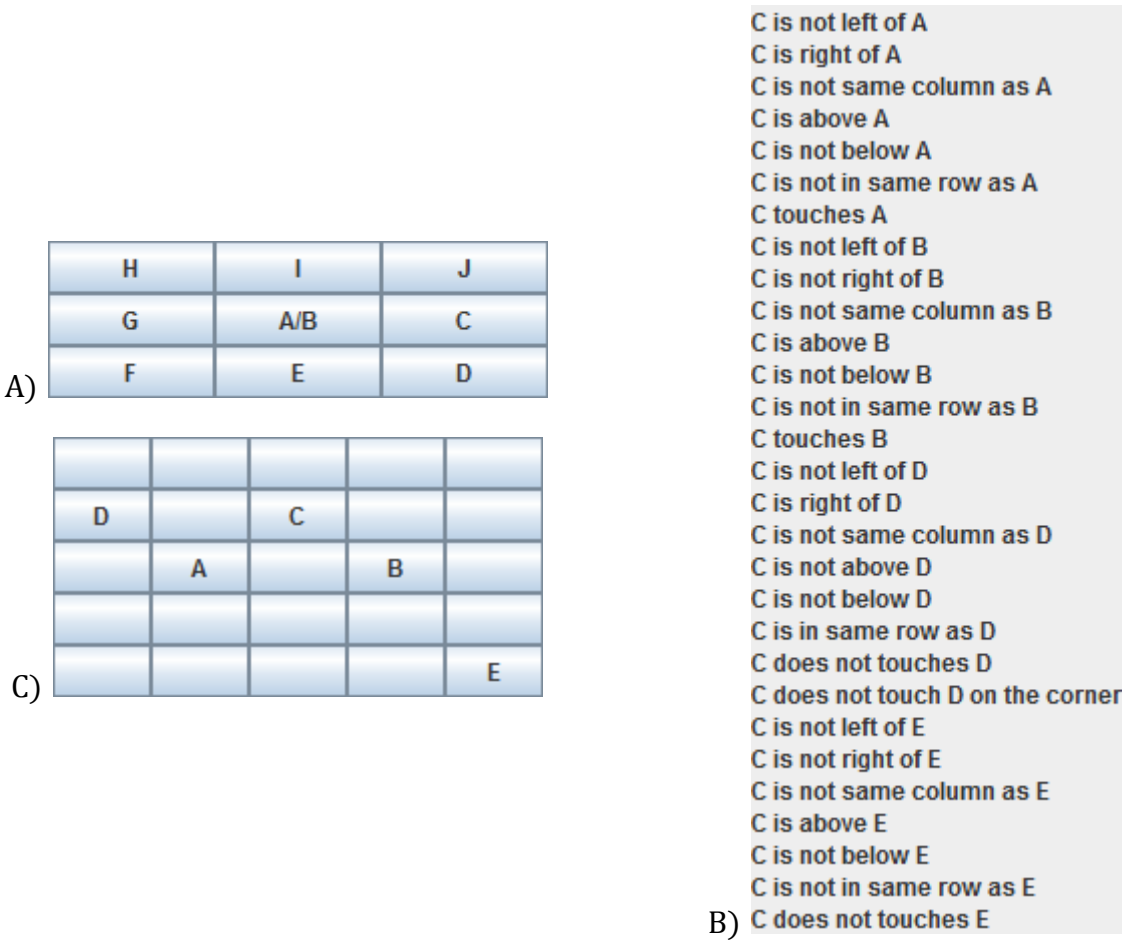


Figure 36. A) Grid used for training. Sentences (B) created by program using five word augmented n-grams for point C to other points' relations in grid (C).

4.4. Machine-Learning Algorithms

The machine-learning algorithms have the ability to infer proper relations while ignoring errors called the noise. Our experiments contained sentences that were 100% consistent making the language learning faster, yet the experiments did not address the issue of noise to building models. This leaves room for changing experiments to include noise, so that the experiments are more resembling real world, but this is a topic for the doctorate thesis.

4.4.1. Experiment 10: Choice of Machine-Learning Algorithms Had Impact on Learning

The object of this experiment was to deduce which machine-learning algorithms are suitable for learning relations. The input were sentences in Appendix A Figure 41 that describe grid in Figure 38.A below, which were parsed using the improved parser and then described with the help of the probabilistic sentence creation algorithm. The output were sentences that describe grid in Figure 38.C. Out of 26 machine-learning algorithms, 1 failed to learn all relations, and 6 have failed to learn the relations "`styka się`" and "`styka się na rogu`" The successful learning algorithm names are shown in Figure 37 below.

While many learning algorithms are quite good at distinguishing simple relations, more advanced relations pose problems for some of the learning algorithms, such as `One Rule`, no matter how large is the data set. In addition, some of the data mining algorithms, such as `Naïve Bayes Simple`, are not adequate for the learning process because the algorithm is not able to handle the data values, such as numeric values and named constants. Furthermore, some learning algorithms while they are slower than the other did not produce perfect models.

In addition, the choice of learning algorithm at the learning phase where word functions in sentences are recognized the choice of learning algorithm seems to have an impact over future performance of the program in describing relations.

Even with all the relations determined correctly, the program still might need more examples to learn a relation as it was in the case of the figure below and the Backpropagation algorithm. This can only be eliminated by providing a bigger set of examples that the algorithm will learn from.

```
LMT                                RIPPER
NBTree                             Nearest Neighbour
Random Forest                      Standard Logistic Regression
Random Tree                        Multilayer Perceptron
REPTree                            Simple Logistic Regression
Bayes Net                          IB1
Naïve Bayes                        IBk
Naïve Bayes Updateable             Nearest Neighbour
Conjunctive Rule                   Hyper Pipes
Simple Decision Table
```

Figure 37. Machine-learning algorithms that succesfully learned all the relations expressed in sentences (see Appendix A Figure 41) that describe grid in Figure 38.A above.

A)

H	I	J
G	A/B	C
F	E	D

B)

C jest powyżej od A
C nie jest poniżej od A
C nie jest w tym samym rzędzie co A
C jest powyżej od B
C nie jest poniżej od B
C nie jest w tym samym rzędzie co B
C nie jest powyżej od D
C nie jest poniżej od D
C jest w tym samym rzędzie co D
C jest powyżej od E
C nie jest poniżej od E
C nie jest w tym samym rzędzie co E

C)

D		C		
	A		B	
				E

Figure 38. A) The grid used for training. B) output of the algorithms J48, One Rule, Ripple Down, Radial Basis Function Network, Sequential Min Optimization, Locally Weighted Learning, which failed to learn "`styka się`" and "`styka się na rogu`" relations, whereas Zero Rule algorithm failed to learn even the simple relations. C) Grid that was described.

Chapter 5: Conclusion and Future Work

By intelligently combining a grammarless parser with relation learning, we are able to teach the computer the basic relations of direction, position, as well as distance in a 2D plane. The approach can be expanded to a real world product and it would require combining the proposed algorithms with shape recognition as well as 3D distance measurements, but such is an excellent topic for a doctorate thesis. This can be done in a simple way. The program will learn first based on pictures and training sentences associated with those pictures, for example the program will distinguish a cup from a pen. Then the program will enter a second phase of learning where it will use the proposed parser again to learn the positional relations between objects in pictures. For such learning, the program will need to detect the boundaries of various objects.

In addition, by using the data collected by a statistical parser, our algorithm is able to produce simple sentences that describe situations in a grid.

The algorithms demonstrated here have potential real world applications that can revolutionize the world for the visually impaired as well, as for totally blind people who do not want implants to help them see or do not have visual cortex developed enough for processing of the signals from implants.

In addition the algorithms that would be used for visually impaired people could be used to program intelligent robots that would respond to any sentence that would be directed as a request, or verbal instructions in the user's own language.

While the simple association of sentences to the simple relations works fine, more complex relations need to be learned through machine-learning algorithms. In addition, the simplistic template which detects only point names is not useful in collecting information for building sentences. A simplistic, yet intelligent parser, coupled together with sentence building algorithm is adequate in building simple sentences. Both parser and sentence building algorithm benefit greatly from machine-learning algorithms.

The performance of the parser is therefore dependent on which machine-learning algorithm is chosen, since some of the machine-learning algorithms are too simple to discover the complex relations between sentence meaning and the situation on an associated grid.

5.1. Future Work

In the future, we would like to extend the program to include more complex relations, and a third dimension for the robotics application, as well as the relations for time of past and future tenses in the sentences.

The proposed parser algorithm that learns word meanings will need to be developed further to be useful for any commercial application, yet it does not require dictionaries or predefined grammar structures to extract relations and should be able to learn from most of sentences written in many natural languages.

The algorithm will need, however, to be updated to learn from languages that do not have clear distinction between words, or even word permutations as permitted by grammar and spelling rules of a language as it would be beneficial in creating smaller dictionaries and recognition of words, but that would be topic for a doctorate thesis.

In addition, combining the shape recognition with learning of the meaning of words can be done with the help of these before-mentioned algorithms and this combination is an excellent topic for the doctorate thesis.

We would like to implement changes that support head-last languages. Instead of relying on point names that fall into predetermined range, we would like to allow other words describing such as pen or apple. As an example, this implementation could be then used to describe where an apple is located in relation to a pen.

Appendix A: Additional Figures

Figures in this appendix span across a few pages, and the descriptions of the figures are placed below them.

```
Algorithm SentenceCreation II
Input: Augmented n-grams, point names, grid, models associated with
RELATIONS, augmented n-gram whose main word type is RELATION.
Output: Sentence and a flag for valid sentence
Steps:
   1. Initialize variable isSentenceValid to TRUE.
   2. Store selected augmented n-gram whose main word type is RELATION to
      relationAugmentedNGram.
   3. For all augmented n-grams.
         a. If augmented n-gram main word type is NEGATION store value of
            main word into negationWord variable and proceed to step 4.
            Otherwise proceed to step 3.a. with another augmented n-gram that
            was not tested.
   4. From models for RELATIONS pick the one that corresponds to the value of
      main word from relationAugmentedNGram.
   5. Build RelationalData object for given point names.
   6. Classify the RelationalData object with the model corresponding to
      RELATION.
   7. Initialize variable isNegationNeeded to the class value that was
      obtained from the model after classifying RelationalData object.
   8. Initialize variables isNegationIncludedInSentence,
      isSentenceTerminatedOnLeft, and isSentenceTerminatedOnRight to FALSE.
   9. Initialize variable sentence to the value of main word from
      relationAugmentedNGram
   10.      Initialize variables currentLeftAugmentedNGram and
      currentRightAugmentedNGram to the value of relationAugmentedNGram.
   11.      For all augmented n-grams.
         a. Chose an augmented n-gram.
         b. If isSentenceTerminatedOnLeft = FALSE go to step 11.b.i.
            Otherwise skip all steps before step 11.c.
               i. If variable previousLeftAugmentedNGram is not initialized
                  go to step 11.b.i.1. Otherwise skip all steps before step
                  11.b.i.2.
                     1. If main word from chosen augmented n-gram precedes
                        main word from currentLeftAugmentedNGram proceed to
                        step 11.b.i.1.a. Otherwise skip all steps before step
                        11.c.
                           a. If isNegationNeeded = TRUE and
                              isNegationIncludedInSentence = FALSE proceed to
                              step 11.b.i.1.a.i. Otherwise skip all steps
                              before step 11.c.
```

i. If main word from chosen augmented n-gram = negationWord proceed to step 11.b.i.1.a.i.1. Otherwise skip all steps before step 11.b.i.1.a.ii.
    1. Change value of sentence to negationWord + " " + sentence.
    2. Change the value of isNegationIncludedInSentence to TRUE.
    3. Initialize the value of previousLeftAugmentedNGram to value of currentLeftAugmentedNGram.
    4. Change the value of currentLeftAugmentedNGram to the value of the chosen augmented n-gram.
    5. If main word from previousLeftAugmentedNGram is preceded by NOTHING and main word from the currentLeftAugmentedNGram, change the value of isSentenceTerminatedOnLeft to TRUE.
    6. Skip all steps before step 11.c.

