Agent behaviour recognition using text analysis

S. Sitanskiy, L. Sebastia, E. Onaindia[1]*
[1]Valencian Research Institute for Artificial Intelligence
Universitat Politècnica de València
stasiig@inf.upv.es; {lsebastia, onaindia}@dsic.upv.es

Abstract

Research on AI is focusing towards explainable technology, able to recognize the behaviour patterns of an application in order to make sensible and informed decisions, predictions or explaining selected choices. In this work, we are interested in recognizing the behaviour that a planning agent follows when solving a planning problem. In other words, our goal is to uncover the agent's reasons to select one action or another at a particular moment, beyond the action model that governs the physics of the domain.

We present in this paper a new recognition mechanism based on machine learning methods. The idea lies in regarding a plan as plain text instead of as a structure of action names and objects. This way, natural language processing techniques can be applied to extract the structure underlying a plan and, therefore, to recognize the behaviour followed by the agent.

We have performed some experiments with several feature extraction methods and several classification techniques for two widely known planning domains. The results show that this is a promising approach.

Introduction

Research on AI is more and more focusing towards explainable technology that accounts for the outcomes of programs and products. One important aspect in this direction is the ability to recognize the behaviour patterns of an application in order to make sensible and informed decisions, predictions or explaining selected choices. Another equally important aspect is the possibility of learning agents behavior to accelerate the construction of intelligent behaviours, or the investigation of certain fragments of the learned behaviour.

In a planning context, an agent is in charge of synthesizing a plan to solve a given planning problem, where a problem is defined by a description of the initial state of the world, a description of the desired goals, and a description of a set of possible actions. The actions represent the model that governs the physics of the domain; i.e., the model determines the legal actions that can be performed and under which conditions. However, the behaviour of a system is not only determined by the model but also by the reasons for choosing one action among many options or the order in which actions will be performed. Generally speaking, the model tells us what can be done, the behaviour tell us how this is done.

In this work, we are interested in recognizing the behaviour that a planning agent follows when solving a planning problem. Our previous work (Sitanskiy, Sebastia, and Onaindia 2020) describes a recognition mechanism based on plan distance metrics. However, this approach has some limitations, mainly related to the use of resources and the order in which goals are solved.

In order to overcome these difficulties, we present in this paper a new recognition mechanism based on machine learning methods. The idea lies in regarding a plan as plain text instead of as a structure of action names and objects. This way, classical machine learning methods that are currently widely used for text recognition can be applied to recognize the behaviour pattern underlying a plan.

This paper is organized as follows: next section introduces the notion of agent behaviour; then, the NLP techniques used in our approach are described and the details of our recognition process are presented; section Experiments shows the experiments we have performed in order to test our approach and we finish with some conclusions and future work.

Agent behaviour

A planning problem is defined as $\mathcal{P} = (\mathcal{I}, \mathcal{G}, \mathcal{A})$, where $\mathcal{I}$ and $\mathcal{G}$ are sets of fluents that described the initial state and the goals, respectively, and $\mathcal{A}$ is the set of actions that can be executed in the domain. Table 1 shows two plans executed by two planning agents when solving a problem of the *Logistics* domain, a classical planning domain introduced in the first International Planning Competition (IPC)[1]. This problem consists in transporting packages `P1` and `P2` from location `L1` to location `L2` with two available trucks, `T1` and `T2`. Both plans are executable in $\mathcal{I}$, and both reach the goals in $\mathcal{G}$ using the actions in $\mathcal{A}$.

However, both plans use different strategies (behaviours)

*This work is partially supported by the Spanish MINECO project TIN2017-88476-C2-1-R and the FPI grant PRE2018-083896.

[1]http://ipc98.icaps-conference.org/

Plan A	Plan B
LOAD P1 T1 L1	LOAD P1 T1 L1
LOAD P2 T1 L1	DRIVE T1 L1 L2
DRIVE T1 L1 L2	UNLOAD P1 T1 L2
UNLOAD P1 T1 L2	DRIVE T1 L2 L1
UNLOAD P2 T1 L2	LOAD P2 T1 L1
	DRIVE T1 L1 L2
	UNLOAD P2 T1 L2

Table 1: Two example plans, representing different behaviours: Plan A - load all of the packages in the truck before starting to unload and Plan B - transport all the packages one by one

to solve the problem. Plan A follows Behaviour 1 (which consists in loading all of the packages onto the truck before starting to unload them), whereas plan B follows Behaviour 2 (which consists in transporting the packages one by one).