ii. If main word from currentLeftAugmentedNGram is not preceded by negation word skip all steps before step 11.b.i.1.b.i. Otherwise skip all steps before step 11.c.

b. If isNegationNeeded = FALSE or isNegationIncludedInSentence = TRUE proceed to step 11.b.i.1.b.i. Otherwise skip all steps before step 11.c.

i. Initialize variable mostProbableWord to the value of most probable word preceding the main word from currentLeftAugmentedNGram.

ii. If mostProbableWord = negationWord proceed to step 11.b.i.1.b.ii.1. Otherwise skip all steps before step 11.c.
    1. Change the value of mostProbableWord to the most probable word that is not negationWord preceding main word from currentLeftAugmentedNGram.

iii. If the main word from chosen augmented n-gram = mostProbableWord proceed to step 11.b.i.1.b.iii.1. Otherwise skip all steps before step 11.c.

1. Change value of sentence to value of main word from selected augmented n-gram + " " + sentence.
2. Initialize variable previousLeftAugmentedNGram to the value of currentLeftAugmentedNGram.
3. Change the value of currentLeftAugmentedNGram to the value of the chosen augmented n-gram.
4. If main word from previousAugmentedNGram is preceded by NOTHING and main word from the currentLeftAugmentedNGram, change the value of isSentenceTerminatedOnLeft to TRUE.
5. Skip all steps before step 11.c.

2. If main word from chosen augmented n-gram precedes main word from currentLeftAugmentedNGram and both of those words in the above mentioned order precede main word from previousLeftAugmentedNGram proceed to step 11.b.i.2.a. Otherwise skip all steps before step 11.c.
   a. If isNegationNeeded = TRUE and isNegationIncludedInSentence = FALSE proceed to step 11.b.i.2.a.i. Otherwise skip all steps before step 11.b.i.2.b.
      i. If main word from chosen augmented n-gram = negationWord proceed to step 11.b.i.1.a.i. Otherwise skip all steps before step 11.c
         1. Change value of sentence to negationWord + " " + sentence.
         2. Change the value of isNegationIncludedInSentence to TRUE.
         3. Change the value of previousLeftAugmentedNGram to the value of currentLeftAugmentedNGram.
         4. Change the value of currentLeftAugmentedNGram to the value of the chosen augmented n-gram.
         5. If main word from previousLeftAugmentedNGram is preceded by NOTHING and main word from the currentLeftAugmentedNGram, change the value of isSentenceTerminatedOnLeft to TRUE.

6. Skip all steps before step 11.c.

ii. If main word from previousLeftAugmentedNGram is not preceded by negation word skip all steps before step 11.b.i.2.b.i. Otherwise skip all steps before step 11.c.

b. If isNegationNeeded = FALSE or isNegationIncludedInSentence = TRUE proceed to step 11.b.i.2.b.i. Otherwise skip all steps before step 11.c.

i. Initialize variable mostProbableWord to the value of most probable word preceding the main word from currentLeftAugmentedNGram and main word from previousLeftAugmentedNGram.

ii. If mostProbableWord = negationWord proced to step 11.b.i.2.b.ii.1. Otherwise skip all steps before step 11.b.i.2.b.iii.

1. Change the value of mostProbableWord to the most probable word that is not negationWord preceding the main word from currentLeftAugmentedNGram and main word from previousLeftAugmentedNGram.

iii. If the main word from chosen augmented n-gram = mostProbableWord proceed to step 11.b.i.2.b.iii.1. Otherwise skip all steps before step 11.c.

1. Change value of sentence to value of main word from selected augmented n-gram + " " + sentence.
2. Change the value of previousLeftAugmentedNGram to the value of the currentLeftAugmentedNGram.
3. Change the value of currentLeftAugmentedNGram to the value of the chosen augmented n-gram.
4. If main word from previousLeftAugmentedNGram is preceded by main word from currentLeftAugmentedNGram and NOTHING, change the value of isSentenceTerminatedOnLeft to TRUE.

c. If isSentenceTerminatedOnRight = FALSE go to step 11.c.i. Otherwise skip all steps before step 11.d.

i. If variable previousRightAugmentedNGram is not initialized go to step 11.c.i.1. Otherwise skip all steps before step 11.c.i.2.
    1. If main word from chosen augmented n-gram follows main word from currentRightAugmentedNGram proceed to step 11.c.i.1.a. Otherwise skip all steps before step 11.d.
        a. If isNegationNeeded = TRUE and isNegationIncludedInSentence = FALSE proceed to step 11.c.i.1.a.i. Otherwise skip all steps before step 11.d.
            i. If main word from chosen augmented n-gram = negationWord proceed to step 11.c.i.1.a.i.1. Otherwise skip all steps before step 11.c.i.1.a.ii.
                1. Change value of sentence to sentence + " " + negationWord.
                2. Change the value of isNegationIncludedInSentence to TRUE.
                3. Initialize the value of previousRightAugmentedNGram to value of currentRightAugmentedNGram.
                4. Change the value of currentRightAugmentedNGram to the value of the chosen augmented n-gram.
                5. If main word from previousRightAugmentedNGram is followed by main word from the currentRightAugmentedNGram and NOTHING, change the value of isSentenceTerminatedOnRight to TRUE.
                6. Skip all steps before step 11.d.
            ii. If main word from currentRightAugmentedNGram is not followed by negation word skip all steps before step 11.c.i.1.b.i. Otherwise skip all steps before step 11.d.
        b. If isNegationNeeded = FALSE or isNegationIncludedInSentence = TRUE proceed to step 11.c.i.1.b.i. Otherwise skip all steps before step 11.d.
            i. Initialize variable mostProbableWord to the value of most probable word following the main word from currentRightAugmentedNGram.

ii. If mostProbableWord = negationWord proceed to step 11.c.i.1.b.ii.1. Otherwise skip all steps before step 11.d.
    1. Change the value of mostProbableWord to the most probable word that is not negationWord following main word from currentRightAugmentedNGram.
iii. If the main word from chosen augmented n-gram = mostProbableWord proceed to step 11.c.i.1.b.iii.1. Otherwise skip all steps before step 11.d.
    1. Change value of sentence to sentence + " " + value of main word from selected augmented n-gram.
    2. Initialize variable previousRightAugmentedNGram to the value of currentRightAugmentedNGram.
    3. Change the value of currentRightAugmentedNGram to the value of the chosen augmented n-gram.
    4. If main word from previousRightAugmentedNGram is followed by main word from the currentLeftAugmentedNGram and NOTHING, change the value of isSentenceTerminatedOnRight to TRUE.
    5. Skip all steps before step 11.d.

2. If main word from chosen augmented n-gram follows main word from currentRightAugmentedNGram and both of those words in the reverse of the above mentioned order follow main word from previousRightAugmentedNGram proceed to step 11.c.i.2.a. Otherwise skip all steps before step 11.d.
    a. If isNegationNeeded = TRUE and isNegationIncludedInSentence = FALSE proceed to step 11.c.i.2.a.i. Otherwise skip all steps before step 11.c.i.2.b.
        i. If main word from chosen augmented n-gram = negationWord proceed to step 11.c.i.1.a.i.1. Otherwise skip all steps before step 11.d
            1. Change value of sentence to sentence + " " + negationWord.

2. Change the value of isNegationIncludedInSentence to TRUE.
3. Change the value of previousRightAugmentedNGram to the value of currentRightAugmentedNGram.
4. Change the value of currentRightAugmentedNGram to the value of the chosen augmented n-gram.
5. If main word from previousRightAugmentedNGram is followed by main word from the currentLeftAugmentedNGram and NOTHING, change the value of isSentenceTerminatedOnRight to TRUE.
6. Skip all steps before step 11.d.

ii. If main word from previousRightAugmentedNGram is not followed by negation word skip all steps before step 11.c.i.2.b.i. Otherwise skip all steps before step 11.d.

b. If isNegationNeeded = FALSE or isNegationIncludedInSentence = TRUE proceed to step 11.c.i.2.b.i. Otherwise skip all steps before step 11.d.

i. Initialize variable mostProbableWord to the value of most probable word following the main word from currentRightAugmentedNGram and main word from previousRightAugmentedNGram.

ii. If mostProbableWord = negationWord proced to step 11.c.i.2.b.ii.1. Otherwise skip all steps before step 11.c.i.2.b.iii.

1. Change the value of mostProbableWord to the most probable word that is not negationWord following the main word from previousRightAugmentedNGram and main word from currentRightAugmentedNGram.

iii. If the main word from chosen augmented n-gram = mostProbableWord proceed to step 11.c.i.2.b.iii.1. Otherwise skip all steps before step 11.d.

1. Change value of sentence to value of sentence + " " + main word from selected augmented n-gram.
2. Change the value of previousRightAugmentedNGram to the value of currentRightAugmentedNGram.
3. Change the value of currentRightAugmentedNGram to the value of the chosen augmented n-gram.
4. If main word from previousRightAugmentedNGram is followed by main word from currentRightAugmentedNGram and NOTHING, change the value of isSentenceTerminatedOnRight to TRUE.

d. If there are more n-grams to choose from from step 11.a skip all steps before step 11.e. Otherwise proceed to step 11.d.i.
   i. If the sentence from step 11 has not been expanded by words from steps 11.b and 11.c set the value of isSentenceValid to FALSE and exit the algorithm. Otherwise jump to step 11.

e. If isSentenceTerminatedOnLeft = TRUE and isSentenceTerminatedOnRight = TRUE go to step 12. Otherwise skip all steps and jump to step 11.a.

12. If sentence has 2 placeholders for point names, replace them with respective point names. Otherwise set the value of isSentenceValid to FALSE and skip all remaining steps.
13. If IsNegationNeeded = TRUE and isNegationIncludedInSentence = FALSE, set the value of isSentenceValid to FALSE.

Figure 39. Probabilistic augmented n-gram algorithm for creating sentences. This algorithm is run inside two loops that choose point names, and augmented n-grams whose main word type is RELATION**. If the value of the isSentenceValid variable remains** TRUE **after the run of the above algorithm, the sentence is given to further processing in those loops.**

A is left of B.
A is not right of B.
A is not in same column as B.
A is not above B.
A is not below B.
A is in same row as B.
A does not touch B.
A is not near to B.
A is not far from B.
A is left of C.
A is not right of C.
A is not in same column as C.
A is not above C.
A is below C.
C is near to B.
C is not far from B.
C is not left of D.
C is right of D.
C is not in same column as D.
C is not above D.
C is not below D.
C is in same row as D.
C does not touch D.
C is not near to D.
C is not far from D.
C is left of E.
C is not right of E.
C is not in same column as E.

A is not in same row as C.
A touches C.
A touches C on the corner.
A is near to C.
A is not far from C.
A is not left of D.
A is right of D.
A is not in same column as D.
A is not above D.
A is below D.
A is not in same row as D.
A touches D.
A touches D on the corner.
A is near to D.
A is not far from D.
A is left of E.
A is not right of E.
A is not in same column as E.
A is above E.
A is not below E.
A is not in same row as E.
A does not touch E.
A is not near to E.
A is not far from E.
B is not left of A.
B is right of A.
B is not in same column as A.
B is not above A.
B is not below A.
B is in same row as A.
B does not touch A.
B is not near to A.
B is not far from A.
B is not left of C.
B is right of C.
B is not in same column as C.
B is not above C.
B is below C.
B is not in same row as C.
B touches C.
B touches C on the corner.
B is near to C.
B is not far from C.
B is not left of D.
B is right of D.
B is not in same column as D.
B is not above D.
B is below D.
B is not in same row as D.
C is above E.
C is not below E.
C is not in same row as E.
C does not touch E.
C is not near to E.
C is not far from E.
D is left of A.
D is not right of A.
D is not in same column as A.
D is above A.
D is not below A.
D is not in same row as A.
D touches A.
D touches A on the corner.
D is near to A.
D is not far from A.
D is left of B.
D is not right of B.
D is not in same column as B.
D is above B.
D is not below B.
D is not in same row as B.
D does not touch B.
D is not near to B.
D is not far from B.
D is left of C.
D is not right of C.
D is not in same column as C.
D is not above C.
D is not below C.
D is in same row as C.
D does not touch C.
D is not near to C.
D is not far from C.
D is left of E.
D is not right of E.
D is not in same column as E.
D is above E.
D is not below E.
D is not in same row as E.
D does not touch E.
D is not near to E.
D is not far from E.
E is not left of A.
E is right of A.
E is not in same column as A.
E is not above A.
E is below A.
E is not in same row as A.

B does not touch D.
B is not near to D.
B is not far from D.
B is left of E.
B is not right of E.
B is not in same column as E.
B is above E.
B is not below E.
B is not in same row as E.
B does not touch E.
B is not near to E.
B is not far from E.
C is not left of A.
C is right of A.
C is not in same column as A.
C is above A.
C is not below A.
C is not in same row as A.
C touches A.
C touches A on the corner.
C is near to A.
C is not far from A.
C is left of B.
C is not right of B.
C is not in same column as B.
C is above B.
C is not below B.
C is not in same row as B.
C touches B.
C touches B on the corner.

E does not touch A.
E is not near to A.
E is not far from A.
E is not left of B.
E is right of B.
E is not in same column as B.
E is not above B.
E is below B.
E is not in same row as B.
E does not touch B.
E is not near to B.
E is not far from B.
E is not left of C.
E is right of C.
E is not in same column as C.
E is not above C.
E is below C.
E is not in same row as C.
E does not touch C.
E is not near to C.
E is not far from C.
E is not left of D.
E is right of D.
E is not in same column as D.
E is not above D.
E is below D.
E is not in same row as D.
E does not touch D.
E is not near to D.
E is not far from D.

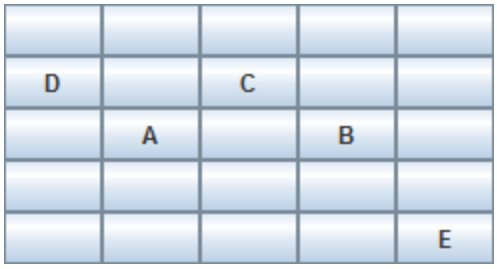


Figure 40. Sentences describing grid shown below sentences. In these above sentences there are 10 sentences for relation "left," 10 for "not left," 10 for "right," 10 for "not right," 0 for "in same column," 20 for "not in same column," 8 for "above," 12 for "not above," 8 for "below," 12 for "not below," 4 for "in same row," 16 for "not in same row," 6 for "touches," 14 for "does not touch," 6 for "touches on the corner," 0 for "does not touch on the corner," 6 for "near," 14 for "not near," 0 for "far," and 20 for "not far."

A is not left of B.
A is not right of B.
A is in same column as B.
A is not above B.

E is below J.
E is not in same row as J.
E does not touch J.
E is not near to J.

A is not below B.
A is in same row as B.
A does not touch B.
A is near to B.
A is left of C.
A is not right of C.
A is not in same column as C.
A is not above C.
A is not below C.
A is in same row as C.
A touches C.
A does not touch C on the corner.
A is not near to C.
A is left of D.
A is not right of D.
A is not in same column as D.
A is above D.
A is not below D.
A is not in same row as D.
A touches D.
A touches D on the corner.
A is not near to D.
A is not left of E.
A is not right of E.
A is in same column as E.
A is above E.
A is not below E.
A is not in same row as E.
A touches E.
A does not touch E on the corner.
A is not near to E.
A is not left of F.
A is right of F.
A is not in same column as F.
A is above F.
A is not below F.
A is not in same row as F.
A touches F.
A touches F on the corner.
A is not near to F.
A is not left of G.
A is right of G.
A is not in same column as G.
A is not above G.
A is not below G.
A is in same row as G.
A touches G.
A does not touch G on the corner.
A is not near to G.

F is left of A.
F is not right of A.
F is not in same column as A.
F is not above A.
F is below A.
F is not in same row as A.
F touches A.
F touches A on the corner.
F is not near to A.
F is left of B.
F is not right of B.
F is not in same column as B.
F is not above B.
F is below B.
F is not in same row as B.
F touches B.
F touches B on the corner.
F is not near to B.
F is left of C.
F is not right of C.
F is not in same column as C.
F is not above C.
F is below C.
F is not in same row as C.
F does not touch C.
F is not near to C.
F is left of D.
F is not right of D.
F is not in same column as D.
F is not above D.
F is not below D.
F is in same row as D.
F does not touch D.
F is not near to D.
F is left of E.
F is not right of E.
F is not in same column as E.
F is not above E.
F is not below E.
F is in same row as E.
F touches E.
F does not touch E on the corner.
F is not near to E.
F is not left of G.
F is not right of G.
F is in same column as G.
F is not above G.
F is below G.
F is not in same row as G.

A is not left of H.
A is right of H.
A is not in same column as H.
A is not above H.
A is below H.
A is not in same row as H.
A touches H.
A touches H on the corner.
A is not near to H.
A is not left of I.
A is not right of I.
A is in same column as I.
A is not above I.
A is below I.
A is not in same row as I.
A touches I.
A does not touch I on the corner.
A is not near to I.
A is left of J.
A is not right of J.
A is not in same column as J.
A is not above J.
A is below J.
A is not in same row as J.
A touches J.
A touches J on the corner.
A is not near to J.
B is not left of A.
B is not right of A.
B is in same column as A.
B is not above A.
B is not below A.
B is in same row as A.
B does not touch A.
B is near to A.
B is left of C.
B is not right of C.
B is not in same column as C.
B is not above C.
B is not below C.
B is in same row as C.
B touches C.
B does not touch C on the corner.
B is not near to C.
B is left of D.
B is not right of D.
B is not in same column as D.
B is above D.
B is not below D.
F touches G.
F does not touch G on the corner.
F is not near to G.
F is not left of H.
F is not right of H.
F is in same column as H.
F is not above H.
F is below H.
F is not in same row as H.
F does not touch H.
F is not near to H.
F is left of I.
F is not right of I.
F is not in same column as I.
F is not above I.
F is below I.
F is not in same row as I.
F does not touch I.
F is not near to I.
F is left of J.
F is not right of J.
F is not in same column as J.
F is not above J.
F is below J.
F is not in same row as J.
F does not touch J.
F is not near to J.
G is left of A.
G is not right of A.
G is not in same column as A.
G is not above A.
G is not below A.
G is in same row as A.
G touches A.
G does not touch A on the corner.
G is not near to A.
G is left of B.
G is not right of B.
G is not in same column as B.
G is not above B.
G is not below B.
G is in same row as B.
G touches B.
G does not touch B on the corner.
G is not near to B.
G is left of C.
G is not right of C.
G is not in same column as C.
G is not above C.

B is not in same row as D.
B touches D.
B touches D on the corner.
B is not near to D.
B is not left of E.
B is not right of E.
B is in same column as E.
B is above E.
B is not below E.
B is not in same row as E.
B touches E.
B does not touch E on the corner.
B is not near to E.
B is not left of F.
B is right of F.
B is not in same column as F.
B is above F.
B is not below F.
B is not in same row as F.
B touches F.
B touches F on the corner.
B is not near to F.
B is not left of G.
B is right of G.
B is not in same column as G.
B is not above G.
B is not below G.
B is in same row as G.
B touches G.
B does not touch G on the corner.
B is not near to G.
B is not left of H.
B is right of H.
B is not in same column as H.
B is not above H.
B is below H.
B is not in same row as H.
B touches H.
B touches H on the corner.
B is not near to H.
B is not left of I.
B is not right of I.
B is in same column as I.
B is not above I.
B is below I.
B is not in same row as I.
B touches I.
B does not touch I on the corner.
B is not near to I.

G is not below C.
G is in same row as C.
G does not touch C.
G is not near to C.
G is left of D.
G is not right of D.
G is not in same column as D.
G is above D.
G is not below D.
G is not in same row as D.
G does not touch D.
G is not near to D.
G is left of E.
G is not right of E.
G is not in same column as E.
G is above E.
G is not below E.
G is not in same row as E.
G touches E.
G touches E on the corner.
G is not near to E.
G is not left of F.
G is not right of F.
G is in same column as F.
G is above F.
G is not below F.
G is not in same row as F.
G touches F.
G does not touch F on the corner.
G is not near to F.
G is not left of H.
G is not right of H.
G is in same column as H.
G is not above H.
G is below H.
G is not in same row as H.
G touches H.
G does not touch H on the corner.
G is not near to H.
G is left of I.
G is not right of I.
G is not in same column as I.
G is not above I.
G is below I.
G is not in same row as I.
G touches I.
G touches I on the corner.
G is not near to I.
G is left of J.