The behaviour of an agent can be represented in several ways. The field of game computing has developed an extensive work in behaviour representation and reasoning. Most games require artificial opponents against which the player is competing and these artificial players should be credible opponents that demonstrate intelligent behaviour in the course of the game. Traditionally, *rule-based systems* and *finite-state machines* have been used to represent the behaviour of non-player characters (NPC) in computer games (Cavazza 2000). A rule-based system comprises several elements: (i) a rule formalism/syntax; (ii) a unification algorithm; and (iii) an inference engine. Behaviour-oriented rule-based systems, such as those used in simulation, tend to adopt forward chaining where traditional decision support systems would use either backward or forward chaining. On the other hand, NPC behaviour can be controlled by a finite-state transition network, in which state transitions are driven by percepts from the embodied agents. Percepts are calculations made within the game world from the agent's perspective. Actions to be carried out by NPC in the game environment are associated with these state transitions.

Recently, *behaviour trees* have been introduced to represent NPC behaviour. A behaviour tree (BT) is a model of plan execution that is graphically represented as a tree. A node in a tree either encapsulates an action to be performed or acts as a control flow component that directs traversal over the tree. Behavior trees are appropriate for specifying the behavior of non-player characters and other entities because of their maintainability, scalability, reusability, and extensibility (Marcotte and Hamilton 2017).

In (Sitanskiy, Sebastia, and Onaindia 2020), BTs are used to represent the behaviour of different planning agents in the Logistics domain. The BTs are used to synthesize a plan that solves a planning problem. The cited work compares a plan generated with the LAMA planner (Richter and Westphal 2010) for a particular behaviour encoded in the planning domain with the plans produced by the BTs. Various plan distance metrics are used so as to identify the underlying behaviour of the LAMA plan. The authors conclude that this approximation to behaviour recognition has some

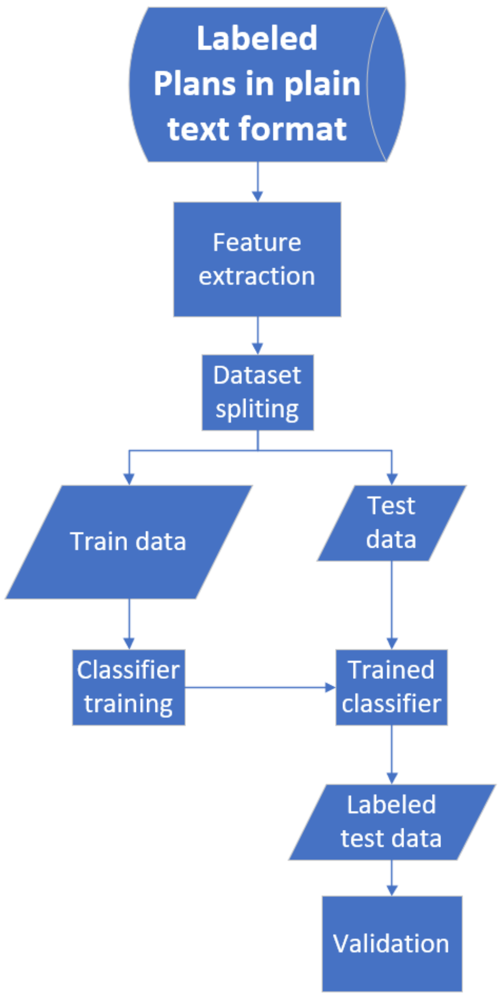

Figure 1: Recognition model building and validation process

limitations mainly due to the inability to discern among resources and the order in which goals must be solved.

In this paper, we analyze the behaviour recognition problem from a different perspective. In particular, each plan is regarded as a sequence of words, each of which has a certain semantic meaning, being some words action names while others represent objects in the problem. Our proposal does not build upon the meaning of the words but on the application of statistical methods to analyze the structure of the plan by using Natural Language Processing (NLP) techniques.

Figure 1 shows the recognition process. The input is a set of plans in plain format, labeled with the behaviour that they follow. Then, a feature extraction process assigns a feature vector to each plan. The result of the plan vectorization is used by a classifier to train a model that learns the different behaviours followed by each agent. The model will then be used at the recognition stage to determine the underlying behaviour of a given plan.