B is left of J.
B is not right of J.
B is not in same column as J.
B is not above J.
B is below J.
B is not in same row as J.
B touches J.
B touches J on the corner.
B is not near to J.
C is not left of A.
C is right of A.
C is not in same column as A.
C is not above A.
C is not below A.
C is in same row as A.
C touches A.
C does not touch A on the corner.
C is not near to A.
C is not left of B.
C is right of B.
C is not in same column as B.
C is not above B.
C is not below B.
C is in same row as B.
C touches B.
C does not touch B on the corner.
C is not near to B.
C is not left of D.
C is not right of D.
C is in same column as D.
C is above D.
C is not below D.
C is not in same row as D.
C touches D.
C does not touch D on the corner.
C is not near to D.
C is not left of E.
C is right of E.
C is not in same column as E.
C is above E.
C is not below E.
C is not in same row as E.
C touches E.
C touches E on the corner.
C is not near to E.
C is not left of F.
C is right of F.
C is not in same column as F.
C is above F.
G is not right of J.
G is not in same column as J.
G is not above J.
G is below J.
G is not in same row as J.
G does not touch J.
G is not near to J.
H is left of A.
H is not right of A.
H is not in same column as A.
H is above A.
H is not below A.
H is not in same row as A.
H touches A.
H touches A on the corner.
H is not near to A.
H is left of B.
H is not right of B.
H is not in same column as B.
H is above B.
H is not below B.
H is not in same row as B.
H touches B.
H touches B on the corner.
H is not near to B.
H is left of C.
H is not right of C.
H is not in same column as C.
H is above C.
H is not below C.
H is not in same row as C.
H does not touch C.
H is left of D.
H is not right of D.
H is not in same column as D.
H is above D.
H is not below D.
H is not in same row as D.
H does not touch D.
H is not near to D.
H is left of E.
H is not right of E.
H is not in same column as E.
H is above E.
H is not below E.
H is not in same row as E.
H does not touch E.
H is not near to E.
H is not left of F.

C is not below F.
C is not in same row as F.
C does not touch F.
C is not near to F.
C is not left of G.
C is right of G.
C is not in same column as G.
C is not above G.
C is not below G.
C is in same row as G.
C does not touch G.
C is not near to G.
C is not left of H.
C is right of H.
C is not in same column as H.
C is not above H.
C is below H.
C is not in same row as H.
C does not touch H.
C is not near to H.
C is not left of I.
C is right of I.
C is not in same column as I.
C is not above I.
C is below I.
C is not in same row as I.
C touches I.
C touches I on the corner.
C is not near to I.
C is not left of J.
C is not right of J.
C is in same column as J.
C is not above J.
C is below J.
C is not in same row as J.
C touches J.
C does not touch J on the corner.
C is not near to J.
D is not left of A.
D is right of A.
D is not in same column as A.
D is not above A.
D is below A.
D is not in same row as A.
D touches A.
D touches A on the corner.
D is not near to A.
D is not left of B.
D is right of B.
H is not right of F.
H is in same column as F.
H is above F.
H is not below F.
H is not in same row as F.
H does not touch F.
H is not near to F.
H is not left of G.
H is not right of G.
H is in same column as G.
H is above G.
H is not below G.
H is not in same row as G.
H touches G.
H does not touch G on the corner.
H is not near to G.
H is left of I.
H is not right of I.
H is not in same column as I.
H is not above I.
H is not below I.
H is in same row as I.
H touches I.
H does not touch I on the corner.
H is not near to I.
H is left of J.
H is not right of J.
H is not in same column as J.
H is not above J.
H is not below J.
H is in same row as J.
H does not touch J.
H is not near to J.
I is not left of A.
I is not right of A.
I is in same column as A.
I is above A.
I is not below A.
I is not in same row as A.
I touches A.
I does not touch A on the corner.
I is not near to A.
I is not left of B.
I is not right of B.
I is in same column as B.
I is above B.
I is not below B.
I is not in same row as B.
I touches B.

D is not in same column as B.
D is not above B.
D is below B.
D is not in same row as B.
D touches B.
D touches B on the corner.
D is not near to B.
D is not left of C.
D is not right of C.
D is in same column as C.
D is not above C.
D is below C.
D is not in same row as C.
D touches C.
D does not touch C on the corner.
D is not near to C.
D is not left of E.
D is right of E.
D is not in same column as E.
D is not above E.
D is not below E.
D is in same row as E.
D touches E.
D does not touch E on the corner.
D is not near to E.
D is not left of F.
D is right of F.
D is not in same column as F.
D is not above F.
D is not below F.
D is in same row as F.
D does not touch F.
D is not near to F.
D is not left of G.
D is right of G.
D is not in same column as G.
D is not above G.
D is below G.
D is not in same row as G.
D does not touch G.
D is not near to G.
D is not left of H.
D is right of H.
D is not in same column as H.
D is not above H.
D is below H.
D is not in same row as H.
D does not touch H.
D is not near to H.

I does not touch B on the corner.
I is not near to B.
I is left of C.
I is not right of C.
I is not in same column as C.
I is above C.
I is not below C.
I is not in same row as C.
I touches C.
I touches C on the corner.
I is not near to C.
I is left of D.
I is not right of D.
I is not in same column as D.
I is above D.
I is not below D.
I is not in same row as D.
I does not touch D.
I is not near to D.
I is not left of E.
I is not right of E.
I is in same column as E.
I is above E.
I is not below E.
I is not in same row as E.
I does not touch E.
I is not near to E.
I is not left of F.
I is right of F.
I is not in same column as F.
I is above F.
I is not below F.
I is not in same row as F.
I does not touch F.
I is not near to F.
I is not left of G.
I is right of G.
I is not in same column as G.
I is above G.
I is not below G.
I is not in same row as G.
I touches G.
I touches G on the corner.
I is not near to G.
I is not left of H.
I is right of H.
I is not in same column as H.
I is not above H.
I is not below H.

D is not left of I.
D is right of I.
D is not in same column as I.
D is not above I.
D is below I.
D is not in same row as I.
D does not touch I.
D is not near to I.
D is not left of J.
D is not right of J.
D is in same column as J.
D is not above J.
D is below J.
D is not in same row as J.
D does not touch J.
D is not near to J.
E is not left of A.
E is not right of A.
E is in same column as A.
E is not above A.
E is below A.
E is not in same row as A.
E touches A.
E does not touch A on the corner.
E is not near to A.
E is not left of B.
E is not right of B.
E is in same column as B.
E is not above B.
E is below B.
E is not in same row as B.
E touches B.
E does not touch B on the corner.
E is not near to B.
E is left of C.
E is not right of C.
E is not in same column as C.
E is not above C.
E is below C.
E is not in same row as C.
E touches C.
E touches C on the corner.
E is not near to C.
E is left of D.
E is not right of D.
E is not in same column as D.
E is not above D.
E is not below D.
E is in same row as D.
I is in same row as H.
I touches H.
I does not touch H on the corner.
I is not near to H.
I is left of J.
I is not right of J.
I is not in same column as J.
I is not above J.
I is not below J.
I is in same row as J.
I touches J.
I does not touch J on the corner.
I is not near to J.
J is not left of A.
J is right of A.
J is not in same column as A.
J is above A.
J is not below A.
J is not in same row as A.
J touches A.
J touches A on the corner.
J is not near to A.
J is not left of B.
J is right of B.
J is not in same column as B.
J is above B.
J is not below B.
J is not in same row as B.
J touches B.
J touches B on the corner.
J is not near to B.
J is not left of C.
J is not right of C.
J is in same column as C.
J is above C.
J is not below C.
J is not in same row as C.
J touches C.
J does not touch C on the corner.
J is not near to C.
J is not left of D.
J is not right of D.
J is in same column as D.
J is above D.
J is not below D.
J is not in same row as D.
J does not touch D.
J is not near to D.
J is not left of E.

E touches D.
E does not touch D on the corner.
E is not near to D.
E is not left of F.
E is right of F.
E is not in same column as F.
E is not above F.
E is not below F.
E is in same row as F.
E touches F.
E does not touch F on the corner.
E is not near to F.
E is not left of G.
E is right of G.
E is not in same column as G.
E is not above G.
E is below G.
E is not in same row as G.
E touches G.
E touches G on the corner.
E is not near to G.
E is not left of H.
E is right of H.
E is not in same column as H.
E is not above H.
E is below H.
E is not in same row as H.
E does not touch H.
E is not near to H.
E is not left of I.
E is not right of I.
E is in same column as I.
E is not above I.
E is below I.
E is not in same row as I.
E does not touch I.
E is not near to I.
E is left of J.
E is not right of J.
E is not in same column as J.
E is not above J.

J is right of E.
J is not in same column as E.
J is above E.
J is not below E.
J is not in same row as E.
J does not touch E.
J is not near to E.
J is not left of F.
J is right of F.
J is not in same column as F.
J is above F.
J is not below F.
J is not in same row as F.
J does not touch F.
J is not near to F.
J is not left of G.
J is right of G.
J is not in same column as G.
J is above G.
J is not below G.
J is not in same row as G.
J does not touch G.
J is not near to G.
J is not left of H.
J is right of H.
J is not in same column as H.
J is not above H.
J is not below H.
J is in same row as H.
J does not touch H.
J is not near to H.
J is not left of I.
J is right of I.
J is not in same column as I.
J is not above I.
J is not below I.
J is in same row as I.
J touches I.
J does not touch I on the corner.
J is not near to I.

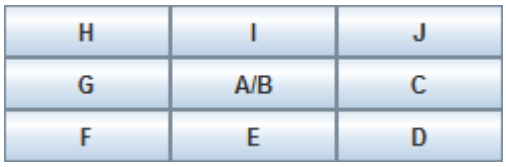


Figure 41. Sentences describing grid (shown below sentences). These sentences were used for almost all experiments for learning purposes only. In these above sentences there are 33 sentences for relation "`left`," 57 for "`not left`," 33 for "`right`," 57 for "`not right`," 24 for "`in same column`," 66 for "`not in same column`," 33 for

“above**,” 57 for “**not above**,” 33 for “**below**,” 57 for “**not below**,” 24 for “**in same row**,” 66 for “**not in same row**,” 56 for “**touches**,” 34 for “**does not touch**,” 24 for “**touches on the corner**,” 32 for “**does not touch on the corner**,” 2 for “**near**,” and 87 for “**not near**.”**

```
A not left of B.                    B does not in same row as E.
A is not the below B.               B does above F.
A does not touches C on the corner. B does right of G.
A is not near to the G.             C is not touches B on the corner.
A is not touch I on the corner.     C is not in the same column as E.
B does not above C.                 C does not below F.
```

Figure 42. Sentences that replace the respective sentences from Figure 41 to introduce incorrect grammar in the training set. These sentences from this figure should be used together with the sentences from Figure 41 that were not replaced.

```
A nie jest na lewo od B.                F jest na lewo od A.
A nie jest na prawo od B.               F nie jest na prawo od A.
A jest w tej samej kolumnie co B.       F nie jest w tej samej kolumnie co A.
A nie jest powyżej od B.                F nie jest powyżej od A.
A nie jest poniżej od B.                F jest poniżej od A.
A jest w tym samym rzędzie co B.        F nie jest w tym samym rzędzie co A.
A styka się z B.                        F styka się z A.
A nie styka się z B na rogu.            F styka się z A na rogu.
A jest na lewo od C.                    F jest na lewo od B.
A nie jest na prawo od C.               F nie jest na prawo od B.
A nie jest w tej samej kolumnie co C.   F nie jest w tej samej kolumnie co B.
A nie jest powyżej od C.                F nie jest powyżej od B.
A nie jest poniżej od C.                F jest poniżej od B.
A jest w tym samym rzędzie co C.        F nie jest w tym samym rzędzie co B.
A styka się z C.                        F styka się z B.
A styka się z C na rogu.                F styka się z B na rogu.
A jest na lewo od D.                    F jest na lewo od C.
A nie jest na prawo od D.               F nie jest na prawo od C.
A nie jest w tej samej kolumnie co D.   F nie jest w tej samej kolumnie co C.
A jest powyżej od D.                    F nie jest powyżej od C.
A nie jest poniżej od D.                F jest poniżej od C.
A nie jest w tym samym rzędzie co D.    F nie jest w tym samym rzędzie co C.
A styka się z D.                        F nie styka się z C.
A styka się z D na rogu.                F styka się z C na rogu.
A nie jest na lewo od E.                F jest na lewo od D.
A nie jest na prawo od E.               F nie jest na prawo od D.
A jest w tej samej kolumnie co E.       F nie jest w tej samej kolumnie co D.
A jest powyżej od E.                    F nie jest powyżej od D.
A nie jest poniżej od E.                F nie jest poniżej od D.
A nie jest w tym samym rzędzie co E.    F jest w tym samym rzędzie co D.
A styka się z E.                        F nie styka się z D.
A styka się z E na rogu.                F nie styka się z D na rogu.
A nie jest na lewo od F.                F jest na lewo od E.
A jest na prawo od F.                   F nie jest na prawo od E.
A nie jest w tej samej kolumnie co F.   F nie jest w tej samej kolumnie co E.
A jest powyżej od F.                    F nie jest powyżej od E.
A nie jest poniżej od F.                F nie jest poniżej od E.
```

A nie jest w tym samym rzędzie co F.
A styka się z F.
A styka się z F na rogu.
A nie jest na lewo od G.
A jest na prawo od G.
A nie jest w tej samej kolumnie co G.
A nie jest powyżej od G.
A nie jest poniżej od G.
A jest w tym samym rzędzie co G.
A styka się z G.
A styka się z G na rogu.
A nie jest na lewo od H.
A jest na prawo od H.
A nie jest w tej samej kolumnie co H.
A nie jest powyżej od H.
A jest poniżej od H.
A nie jest w tym samym rzędzie co H.
A styka się z H.
A styka się z H na rogu.
A nie jest na lewo od I.
A nie jest na prawo od I.
A jest w tej samej kolumnie co I.
A nie jest powyżej od I.
A jest poniżej od I.
A nie jest w tym samym rzędzie co I.
A styka się z I.
A styka się z I na rogu.
A jest na lewo od J.
A nie jest na prawo od J.
A nie jest w tej samej kolumnie co J.
A nie jest powyżej od J.
A jest poniżej od J.
A nie jest w tym samym rzędzie co J.
A styka się z J.
A styka się z J na rogu.
B nie jest na lewo od A.
B nie jest na prawo od A.
B jest w tej samej kolumnie co A.
B nie jest powyżej od A.
B nie jest poniżej od A.
B jest w tym samym rzędzie co A.
B styka się z A.
B nie styka się z A na rogu.
B jest na lewo od C.
B nie jest na prawo od C.
B nie jest w tej samej kolumnie co C.
B nie jest powyżej od C.
B nie jest poniżej od C.
B jest w tym samym rzędzie co C.

F jest w tym samym rzędzie co E.
F styka się z E.
F styka się z E na rogu.
F nie jest na lewo od G.
F nie jest na prawo od G.
F jest w tej samej kolumnie co G.
F nie jest powyżej od G.
F jest poniżej od G.
F nie jest w tym samym rzędzie co G.
F styka się z G.
F styka się z G na rogu.
F nie jest na lewo od H.
F nie jest na prawo od H.
F jest w tej samej kolumnie co H.
F nie jest powyżej od H.
F jest poniżej od H.
F nie jest w tym samym rzędzie co H.
F nie styka się z H.
F nie styka się z H na rogu.
F jest na lewo od I.
F nie jest na prawo od I.
F nie jest w tej samej kolumnie co I.
F nie jest powyżej od I.
F jest poniżej od I.
F nie jest w tym samym rzędzie co I.
F nie styka się z I.
F styka się z I na rogu.
F jest na lewo od J.
F nie jest na prawo od J.
F nie jest w tej samej kolumnie co J.
F nie jest powyżej od J.
F jest poniżej od J.
F nie jest w tym samym rzędzie co J.
F nie styka się z J.
F nie styka się z J na rogu.
G jest na lewo od A.
G nie jest na prawo od A.
G nie jest w tej samej kolumnie co A.
G nie jest powyżej od A.
G nie jest poniżej od A.
G jest w tym samym rzędzie co A.
G styka się z A.
G styka się z A na rogu.
G jest na lewo od B.
G nie jest na prawo od B.
G nie jest w tej samej kolumnie co B.
G nie jest powyżej od B.
G nie jest poniżej od B.
G jest w tym samym rzędzie co B.

B styka się z C.
B styka się z C na rogu.
B jest na lewo od D.
B nie jest na prawo od D.
B nie jest w tej samej kolumnie co D.
B jest powyżej od D.
B nie jest poniżej od D.
B nie jest w tym samym rzędzie co D.
B styka się z D.
B styka się z D na rogu.
B nie jest na lewo od E.
B nie jest na prawo od E.
B jest w tej samej kolumnie co E.
B jest powyżej od E.
B nie jest poniżej od E.
B nie jest w tym samym rzędzie co E.
B styka się z E.
B styka się z E na rogu.
B nie jest na lewo od F.
B jest na prawo od F.
B nie jest w tej samej kolumnie co F.
B jest powyżej od F.
B nie jest poniżej od F.
B nie jest w tym samym rzędzie co F.
B styka się z F.
B styka się z F na rogu.
B nie jest na lewo od G.
B jest na prawo od G.
B nie jest w tej samej kolumnie co G.
B nie jest powyżej od G.
B nie jest poniżej od G.
B jest w tym samym rzędzie co G.
B styka się z G.
B styka się z G na rogu.
B nie jest na lewo od H.
B jest na prawo od H.
B nie jest w tej samej kolumnie co H.
B nie jest powyżej od H.
B jest poniżej od H.
B nie jest w tym samym rzędzie co H.
B styka się z H.
B styka się z H na rogu.
B nie jest na lewo od I.
B nie jest na prawo od I.
B jest w tej samej kolumnie co I.
B nie jest powyżej od I.
B jest poniżej od I.
B nie jest w tym samym rzędzie co I.
B styka się z I.

G styka się z B.
G styka się z B na rogu.
G jest na lewo od C.
G nie jest na prawo od C.
G nie jest w tej samej kolumnie co C.
G nie jest powyżej od C.
G nie jest poniżej od C.
G jest w tym samym rzędzie co C.
G nie styka się z C.
G nie styka się z C na rogu.
G jest na lewo od D.
G nie jest na prawo od D.
G nie jest w tej samej kolumnie co D.
G jest powyżej od D.
G nie jest poniżej od D.
G nie jest w tym samym rzędzie co D.
G nie styka się z D.
G styka się z D na rogu.
G jest na lewo od E.
G nie jest na prawo od E.
G nie jest w tej samej kolumnie co E.
G jest powyżej od E.
G nie jest poniżej od E.
G nie jest w tym samym rzędzie co E.
G styka się z E.
G styka się z E na rogu.
G nie jest na lewo od F.
G nie jest na prawo od F.
G jest w tej samej kolumnie co F.
G jest powyżej od F.
G nie jest poniżej od F.
G nie jest w tym samym rzędzie co F.
G styka się z F.
G styka się z F na rogu.
G nie jest na lewo od H.
G nie jest na prawo od H.
G jest w tej samej kolumnie co H.
G nie jest powyżej od H.
G jest poniżej od H.
G nie jest w tym samym rzędzie co H.
G styka się z H.
G styka się z H na rogu.
G jest na lewo od I.
G nie jest na prawo od I.
G nie jest w tej samej kolumnie co I.
G nie jest powyżej od I.
G jest poniżej od I.
G nie jest w tym samym rzędzie co I.
G styka się z I.

B styka się z I na rogu.
B jest na lewo od J.
B nie jest na prawo od J.
B nie jest w tej samej kolumnie co J.
B nie jest powyżej od J.
B jest poniżej od J.
B nie jest w tym samym rzędzie co J.
B styka się z J.
B styka się z J na rogu.
C nie jest na lewo od A.
C jest na prawo od A.
C nie jest w tej samej kolumnie co A.
C nie jest powyżej od A.
C nie jest poniżej od A.
C jest w tym samym rzędzie co A.
C styka się z A.
C styka się z A na rogu.
C nie jest na lewo od B.
C jest na prawo od B.
C nie jest w tej samej kolumnie co B.
C nie jest powyżej od B.
C nie jest poniżej od B.
C jest w tym samym rzędzie co B.
C styka się z B.
C styka się z B na rogu.
C nie jest na lewo od D.
C nie jest na prawo od D.
C jest w tej samej kolumnie co D.
C jest powyżej od D.
C nie jest poniżej od D.
C nie jest w tym samym rzędzie co D.
C styka się z D.
C styka się z D na rogu.
C nie jest na lewo od E.
C jest na prawo od E.
C nie jest w tej samej kolumnie co E.
C jest powyżej od E.
C nie jest poniżej od E.
C nie jest w tym samym rzędzie co E.
C styka się z E.
C styka się z E na rogu.
C nie jest na lewo od F.
C jest na prawo od F.
C nie jest w tej samej kolumnie co F.
C jest powyżej od F.
C nie jest poniżej od F.
C nie jest w tym samym rzędzie co F.
C nie styka się z F.
C styka się z F na rogu.

G styka się z I na rogu.
G jest na lewo od J.
G nie jest na prawo od J.
G nie jest w tej samej kolumnie co J.
G nie jest powyżej od J.
G jest poniżej od J.
G nie jest w tym samym rzędzie co J.
G nie styka się z J.
G styka się z J na rogu.
H jest na lewo od A.
H nie jest na prawo od A.
H nie jest w tej samej kolumnie co A.
H jest powyżej od A.
H nie jest poniżej od A.
H nie jest w tym samym rzędzie co A.
H styka się z A.
H styka się z A na rogu.
H jest na lewo od B.
H nie jest na prawo od B.
H nie jest w tej samej kolumnie co B.
H jest powyżej od B.
H nie jest poniżej od B.
H nie jest w tym samym rzędzie co B.
H styka się z B.
H styka się z B na rogu.
H jest na lewo od C.
H nie jest na prawo od C.
H nie jest w tej samej kolumnie co C.
H jest powyżej od C.
H nie jest poniżej od C.
H nie jest w tym samym rzędzie co C.
H nie styka się z C.
H styka się z C na rogu.
H jest na lewo od D.
H nie jest na prawo od D.
H nie jest w tej samej kolumnie co D.
H jest powyżej od D.
H nie jest poniżej od D.
H nie jest w tym samym rzędzie co D.
H nie styka się z D.
H nie styka się z D na rogu.
H jest na lewo od E.
H nie jest na prawo od E.
H nie jest w tej samej kolumnie co E.
H jest powyżej od E.
H nie jest poniżej od E.
H nie jest w tym samym rzędzie co E.
H nie styka się z E.
H styka się z E na rogu.