In the next section, we explain several vectorization techniques that have been used to extract plan features. Several classification methods have also been tested to analyze the one that returns the best accuracy.

Feature extraction

Text classification is a very common task in NLP. NLP technologies are widely used in areas such as speech recognition, natural language understanding, and natural language generation. Specifically, bag of words and n-grams models have been used for age and gender prediction and sentiment and mood analysis based on news in social networks (Peersman, Daelemans, and Van Vaerenbergh 2011) (Rodrigues et al. 2016) (Wang et al. 2014), checking e-mails for spam (Barushka and Hajek 2019), translation (Ma et al. 2018) (De Vries, Schoonvelde, and Schumacher 2018) and many others.

Additionally, text analysis techniques have been used to extract and model the structure of other type of sequences, such as musical chord sequences (Scholz, Vincent, and Bimbot 2009), genome sequences (Tomović, Janičić, and Kešelj 2006), microRNAs (Ding, Zhou, and Guan 2011), proteins (Vishnoi, Garg, and Arora 2020), etc. In these cases, n-grams analysis is usually the most popular technique. In the context of automated planning, n-grams have been used to predict operator sequences in plans (Muise et al. 2009).

Our work is based on these ideas. First, the family of *bag of words* techniques were investigated to be used in our approach. Then, we also analyzed *n-grams*, which are able to take in account the word order.

Bag of words

One of the simplest methods for text analysis is the *bag of words*. It consists in representing a text as the bag that contains the words of the text, ignoring grammar and word order, but keeping multiplicity. Given a corpus of text data, a list of words appearing in this corpus (dictionary) is created. For example, the dictionary obtained when processing the plans in Table 1 would be {`LOAD, DRIVE, UNLOAD, P1, P2, L1, L2, T1, T2`}; that is, the action names in $\mathcal{A}$ and the objects appearing in $\mathcal{I}$.

Once the dictionary is created, each document or data record, each plan in our case, is represented as a numerical vector based on the dictionary. There are several methods that can be used to create this vector. In this work, we have used the *count* and the *tf-idf* vectorizer methods:

- *Count vectorizer* converts a collection of text documents into a matrix of token counts, counting the number of repetitions of each unique word in a document. As an example, the vector for plan A in Table 1 would be:

load	drive	unload	P1	P2	L1	L2	T1	T2
2	1	2	2	2	3	3	5	0

  The plan length and its content can be represented with this method, but no mechanism is provided to represent the ordering of actions.

- *tf-idf vectorizer*, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a corpus (Rajaraman, Leskovec, and Ullman 2014). The value for each term t in a document d given a corpus D is the result of the following calculation:

$$\text{tf-idf}(t, d, D) = tf(t, d) * idf(t, D) \tag{1}$$

tf is the relative frequency of terms calculated for each term in each document:

$$tf(t, d) = \frac{n_t}{T_d}, \tag{2}$$

where n_t is the number of times term *t* appears in document *d* and T_d is the total number of terms in document *d*.

idf is the inverse document frequency, which measures how important a word is for differentiating each document, and it is calculated as follows:

$$idf(t, D) = log\left(\frac{D}{D_t}\right), \tag{3}$$

where *D* is the total number of documents and D_t is the number of documents that contain term *t*.

The application of this method to the term ”T1” (truck 1) of plan A in Table 1 produces the result:

$$tf(T1, PlanA) = \frac{5}{20} = 0.25$$
$$idf(T1, \{PlanA, PlanB\}) = log\left(\frac{2}{2}\right) = 0$$
$$\text{tf-idf}(T1, PlanA, \{PlanA, PlanB\}) = 0.25 * 0 = 0$$

Clearly, this method is not appropriate when comparing plans that employ the same resource pool. Moreover, it does not enable to represent the plan length and, like the *count vectorizer* method, it has no mechanism to represent the ordering of actions.

N-grams

The two methods described above ignore the text structure, and the *tf-idf* vectorizer method ignores the text length as well. With the aim to extract more helpful features for text classification, we carried out an analysis based on n-grams.