C nie jest na lewo od G.
C jest na prawo od G.
C nie jest w tej samej kolumnie co G.
C nie jest powyżej od G.
C nie jest poniżej od G.
C jest w tym samym rzędzie co G.
C nie styka się z G.
C nie styka się z G na rogu.
C nie jest na lewo od H.
C jest na prawo od H.
C nie jest w tej samej kolumnie co H.
C nie jest powyżej od H.
C jest poniżej od H.
C nie jest w tym samym rzędzie co H.
C nie styka się z H.
C styka się z H na rogu.
C nie jest na lewo od I.
C jest na prawo od I.
C nie jest w tej samej kolumnie co I.
C nie jest powyżej od I.
C jest poniżej od I.
C nie jest w tym samym rzędzie co I.
C styka się z I.
C styka się z I na rogu.
C nie jest na lewo od J.
C nie jest na prawo od J.
C jest w tej samej kolumnie co J.
C nie jest powyżej od J.
C jest poniżej od J.
C nie jest w tym samym rzędzie co J.
C styka się z J.
C styka się z J na rogu.
D nie jest na lewo od A.
D jest na prawo od A.
D nie jest w tej samej kolumnie co A.
D nie jest powyżej od A.
D jest poniżej od A.
D nie jest w tym samym rzędzie co A.
D styka się z A.
D styka się z A na rogu.
D nie jest na lewo od B.
D jest na prawo od B.
D nie jest w tej samej kolumnie co B.
D nie jest powyżej od B.
D jest poniżej od B.
D nie jest w tym samym rzędzie co B.
D styka się z B.
D styka się z B na rogu.
D nie jest na lewo od C.

H nie jest na lewo od F.
H nie jest na prawo od F.
H jest w tej samej kolumnie co F.
H jest powyżej od F.
H nie jest poniżej od F.
H nie jest w tym samym rzędzie co F.
H nie styka się z F.
H nie styka się z F na rogu.
H nie jest na lewo od G.
H nie jest na prawo od G.
H jest w tej samej kolumnie co G.
H jest powyżej od G.
H nie jest poniżej od G.
H nie jest w tym samym rzędzie co G.
H styka się z G.
H styka się z G na rogu.
H jest na lewo od I.
H nie jest na prawo od I.
H nie jest w tej samej kolumnie co I.
H nie jest powyżej od I.
H nie jest poniżej od I.
H jest w tym samym rzędzie co I.
H styka się z I.
H styka się z I na rogu.
H jest na lewo od J.
H nie jest na prawo od J.
H nie jest w tej samej kolumnie co J.
H nie jest powyżej od J.
H nie jest poniżej od J.
H jest w tym samym rzędzie co J.
H nie styka się z J.
H nie styka się z J na rogu.
I nie jest na lewo od A.
I nie jest na prawo od A.
I jest w tej samej kolumnie co A.
I jest powyżej od A.
I nie jest poniżej od A.
I nie jest w tym samym rzędzie co A.
I styka się z A.
I styka się z A na rogu.
I nie jest na lewo od B.
I nie jest na prawo od B.
I jest w tej samej kolumnie co B.
I jest powyżej od B.
I nie jest poniżej od B.
I nie jest w tym samym rzędzie co B.
I styka się z B.
I styka się z B na rogu.
I jest na lewo od C.

D nie jest na prawo od C.	I nie jest na prawo od C.
D jest w tej samej kolumnie co C.	I nie jest w tej samej kolumnie co C.
D nie jest powyżej od C.	I jest powyżej od C.
D jest poniżej od C.	I nie jest poniżej od C.
D nie jest w tym samym rzędzie co C.	I nie jest w tym samym rzędzie co C.
D styka się z C.	I styka się z C.
D styka się z C na rogu.	I styka się z C na rogu.
D nie jest na lewo od E.	I jest na lewo od D.
D jest na prawo od E.	I nie jest na prawo od D.
D nie jest w tej samej kolumnie co E.	I nie jest w tej samej kolumnie co D.
D nie jest powyżej od E.	I jest powyżej od D.
D nie jest poniżej od E.	I nie jest poniżej od D.
D jest w tym samym rzędzie co E.	I nie jest w tym samym rzędzie co D.
D styka się z E.	I nie styka się z D.
D styka się z E na rogu.	I styka się z D na rogu.
D nie jest na lewo od F.	I nie jest na lewo od E.
D jest na prawo od F.	I nie jest na prawo od E.
D nie jest w tej samej kolumnie co F.	I jest w tej samej kolumnie co E.
D nie jest powyżej od F.	I jest powyżej od E.
D nie jest poniżej od F.	I nie jest poniżej od E.
D jest w tym samym rzędzie co F.	I nie jest w tym samym rzędzie co E.
D nie styka się z F.	I nie styka się z E.
D nie styka się z F na rogu.	I nie styka się z E na rogu.
D nie jest na lewo od G.	I nie jest na lewo od F.
D jest na prawo od G.	I jest na prawo od F.
D nie jest w tej samej kolumnie co G.	I nie jest w tej samej kolumnie co F.
D nie jest powyżej od G.	I jest powyżej od F.
D jest poniżej od G.	I nie jest poniżej od F.
D nie jest w tym samym rzędzie co G.	I nie jest w tym samym rzędzie co F.
D nie styka się z G.	I nie styka się z F.
D styka się z G na rogu.	I styka się z F na rogu.
D nie jest na lewo od H.	I nie jest na lewo od G.
D jest na prawo od H.	I jest na prawo od G.
D nie jest w tej samej kolumnie co H.	I nie jest w tej samej kolumnie co G.
D nie jest powyżej od H.	I jest powyżej od G.
D jest poniżej od H.	I nie jest poniżej od G.
D nie jest w tym samym rzędzie co H.	I nie jest w tym samym rzędzie co G.
D nie styka się z H.	I styka się z G.
D nie styka się z H na rogu.	I styka się z G na rogu.
D nie jest na lewo od I.	I nie jest na lewo od H.
D jest na prawo od I.	I jest na prawo od H.
D nie jest w tej samej kolumnie co I.	I nie jest w tej samej kolumnie co H.
D nie jest powyżej od I.	I nie jest powyżej od H.
D jest poniżej od I.	I nie jest poniżej od H.
D nie jest w tym samym rzędzie co I.	I jest w tym samym rzędzie co H.
D nie styka się z I.	I styka się z H.
D styka się z I na rogu.	I styka się z H na rogu.
D nie jest na lewo od J.	I jest na lewo od J.
D nie jest na prawo od J.	I nie jest na prawo od J.

D jest w tej samej kolumnie co J.
D nie jest powyżej od J.
D jest poniżej od J.
D nie jest w tym samym rzędzie co J.
D nie styka się z J.
D nie styka się z J na rogu.
E nie jest na lewo od A.
E nie jest na prawo od A.
E jest w tej samej kolumnie co A.
E nie jest powyżej od A.
E jest poniżej od A.
E nie jest w tym samym rzędzie co A.
E styka się z A.
E styka się z A na rogu.
E nie jest na lewo od B.
E nie jest na prawo od B.
E jest w tej samej kolumnie co B.
E nie jest powyżej od B.
E jest poniżej od B.
E nie jest w tym samym rzędzie co B.
E styka się z B.
E styka się z B na rogu.
E jest na lewo od C.
E nie jest na prawo od C.
E nie jest w tej samej kolumnie co C.
E nie jest powyżej od C.
E jest poniżej od C.
E nie jest w tym samym rzędzie co C.
E styka się z C.
E styka się z C na rogu.
E jest na lewo od D.
E nie jest na prawo od D.
E nie jest w tej samej kolumnie co D.
E nie jest powyżej od D.
E nie jest poniżej od D.
E jest w tym samym rzędzie co D.
E styka się z D.
E styka się z D na rogu.
E nie jest na lewo od F.
E jest na prawo od F.
E nie jest w tej samej kolumnie co F.
E nie jest powyżej od F.
E nie jest poniżej od F.
E jest w tym samym rzędzie co F.
E styka się z F.
E styka się z F na rogu.
E nie jest na lewo od G.
E jest na prawo od G.
E nie jest w tej samej kolumnie co G.

I nie jest w tej samej kolumnie co J.
I nie jest powyżej od J.
I nie jest poniżej od J.
I jest w tym samym rzędzie co J.
I styka się z J.
I styka się z J na rogu.
J nie jest na lewo od A.
J jest na prawo od A.
J nie jest w tej samej kolumnie co A.
J jest powyżej od A.
J nie jest poniżej od A.
J nie jest w tym samym rzędzie co A.
J styka się z A.
J styka się z A na rogu.
J nie jest na lewo od B.
J jest na prawo od B.
J nie jest w tej samej kolumnie co B.
J jest powyżej od B.
J nie jest poniżej od B.
J nie jest w tym samym rzędzie co B.
J styka się z B.
J styka się z B na rogu.
J nie jest na lewo od C.
J nie jest na prawo od C.
J jest w tej samej kolumnie co C.
J jest powyżej od C.
J nie jest poniżej od C.
J nie jest w tym samym rzędzie co C.
J styka się z C.
J styka się z C na rogu.
J nie jest na lewo od D.
J nie jest na prawo od D.
J jest w tej samej kolumnie co D.
J jest powyżej od D.
J nie jest poniżej od D.
J nie jest w tym samym rzędzie co D.
J nie styka się z D.
J nie styka się z D na rogu.
J nie jest na lewo od E.
J jest na prawo od E.
J nie jest w tej samej kolumnie co E.
J jest powyżej od E.
J nie jest poniżej od E.
J nie jest w tym samym rzędzie co E.
J nie styka się z E.
J styka się z E na rogu.
J nie jest na lewo od F.
J jest na prawo od F.
J nie jest w tej samej kolumnie co F.

E nie jest powyżej od G.
E jest poniżej od G.
E nie jest w tym samym rzędzie co G.
E styka się z G.
E styka się z G na rogu.
E nie jest na lewo od H.
E jest na prawo od H.
E nie jest w tej samej kolumnie co H.
E nie jest powyżej od H.
E jest poniżej od H.
E nie jest w tym samym rzędzie co H.
E nie styka się z H.
E styka się z H na rogu.
E nie jest na lewo od I.
E nie jest na prawo od I.
E jest w tej samej kolumnie co I.
E nie jest powyżej od I.
E jest poniżej od I.
E nie jest w tym samym rzędzie co I.
E nie styka się z I.
E nie styka się z I na rogu.
E jest na lewo od J.
E nie jest na prawo od J.
E nie jest w tej samej kolumnie co J.
E nie jest powyżej od J.
E jest poniżej od J.
E nie jest w tym samym rzędzie co J.
E nie styka się z J.
E styka się z J na rogu.