An *n-gram* is a sequence of n words (Jurafsky 2000). Given Plan A in Table 1, a 2-gram (or bigram) is a two-word sequence like `"LOAD P1"`, `"P1 T1"` or `"T1 L1"`, and a 3-gram (or trigram) is a three-word sequence; e.g., `"LOAD P1 T1"`, `"P1 T1 L1"` or `"T1 L1 LOAD"`. Thus, n-grams enable to capture from a fragment of one action to several actions (and their variables) as a whole.

Unlike the bag of words, where a dictionary entry is a single word, in the n-gram model the dictionary entry is a word sequence, and the vectorization process consists in counting how many times an n-gram appears in a given plan. An n-gram model is used to assign probabilities to entire sequences of words (Jurafsky 2000). It allows to detect differences in the ordering of actions and to follow up the use of a resource object of a plan.

We developed three different n-gram approaches in the vectorization process:

- *Basic* approach: a plan is assigned a vector that captures how many times an n-gram appears in the plan.
- *No resources* approach: plans are preprocessed to remove mentions to objects and resources, giving as a result a plan that consists solely of action names. After the preprocessing, the n-grams are calculated. For example, the two plans given in Table 1 are transformed as follows:

Plan A	Plan B
LOAD	LOAD
LOAD	DRIVE
DRIVE	UNLOAD
UNLOAD	DRIVE
UNLOAD	LOAD
	DRIVE
	UNLOAD

  Examples of 3-grams in Plan A are "`LOAD LOAD DRIVE`" or "`LOAD DRIVE UNLOAD`".

  This method ignores the objects and resources selected to achieve the goal. The focus is on detecting patterns of action-name sequences rather than patterns of objects and resources.
- *Anonymous resources* approach: after calculating the n-grams, resources are anonymized inside the n-grams. This way, references to different resources in the n-grams may remain the same if they have the same structure. This method enables to consider whether the objects and resources used in a n-gram are the same or not. We show this with two examples:
  - Example 1: given the following two 4-grams:

```
DRIVE T1 L2 L3
DRIVE T2 L4 L2
```

    the anonymization process will translate them into the same anonymized 4-gram:

```
DRIVE TX LX LY
```

    , which represents that a truck is driven from one location to another, regardless the truck and the locations. Therefore, it can be concluded that the two 4-grams given above share the same structure.
  - Example 2: given the 8-gram

```
DRIVE T1 L3 L2 DRIVE T1 L2 L1
```

    it will be transformed into:

```
DRIVE TX LX LY DRIVE TX LY LZ
```

    In this case, the same anonymized value is used for the two appearances of the term "T1". This may help to learn sequences in which the same resources are used.

  By using this anonymization process, syntactical differences in the use of objects and resources in semantically identical plans are ignored. We believe this will improve the accuracy of the recognition process because the focus is on the text structure rather than on the specific objects/resources that are used in the plans.

Experimental setting

This section describes the planning domains and behaviours they represent used in the experiments as well as the tested classification techniques.

Domains and behaviours

We used two planning domains in the experimental evaluation: the *Logistics* domain introduced in the first IPC and the *Rovers* domain introduced in the third IPC[2]. We used the LAMA (Richter and Westphal 2010) planner for solving the problems.

The *Logistics* domain consists in transporting packages from one location to another. Three behaviours were defined for this domain:

1. Behaviour ByOne consists in transporting one by one the packages to their destinations.
2. Behaviour LoadAll consists in loading all of the packages onto the truck before starting to unload.
3. Behaviour WithoutRestrictions consists in transporting the packages using the shortest plan possible. This behaviour does not impose any restriction on the order in which the actions are executed.

In the *Rovers* domain, a rover collects samples of soil, stones, and photographs and sends them to the central station. Five behaviours were defined for this domain:

1. Behaviour CollectFirst consists in collecting all samples before starting the transmission to the central station.
2. Behaviour RockFirst consists in collecting all of the rock samples before any of the soil samples.
3. Behaviour SoilFirst consists in collecting all of the soil samples before any of the rock samples.
4. Behaviour WithoutRestrictions consists in solving the problem using the shortest plan possible. This behaviour does not impose any restriction on the order in which the actions are executed.
5. Behaviour CollectFirstRock is a combination of RockFirst and CollectFirst behaviours. The agent first collects all rock samples, then collects all soil samples and, finally, it will start the transmission.

We generated 80 problems for the *Logistics* domain and 80 problems for the *Rovers* domain. Since every problem is run with all the behaviours defined for each domain, our set of samples will contain 240 plans for *Logistics* and 400 plans for *Rovers* domain.

Vectorization techniques

As previously described, we defined five different vectorization methods for feature extraction. This implies that, for each domain, five datasets are generated:

1. One dataset created by using the basic count vectorizer with bag of words (denoted as count-BoW)

[2]http://ipc02.icaps-conference.org/

2. One dataset created by using the tf-idf vectorizer with bag of words (denoted as tf-idf-BoW)
3. One dataset created by using the basic approach with n-grams (denoted as basic-n-gram) with length from 4 to 10 terms.
4. One dataset created by using the *no resources* approach with n-grams (denoted as no-resources-n-gram) with length for 1 to 5 terms
5. One dataset created by using the *anonymous resources* approach with n-grams (denoted as anonymous-resources-n-gram) with length from 4 to 6 terms.

The n-grams length has been determined empirically.

As shown in Figure 1, after the feature extraction step, each dataset is split into train and test data. We applied the *K-fold* cross-validation method with 5 splits (4 parts are used for training and 1 part for testing). Data is shuffled before splitting and no randomness is used in the folds (in order to make results more stable during the model testing). Finally, the average and standard deviation of the accuracy score for each fold is calculated.

Classification methods

In order to learn the behaviour model, four different classifiers, usually applied in text recognition, have been tested:

- Linear Support Vector Classification (LSVC) is based on the translation of the original vectors into a space of higher dimension. It searches for a separating hyperplane that maximizes the distance to two parallel hyperplanes representing the classes. This algorithm, among others, is suitable for solving problems of text classification (Cardoso-Cachopo and Oliveira 2003).
- Multinomial Naive Bayes (MNB) is a simple and widely used classification algorithms in NLP (Zhang and Li 2007).
- Decision Tree Classifier (DTC) is a supervised learning algorithm based on how humans solve forecasting problems. It defines a tree-like structure and splits the object space accordingly to a set of splitting rules located in non-leaf nodes. This technique is useful in text classification tasks (Saad and Ashour 2010). For this classifier, we have used the *entropy* to measure the quality of a split of a node based on the information gain.
- Random Forest Classifier (RFC) is a supervised learning algorithm that consists of many instances of the previously described decision trees. This is a fairly universal algorithm, including the one used in text classification problems (Xu et al. 2012).

Results

In the experiments with both described domains, the vectorization methods introduced in section ”Feature extraction” have been tested.

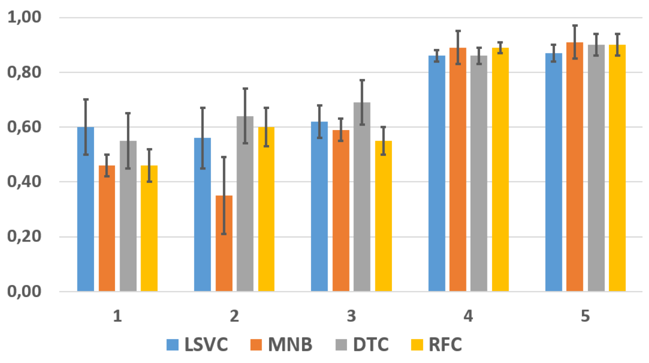

Figure 2: Behaviour recognition for Logistics domain depending on the vectorization method. 1-count-BoW, 2-tf-idf-BoW, 3-basic-n-gram, 4-no-resources-n-gram, 5-anonymous-n-gram

Logistics

Figure 2 show some results obtained for the *Logistics* domain, where bars represent the accuracy in average obtained with each classification technique and whiskers denote the standard deviation. Some interesting points regarding to these results can be highlighted:

1. We can see that count-BoW and tf-idf-BoW output similar results, with an accuracy below 60%. For these two vectorization methods, the lowest accuracy values are obtained when using Naive Bayes. The other classification methods, LSVC, DTC and TFC, do not exhibit a clear pattern. Low accuracy can be explained by the information lost during vectorization, given that these techniques are unable to capture the ordering of actions and the use of resources. Besides, the high standard deviation (7-14%) shows a high reliance on data splitting.
2. With respect to the basic-n-gram vectorization method, the results improve slightly, but the accuracy is still below 70%. This slight improvement can be explained by the additional information (ordering of actions) that n-grams take into account, which is a valuable information during the recognition stage.
3. Both no-resources-n-gram and anonymous-n-gram show the best results. The improvement with respect to the other vectorization methods is fairly significant, with accuracy values around 90%. Besides, the sensitivity of these methods on data splitting is lower (6% for Naive Bayes and 2-4% for the other classification algorithms). The reason is that these two vectorization methods are able to correctly identify the action order patterns in the plans representing each behaviour. Moreover, the slightly better results of the anonymous-n-gram method may be due to the fact that the for this domain resource usage is also important in each behaviour and this method is able to capture this feature.