J jest powyżej od F.
J nie jest poniżej od F.
J nie jest w tym samym rzędzie co F.
J nie styka się z F.
J nie styka się z F na rogu.
J nie jest na lewo od G.
J jest na prawo od G.
J nie jest w tej samej kolumnie co G.
J jest powyżej od G.
J nie jest poniżej od G.
J nie jest w tym samym rzędzie co G.
J nie styka się z G.
J styka się z G na rogu.
J nie jest na lewo od H.
J jest na prawo od H.
J nie jest w tej samej kolumnie co H.
J nie jest powyżej od H.
J nie jest poniżej od H.
J jest w tym samym rzędzie co H.
J nie styka się z H.
J nie styka się z H na rogu.
J nie jest na lewo od I.
J jest na prawo od I.
J nie jest w tej samej kolumnie co I.
J nie jest powyżej od I.
J nie jest poniżej od I.
J jest w tym samym rzędzie co I.
J styka się z I.
J styka się z I na rogu.

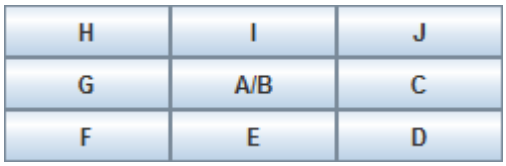


Figure 43. Sentences describing grid (shown below sentences). These sentences were used for almost all experiments for learning purposes only. In these above sentences there are 33 sentences for relation "na lewo**," 57 for "**nie na lewo**," 33 for "**na prawo**," 57 for "**nie na prawo**," 24 for "**w tej samej kolumnie**," 66 for "**nie w tej samej kolumnie**," 33 for "**powyżej**," 57 for "**nie powyżej**," 33 for "**poniżej**," 57 for "**nie poniżej**," 24 for "**w tym samym rzędzie**," 66 for "**nie w tym samym rzędzie**," 56 for "**styka się**," 34 for "**nie styka się**," 24 for "**styka się na rogu**," and 32 for "**nie styka się na rogu**."**

Sentences associated with simple relation LEFT:
[[A is left of B., A is left of C., A is left of E., B is left of E., C is left of B., C is left of E., D is left of A., D is left of B., D is left of C., D is left of E.]]

Sentences associated with negation of simple relation LEFT:
[[A is not left of D., B is not left of A., B is not left of C., B is not left of D., C is not left of A., C is not left of D., E is not left of A., E is not left of B., E is not left of C., E is not left of D.]]

Sentences associated with simple relation RIGHT:
[[A is right of D., B is right of A., B is right of C., B is right of D., C is right of A., C is right of D., E is right of A., E is right of B., E is right of C., E is right of D.]]

Sentences associated with negation of simple relation RIGHT:
[[A is not right of B., A is not right of C., A is not right of E., B is not right of E., C is not right of B., C is not right of E., D is not right of A., D is not right of B., D is not right of C., D is not right of E.]]

Sentences associated with simple relation SAME_COLUMN:
[]

Sentences associated with negation of simple relation SAME_COLUMN:
[[A is not in same column as B., A is not in same column as C., A is not in same column as D., A is not in same column as E., B is not in same column as A., B is not in same column as C., B is not in same column as D., B is not in same column as E., C is not in same column as A., C is not in same column as B., C is not in same column as D., C is not in same column as E., D is not in same column as A., D is not in same column as B., D is not in same column as C., D is not in same column as E., E is not in same column as A., E is not in same column as B., E is not in same column as C., E is not in same column as D.], [A is not above B., A is not above C., A is not above D., B is not above A., B is not above C., B is not above D., C is not above D., D is not above C., E is not above A., E is not above B., E is not above C., E is not above D.], [A is not below B., A is not below E., B is not below A., B is not below E., C is not below A., C is not below B., C is not below D., C is not below E., D is not below A., D is not below B., D is not below C., D is not below E.], [A does not touch B., A does not touch E., B does not touch A., B does not touch D., B does not touch E., C does not touch D., C does not touch E., D does not touch B., D does not touch C., D does not touch E., E does not touch A., E does not touch B., E does not touch C., E does not touch D.], [A is not in same row as C., A is not in same row as D., A is not in same row as E., B is not in same row as C., B is not in same row as D., B is not in same row as E., C is not in same row as A., C is not in same row as B., C is not in same row as E., D is not in same row as A., D is not in same row as B., D is not in same row as E., E is not in same row as A., E is not in same row as B., E is not in same row as C., E is not in same row as D.], [A touches C., A touches D., B touches C., C touches A., C touches B., D touches A.], [A touches C on the corner., A touches D on the corner., B touches C on the corner., C touches A on the corner., C touches B on the corner., D touches A on the corner.]]

```
Sentences associated with simple relation ABOVE:
[[A is above E., B is above E., C is above A., C is above B., C is above E.,
D is above A., D is above B., D is above E.]]

Sentences associated with negation of simple relation ABOVE:
[[A is below C., A is below D., B is below C., B is below D., E is below A.,
E is below B., E is below C., E is below D.]]

Sentences associated with simple relation BELOW:
[[A is below C., A is below D., B is below C., B is below D., E is below A.,
E is below B., E is below C., E is below D.]]

Sentences associated with negation of simple relation BELOW:
[[A is above E., B is above E., C is above A., C is above B., C is above E.,
D is above A., D is above B., D is above E.]]

Sentences associated with simple relation SAME_ROW:
[[A is in same row as B., B is in same row as A., C is in same row as D., D
is in same row as C.]]

Sentences associated with negation of simple relation SAME_ROW:
[[A is below C., A is below D., B is below C., B is below D., E is below A.,
E is below B., E is below C., E is below D.]]

Sentences associated with simple relation TOUCHING:
[[A is near to C., A is near to D., B is near to C., C is near to A., C is
near to B., D is near to A.]]

Sentences associated with negation of simple relation TOUCHING:
[[A is in same row as B., B is in same row as A., C is in same row as D., D
is in same row as C.]]
```

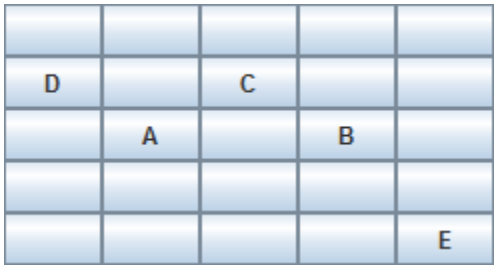


Figure 44. Results of assigning sentences describing grid shown below sentences to simple relations when not all sentences are present in the set such as "`in same column as`**."**

```
Sets of sentences true for machine-learning model whose class name is "is
left of:"
[[A is left of B., A is left of C., A is left of E., B is left of E., C is
left of B., C is left of E., D is left of A., D is left of B., D is left of
C., D is left of E.]]
```

Sets of sentences true for machine-learning model whose class name is "is not left of:"
[[A is right of D., B is right of A., B is right of C., B is right of D., C is right of A., C is right of D., E is right of A., E is right of B., E is right of C., E is right of D.]]

Sets of sentences true for machine-learning model whose class name is "is not right of:"
[[A is not right of B., A is not right of C., A is not right of E., B is not right of E., C is not right of B., C is not right of E., D is not right of A., D is not right of B., D is not right of C., D is not right of E.]]

Sets of sentences true for machine-learning model whose class name is "is not in same column as:"
[[A is not in same column as B., A is not in same column as C., A is not in same column as D., A is not in same column as E., B is not in same column as A., B is not in same column as C., B is not in same column as D., B is not in same column as E., C is not in same column as A., C is not in same column as B., C is not in same column as D., C is not in same column as E., D is not in same column as A., D is not in same column as B., D is not in same column as C., D is not in same column as E., E is not in same column as A., E is not in same column as B., E is not in same column as C., E is not in same column as D.], [A is not above B., A is not above C., A is not above D., B is not above A., B is not above C., B is not above D., C is not above D., D is not above C., E is not above A., E is not above B., E is not above C., E is not above D.], [A is not in same row as C., A is not in same row as D., A is not in same row as E., B is not in same row as C., B is not in same row as D., B is not in same row as E., C is not in same row as A., C is not in same row as B., C is not in same row as E., D is not in same row as A., D is not in same row as B., D is not in same row as E., E is not in same row as A., E is not in same row as B., E is not in same row as C., E is not in same row as D.], [A does not touch B., A does not touch E., B does not touch A., B does not touch D., B does not touch E., C does not touch D., C does not touch E., D does not touch B., D does not touch C., D does not touch E., E does not touch A., E does not touch B., E does not touch C., E does not touch D.], [A is not left of D., B is not left of A., B is not left of C., B is not left of D., C is not left of A., C is not left of D., E is not left of A., E is not left of B., E is not left of C., E is not left of D.], [A touches C., A touches D., B touches C., C touches A., C touches B., D touches A.], [A touches C on the corner., A touches D on the corner., B touches C on the corner., C touches A on the corner., C touches B on the corner., D touches A on the corner.], [A is not below B., A is not below E., B is not below A., B is not below E., C is not below A., C is not below B., C is not below D., C is not below E., D is not below A., D is not below B., D is not below C., D is not below E.]]

Sets of sentences true for machine-learning model whose class name is "is not above:"
[[A is in same row as B., B is in same row as A., C is in same row as D., D is in same row as C.]]

Sets of sentences true for machine-learning model whose class name is "is below:"
[[A is below C., A is below D., B is below C., B is below D., E is below A., E is below B., E is below C., E is below D.]]

Sets of sentences true for machine-learning model whose class name is "is not in same row as:"
[[A is below C., A is below D., B is below C., B is below D., E is below A., E is below B., E is below C., E is below D.]]

Sets of sentences true for machine-learning model whose class name is "does not touch:"
[[A is in same row as B., B is in same row as A., C is in same row as D., D is in same row as C.]]

Sets of sentences true for machine-learning model whose class name is "is not left of:"
[[A is right of D., B is right of A., B is right of C., B is right of D., C is right of A., C is right of D., E is right of A., E is right of B., E is right of C., E is right of D.]]

Sets of sentences true for machine-learning model whose class name is "is right of:"
[[A is right of D., B is right of A., B is right of C., B is right of D., C is right of A., C is right of D., E is right of A., E is right of B., E is right of C., E is right of D.]]

Sets of sentences true for machine-learning model whose class name is "touches:"
[[A is near to C., A is near to D., B is near to C., C is near to A., C is near to B., D is near to A.]]

Sets of sentences true for machine-learning model whose class name is "touches on the corner:"
[[A is near to C., A is near to D., B is near to C., C is near to A., C is near to B., D is near to A.]]

Sets of sentences true for machine-learning model whose class name is "is near to:"
[[A is near to C., A is near to D., B is near to C., C is near to A., C is near to B., D is near to A.]]
Sets of sentences true for machine-learning model whose class name is "is above:"
[[A is above E., B is above E., C is above A., C is above B., C is above E., D is above A., D is above B., D is above E.]]