Table 2 shows the confusion matrix obtained when using RFC with the anonymous-resources-n-gram vectorizer in the *Logistics* domain. It can be observed that the accuracy

	ByOne	LoadAll	WithoutRestr.
ByOne	1	0	0
LoadAll	0	0.88	0.12
WithoutRestr.	0	0	1

Table 2: Confusion matrix when using the Random Forest Classifier with the anonymous-resources-n-gram vectorizer method for the *Logistics* domain

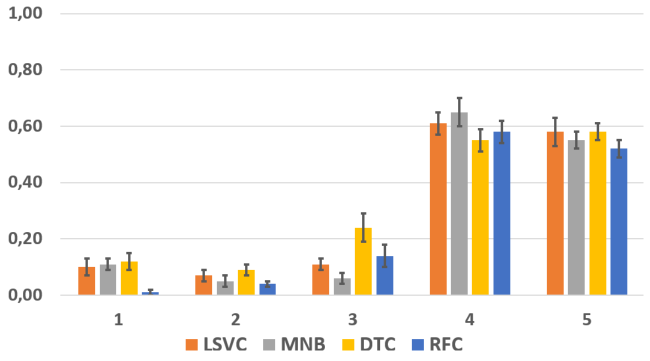

Figure 3: Behaviour recognition for Rovers domain depending on the vectorization method. 1-count-BoW, 2-tf-idf-BoW, 3-basic-n-gram, 4-no-resources-n-gram, 5-anonymous-n-gram

when identifying the ByOne behaviour is 100%, whereas it is slightly more difficult to distinguish between the LoadAll and WithoutRestrictions behaviours, given that these behaviours are more similar to each other.

Rovers

One of the objectives of the experimentation with the *Rovers* domain is to analyze the effect of having a higher number of behaviours. The confusion matrix with all the behaviours is shown in Table 3. It can be observed that the CollectFirstRock and CollectFirst behaviours are sometimes incorrectly identified. This is because both behaviours establish an similar order between the collecting and transmission actions. Differences in the plans of these two behaviors are observed only at the stage of sample collection, and the parts of the plan responsible for transmitting data after sampling can be exactly the same. Thus, in simple tasks, where only one type of samples is presented to collect, the plans for these behaviors can be absolutely identical. The same happens with RockFirst and SoilFirst; in this case, the main difference between the actions involves the objects that are processed (rock and soil). And the only difference in the data obtained after conversion to n-grams is only due to the part of the plan in which the collection mode changes from one type of sample to another.

Figure 3 and Table 3 shows the results for the *Rovers* domain when **all the behaviours** (CF- CollectFirst, WR-WithoutRestr, SF-SoilFirst, RF- RockFirst and CFR-CollectFirstRock) are considered using all described vectorization and classification techniques. The best recognition value is around 65%, obtained with Multinomial Naive Bayes and no-resources-n-gram, albeit this value is far from the 90% obtained for the *Logistics* domain. This worsening in accuracy may be due to the higher number of behaviours. Therefore, we decided to perform two more experiments removing some behaviours to analyze how much they affect the recognition accuracy.

As stated earliery, behaviours CollectFirstRock and CollectFirst are very similar. Confusion matrix presented by Table 3 shows that having behaviour CollectFirst gives a 40% error on recognition behaviour CollectFirstRock and having behaviour CollectFirstRock gives 60% error on recognition CollectFirst. First we remove CollectFirstRock since CollectFirst is a more general behaviour. Confusion matrix for recognition results is presented in Table 4. It can be observed that the accuracy has improved compared to the prior experiments. Accuracy on recognition for behaviour CollectFirst improved from 28% to 80%. In this confusion matrix, it can also be observed that RockFirst and SoilFirst behaviours are still not well identified. Having behaviour RockFirst, gives a 36% error on recognition of SoilFirst behaviour, and having SoilFirst behaviour gives a 16% error on recognition of RockFirst behaviour. Experiments have shown that there is not much difference in overall accuracy when any of these two behaviors is excluded. Confusion matrix with recognition results after removing RockFirst behaviour is presented in Table 5. It can be observed that the accuracy on recognition of SoilFirst behaviour improved from 60% to 90%.

The dependence of overall recognition accuracy on the number of included behaviors is shown in Figure 4. The first group of bars of Figure 4 repeats data shown in bar 4 of Figure 3, representing recognition results for classification methods applied with anonymous-n-gram. The second group of bars of Figure 4 show the results obtained when removing behaviour CollectFirstRock and the third group of bars of Figure 4 show the results obtained when removing behaviours CollectFirstRock and RockFirst. It is seen that accuracy improves from (52-58)% with all of the behaviours presented in dataset to (72-76)% when behaviour CollectFirstRock is excluded and to (78-89)% when behaviours CollectFirstRock and RockFirst are excluded. This configuration obtains the best results in accuracy for this domain.

In summary, it can be concluded that increasing the number of behaviours, when some of them are similar between each other, causes a decrease in the recognition accuracy.

Both domains described in this paper mostly relate to the transport type of problems, however, *Rovers* is a more complex version of it, since it has a large number of possible actions and resources and allow you to solve a wider range of tasks, which can affect the accuracy of plan recognition.

Conclusions

This paper focuses on the use of text analysis techniques in order to identify the behaviour followed by an agent when solving a planning problem. The main idea is to transfer

	CollectFirstRock	CollectFirst	RockFirst	SoilFirst	WithoutRestr.
CollectFirstRock	0.56	0.4	0	0	0.04
CollectFirst	0.6	0.28	0	0.06	0.06
RockFirst	0.15	0.04	0.59	0.22	0
SoilFirst	0.09	0	0.39	0.52	0
WithoutRestr.	0.16	0	0.4	0.12	0.68

Table 3: Confusion matrix when using the Decision Trees Classifier with the anonymous-resources-n-gram vectorizer method for all the behaviour defined in the *Rovers* domain

	CollectFirst	RockFirst	SoilFirst	WithoutRestr.
CollectFirst	0.86	0.04	0	0.1
RockFirst	0.08	0.76	0.16	0
SoilFirst	0.04	0.36	0.6	0
WithoutRestr.	0.24	0.04	0	0.72

Table 4: Confusion matrix when using the Decision Trees Classifier with the anonymous-resources-n-gram vectorizer method for behaviours CF+WR+SF+RF defined in the *Rovers* domain

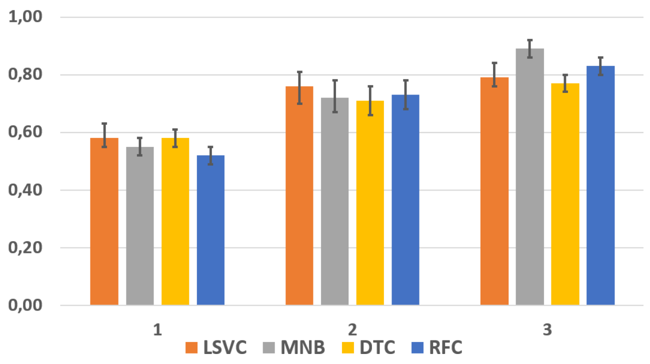

Figure 4: Behaviour recognition for the *Rovers* with behaviours 1) CF+WR+SF+RF+CFR, 2) CF+WR+SF+RF, 3) CF+WR+SF

	CollectFirst	SoilFirst	WithoutRestr.
CollectFirst	0.76	0.04	0.2
SoilFirst	0.05	0.9	0.05
WithoutRestr.	0.15	0.4	0.81

Table 5: Confusion matrix when using the Decision Trees Classifier with the anonymous-resources-n-gram vectorizer method for behaviours CF+SF+WR defined in the *Rovers* domain

the use of these techniques to extract the structure of the sequence of actions in a plan that denotes a particular behaviour. Specifically, bag of words and n-grams vectorization methods have been analyzed, using different forms of input plans. Additionally, several classification techniques have been tested. All in all, the best results have been obtained with n-grams models using anonymized resources in plan actions, combined with decision-tree based classification techniques (including also random forests).