Sets of sentences true for machine-learning model whose class name is "is not below:"

```
[[A is above E., B is above E., C is above A., C is above B., C is above E.,
D is above A., D is above B., D is above E.]]

Sets of sentences true for machine-learning model whose class name is "is in
same row as:"
[[A is in same row as B., B is in same row as A., C is in same row as D., D
is in same row as C.]]
```

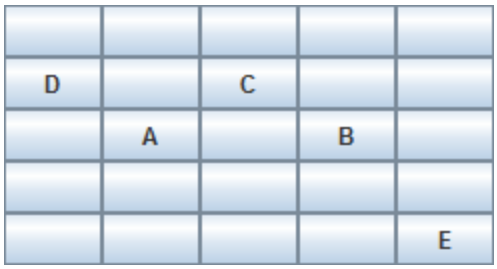


Figure 45. Sentences categorized by machine learning as true for a machine0learning model whose class name is relation text. The sentences for training are in Figure 40, and the grid below those sentences. Since the list of the sentences true for each machine learning model was too long, the output for each group was shortened automatically by the program.

Appendix B: License and Copyright

This license governs works based on the algorithms mentioned in this thesis and any derivatives of this thesis, and must appear on any derivatives of this thesis or with any fragments of this thesis.

This license grants you non-exclusive non-transferable right to use or implement the algorithms described in this thesis royalty-free if you use the algorithms or this work for non-commercial or academic purposes, with the exception for business conducted by Ryerson University, Toronto, Ontario, Canada.

It does not give you the permission to republish the work in any way, with the exception of publishing made by Ryerson University, Toronto, Ontario, Canada.

Commercial use and implementation of the algorithms mentioned in this thesis is allowed after the license fee of 50% of the profits made from sale of products based on the algorithms described in this thesis is paid to the author(s) of this and any derivative works that were used in creating the product.

The algorithms should be publicity available for viewing in electronic or paper format as part of the Ryerson University's Library collection.

The copyright of this or any derivative works is governed by the copyright laws of countries where the work or the derivatives are created or the algorithms are implemented and the copyright law of Canada.

If any term of this license is violated, the author(s) and/or the copyright holder(s) have the right to prosecute the offender(s) to the full extent allowed by laws of the countries for which the license is in effect.

References

[1] T. Patten, and P. Jacobs. Natural-Language Processing. *IEEE Expert*. 9(1):35, February 1994.

[2] Z. Harris. *Methods in Structural Linguistics*. Chicago: University of Chicago Press, 1951.

[3] C. B. Baker. *The Atoms of Language: The Mind's Hidden Rules of Grammar*. New York: Basic Books, 2001.

[4] P. Fromm, and P. Drews. *Natural Language Processing for Dynamic Environments*. Industrial Electronics Society. IECON '98. In *Proceedings of the 24th Annual Conference of the IEEE*. 4:2018-2021, August-September 1998.

[5] F. Siasar djahantighi, M. Norouzifrad, S. H. Davarpanah, and M. H. Shenassa. *Using Natural Language Processing in Order to Create SQL Queries*. International Conference on Computer and Communication Engineering, 2008. ICCCE 2008. pp. 600-604. May 2008. Kuala Lumpur.

[6] C. D. Manning and H. Shütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[7] M. A. Covington. *Natural Language Processing for Prolog Programmers*. Englewood Cliffs: Prentice Hall, 1994.

[8] J. Allen. *Natural Language Understanding*. Menlo Park: the Benjamin/Cummings Publishing Company, Inc. California, 1987.

[9] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson Education, Inc. New Jersey, 2009.

[10] L. Guohuan, Z. Hao, and W. Hongui. Research on Methods of Semantic Disambiguation about Natural Language Processing. *International Conference on Wireless Networks and Information Systems, 2009. WNIS '09*. pp. 347-349, December 2009.

[11] Y. Dang, Y. Zhang, and D. Zhang. Statistical parser for RNA secondary structure prediction. In *Proceedings of 2005 International Conference on Machine Learning and Cybernetics, 2005*. (6):3399-3403. August 2005. Guangzhou, China.

[12] H. Yamamoto, S. Isogai, and Y. Sagisaka. Multi-Class Composite N-Gram Language Model for Spoken Language Processing Using Multiple Word Clusters. In *Proceedings of*

the 39th Annual Meeting on Association for Computational Linguistics, Toulouse, France. Association for Computational Linguistics, Morristown, NJ, pp. 531-538, 2001.

[13] M. Meeter, and J. R. Rohlicek. Statistical Language Modeling Combining N-Gram and Context-Free Grammars. *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing, 1993*. (2):37-40. Troy, NY. April 1993.

[14] A. Moschitti, G. Riccardi, and C. Raymond. Spoken Language Understanding With Kernels for Syntatic/Semantic Structures. *IEEE Workshop on Automatic Speech Recognition and Understanding, 2007. ASRU*. 183-188. December 2007. Kyoto.

[15] F. Perraud, C. Viard-Gaudin, E. Morin, and P.-M. Lallican. N-Gram and N-Class Models for On Line Handwriting Recognition. *Seventh International Conference on Document Analysis and Recognition, 2003*. 1053-1057. August 2003.

[16] J. B. Mariño, R. E. Banchs, J. E. Crego, A. de Gispert, P. Lambert, J. A. R. Fonollosa, and M. R. Costa-jussà. N-Gram-Based Machine Translation. *Computational Linguistics*, 32(4):527-549, December 2006. MIT Press, Cambridge, MA.

[17] M. Khalilov, and J. A. R. Fonollosa. N-Gram-Based Statistical Machine Translation Versus Syntax Augmented Machine Translation: Comparison and System Combination. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, Athens, Greece, 2009, pp. 424-432. Association for Computational Linguistics, Morristown, NJ.

[18] O. White, T. Dunning, G. Sutton, M. Adams, J. Craig Venter, and C. Fields. A Quality Control Algorithm for DNA Sequencing Projects. *Nucleic Acids Research*, 21(16):3829-3838, 1993.

[19] I. Suzuki, Y. Mikami, A. Ohisato, and Y. Chubachi. A Language and Character Set Determination Method Based on N-Gram Statistics. *ACM Transactions on Asian Language Information Processing (TALIP)*, 1(3):269-278, September 2002. ACM, New York, NY.

[20] S. Kwon. A Parsing Algorithm for Korean Implemented in Prolog. In *Proceedings of the 44th Annual Southeast Regional Conference*, Melbourne, Florida, 2006, pp. 40-44. ACM, New York, NY.

[21] W. Wang, and M. P. Harper. A Statistical Constraint Dependency Grammar (CDG) Parser. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cog-*

nition Together, Barcelona, Spain, 2004, pp. 42-49. Association for Computational Linguistics, Morristown, NJ.

[22] J. Hammerton, M. Osborne, S. Armstrong, and W. Daelemans. Introduction to Special Issue on Machine Learning Approaches to Shallow Parsing. *The Journal of Machine Learning Research*. 2(SPECIAL ISSUE): 551-558, March 2002.

[23] B. Kouchnir. A Machine Learning Approach to German Pronoun Resolution. *Proceedings of the ACL 2004 workshop on Student research*. art. 31, Barcelona, Spain, 2004.

[24] O. Maimon, and L. Rokach. *The Data Mining and Knowledge Discovery Handbook*. Springer, New York, NY.

[25] L. Guo-Chang. Research on "Inaccurate Learning" and Its Countermeasure in Machine Learning. *Fourth International Conference on Natural Computation, 2008. ICNC '08*. (1):227-231. October 2008. Jinan.

[26] G. Ng. and E. Harley. *GridLearner,* Source code: © 2007, Ryerson University.

[27] K. P. Bennett, and E. Parrado-Hermández. The Interplay of Optimization and Machine Learning Research. *The Journal of Machine Learning Research*. 7:1265-1281, 2006.

[28] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. *The WEKA Data Mining Software: An Update*. SIGKDD Explorations. 11(1):10-18. November 2009. http://www.cs.waikato.ac.nz/ml/weka/

[29] R. Kamimura. Application of Temporal Supervised Learning Algorithm to Generation of Natural Language. *1990 IJCNN International Joint Conference on Neural Networks, 1990*. (1):201-207. June 1990. San Diego, CA.

[30] Y. Guo, J. van Genabith, and H. Wang. Dependency-Based N-Gram Models for General Purpose Sentence Realisation. In *Proceedings of the 22nd International Conference on Computational Linguistics*. (1):297-304. 2008. Manchester, United Kingdom.

[31] M. Dacke. Evidence for Counting in Insects. *Animal Cognition*, 11(4):683-689, October, 2008.

Glossary

Augmented n-gram	The data that is collected about a chosen word, it's role in sentence, and up to n of the surrounding words, the number of occurrences of the chosen word, and the number of occurrences of the chosen word together with the number of occurrences of the surrounding words. It is used to build sentences in SentenceLearner.
Bigram	An n-gram with 2 tokens.
Grid	Matrix of points that can be randomly distributed or edited by user to the required specifications. The algorithms have access to the grid to determine the simple relations.
GridLearner	Program that uses 19 by 19 grid with two points, and truth values to learn relation strings entered by user. The program does not parse the relation strings.
GUI	Graphical user interface.
Inner word	A word that is in the sequence: (main word, inner word) or (inner word, main word) in the augmented n-gram.
Model	A formal construct that stands for the state of affairs in the world that we are trying to represent. In this thesis, the world is a grid and sentences describing the grid.
N-gram	An n-token sequence of data of the same type. The tokens are usually words.
N-gram model	A way to predict the n^{th} token based on the previous $n-1$ tokens by using a probabilistic model.
Outer word	A word that is in the sequence: (main word, inner word, outer word) or (outer word, inner word, main word) in the augmented n-gram.
Probabilistic parser	An algorithm that predicts the most probable grammatical structure of a given sentence. That algorithm uses statistics of grammatical structures that are obtained from hand parsed sentences.
`RelationalData`	The collection of not conflicting simple relations that apply to two given points in a given grid. This collection includes horizontal simple relations,

	horizontal distance, vertical simple relations, vertical distance, is touching simple relation, and the dimensions of the given grid. `RelationalData` objects in our approach simulate innate concepts as they might be present in human brain.
`RelationText`	String that remains after the removal of point names from a sentence.
Point name	The one letter word representing point name. A capital letter from `A` to `Z`.
Phrase	Two or more words that must go together to convey a relation.
Relational verb	The verb which itself states a relation without the need for helper words. Examples are: touches, flies, barks.
`SentenceData`	Data structure that holds the original sentence text, the relation text, and the point names.
`SentenceLearner`	Combination of the parser and sentence creation algorithms used to describe a situation between two points.
Shallow parser	A probabilistic parser that does not extract complete information about grammatical structures of sentences from a given text. A shallow parser only identifies and extracts information from the words that are most likely to convey information.
Situation	The relation between pairs of points in a grid.
Trigram	An n-gram with 3 tokens.
Verb of being	The verb that introduces a relation. This verb by itself it does not state any relation. Example of the verb of being in English "`is`," in Polish "`jest`." The type for the verb of being is `VERB`.
Word type	The function that a word is having in the sentence. Valid types are `NEGATION`, `NOTHING`, `RELATION`, `RELATION_VERB`, `POINT_NAME`, `VERB`, and `UNKNOWN`. A word cannot have more than one type assigned to it. Empty spot in sliding window is marked as `NOTHING` to indicate no word in that position.