In our experiments, several behaviours have been defined, mainly imposing restrictions in the order in which actions and goals have to be reached. The results show that, when behaviours are clearly distinguishable, the accuracy is high, like in the *Logistics* domain. However, when similar behaviours (because the action structure is similar, although using different objects, for example) are defined, it is more difficult to correctly identify these behaviours. This is a point where we must work on.

First, we will test described methods on additional domains. Moreover, we have observed that to solve problems in the *Rovers* domain, many actions not directly related with the specific behaviour are included in the plans (that is, the same actions appear in all plans and, therefore, it is more difficult to distinguish between behaviours). For this reason, we will try to improve the plans used to train the models, maybe preprocessing them to extract the key actions or generating the plans by means of behaviour trees.

References

Barushka, A., and Hajek, P. 2019. Review spam detection using word embeddings and deep neural networks. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, 340–350. Springer.

Cardoso-Cachopo, A., and Oliveira, A. 2003. An empirical comparison of text categorization methods. *Lecture Notes in Computer Science* 2857:183–196.

Cavazza, M. 2000. AI in computer games: Survey and perspectives. *Virtual Reality* 5(4):223–235.

De Vries, E.; Schoonvelde, M.; and Schumacher, G. 2018. No longer lost in translation: Evidence that google translate works for comparative bag-of-words text applications. *Political Analysis* 26(4):417–430.

Ding, J.; Zhou, S.; and Guan, J. 2011. mirfam: an effective automatic mirna classification method based on n-grams and a multiclass svm. *BMC bioinformatics* 12(1):216.

Jurafsky, D. 2000. *Speech & language processing*. Pearson Education India.

Ma, S.; Sun, X.; Wang, Y.; and Lin, J. 2018. Bag-of-words as target for neural machine translation. *arXiv preprint arXiv:1805.04871*.

Marcotte, R., and Hamilton, H. J. 2017. Behavior trees for modelling artificial intelligence in games: A tutorial. *The Computer Games Journal* 6(3):171–184.

Muise, C.; McIlraith, S.; Baier, J. A.; and Reimer, M. 2009. Exploiting n-gram analysis to predict operator sequences. In *Nineteenth International Conference on Automated Planning and Scheduling*.

Peersman, C.; Daelemans, W.; and Van Vaerenbergh, L. 2011. Predicting age and gender in online social networks. In *Proceedings of the 3rd international workshop on Search and mining user-generated contents*, 37–44.

Rajaraman, A.; Leskovec, J.; and Ullman, J. 2014. *Mining of Massive Datasets*.

Richter, S., and Westphal, M. 2010. The lama planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research* 39:127–177.

Rodrigues, R. G.; das Dores, R. M.; Camilo-Junior, C. G.; and Rosa, T. C. 2016. Sentihealth-cancer: a sentiment analysis tool to help detecting mood of patients in online social networks. *International journal of medical informatics* 85(1):80–95.

Saad, M., and Ashour, W. 2010. Arabic text classification using decision trees.

Scholz, R.; Vincent, E.; and Bimbot, F. 2009. Robust modeling of musical chord sequences using probabilistic n-grams. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, 53–56. IEEE.

Sitanskiy, S.; Sebastia, L.; and Onaindia, E. 2020. Behaviour recognition of planning agents using behaviour trees. In *24th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems*, in press.

Tomović, A.; Janičić, P.; and Kešelj, V. 2006. n-gram-based classification and unsupervised hierarchical clustering of genome sequences. *Computer methods and programs in biomedicine* 81(2):137–153.

Vishnoi, S.; Garg, P.; and Arora, P. 2020. Physicochemical n-grams tool: A tool for protein physicochemical descriptor generation via chou's 5-step rule. *Chemical Biology & Drug Design* 95(1):79–86.

Wang, M.; Cao, D.; Li, L.; Li, S.; and Ji, R. 2014. Microblog sentiment analysis based on cross-media bag-of-words model. In *Proceedings of international conference on internet multimedia computing and service*, 76–80.

Xu, B.; Guo, X.; Ye, Y.; and Cheng, J. 2012. An improved random forest classifier for text categorization. *JCP* 7:2913–2920.

Zhang, H., and Li, D. 2007. Naïve bayes text classifier. 708 – 708.