Testing with Artificial Intelligence

whitepaper
TestNet workgroup ‘Testing and AI’
version 1.0 - May 2019

written by:
Sander Mol, Hannie van Kooten, Gerald de Vrieze, Rik Marselis,
Derk Harmanny, Lysette Caetano do Rego, Marco Verhoeven,
Marek Lof, Martin Heining, Martin van Helden, Oktorijanto Wahjuwibowo,
Ramin Sadeghi Zadeh, René Hegeman, Reza Samie Fanny, Sander Schrama,
Sebastiaan van Houdt and Taco Verhagen.

Table of contents

1. Introduction

Self-thinking and self-learning systems play an increasingly important role in society. The testing profession is also changing considerably because of the introduction of artificial intelligence (AI). The TestNet workgroup 'Testing and AI', until May under the name 'Testing with AI', investigates the possibilities of testing using AI. With this whitepaper we provide insight into the current state of affairs and a glimpse into the future.

The reason for starting this workgroup was an article that Sander Mol and Rik Marselis wrote about 'testing with self-learning and self-exploring testing tools', followed by a presentation on a TestNet theme evening in September 2017.

What is AI?

Simply put, AI is the ability of machines to perform tasks and activities that we regard as 'intelligent.' AI, more broadly defined, is the ability of an intelligent system to observe its environment and perform specific tasks in order to achieve a goal as well as possible.

In testing, the goal is to collect information about the quality and risks of an information system. AI can support this by observing the information system and collecting data. This collected data can be used in a report. The AI can also learn from the collected data and thereby become increasingly better at performing the tasks.

The forms of AI relevant to this whitepaper are based on 'Machine Learning' where a smart algorithm analyzes data and can draw conclusions based on that. In the case of unsupervised learning, the algorithm ensures grouping of data. With supervised learning, people help to determine the grouping, for example by labeling the data. Usually unsupervised and supervised learning go together.

If a model is good enough (and therefore has learned enough), then it can stop training. In other situations it is desirable that the model also continues to learn while being used in the live environment.

Many governments are offering a course to introduce the concepts and implications of AI. In the Netherlands this course can be found at www.ai-cursus.nl and has been prepared in collaboration with universities and the business community. If you mainly want to get to know the most used AI techniques, you can find our initial introduction on our TestNet page.

AI in testing

When talking about testing and AI, this could lead to confusion. It can be about 'testing of AI' whereby the quality of an AI system is investigated, but also about 'testing with AI' where AI is used to support the test work. This document is about 'testing with AI', where AI is a technique that should make testing tools more efficient and / or more effective.

A regularly emerging discussion is whether a testing tool is ‘real AI’. Some testing tools are ‘only’ Business Intelligence (BI) or ‘only’ statistics. What is seen as ‘real AI’ varies over time and per situation, there is no comprehensive, indisputable definition. On balance it is all about the interpretation of a lot of data (whether or not collected live) to improve the tests. So we are mainly looking for ‘tools that use large amounts of data to make testing smarter.’

The ultimate situation is that AI will be able to independently explore an application, assess the quality and risks and write a legible report about it. We are by no means here yet. On the other hand, it is already clear that there will not be one AI testing tool that does all of this, but a chain of AI testing tools that together handle the various tasks. In the growth process towards the ultimate situation, we will see that one task is first taken up with AI and that more and more tasks are gradually being entrusted to the AI testing tool.

We doubt whether most of the testing activities will be taken over by AI anytime soon. Some test tasks are easy to transfer, but others require insight and creativity, input specifically from human testers.

This whitepaper covers the current state of affairs (spring 2019) from the entire testing process and examines where AI can offer added value now and in the future. It is the result of discussions within the workgroup and with national and international experts, research and demos about existing AI testing tools and own experiments with the various concepts.

The workgroup members wish you, our reader, a lot of success and joy in applying AI in the testing profession.

2. Possibilities for testing with AI

What can AI - and especially machine learning - really add to the testing profession? To specify this further, a better understanding of what AI actually does is needed first. This chapter discusses the most important building blocks, in the context of the testing process.

Recogize objects

Existing testing tools have two ways to identify objects; by capturing properties (such as dimensions, text in the object or XPath) or by capturing the appearance and then scanning the page. In the latter case, an image can be converted to a text (or OCR). Despite all these options, a common problem of user interface testing is that they are slow, brittle and require a lot of maintenance.

The use of AI can help with this. AI can learn the appearance and properties of an object. If something changes, the AI will return a certain degree of certainty that the changed object is what you are looking for. For example, the AI does not find the Login Button object with 100% certainty, but an object that resembles 78% and another object that resembles 65%. As a tester you can determine in advance how certain the AI should be, in order to have your test continue automatically. Of course, the reason for the reduced recognition must also be investigated later, so that this can be solved.

Although this improved recognition looks intelligent and is therefore AI, it is not yet the machine learning application that can really surprise us. This application lies in the ability of an AI to interpret objects more broadly on the basis of many examples and on the basis of context.

Learning from examples means that we show a lot of different objects to an AI, which we think belong to the same group. This mean we have to label everything, which makes it a form of supervised learning. For example, all buttons on our test object – such as a 'website' - will probably have roughly the same properties. If an AI trains on that, it can later immediately recognize and use a new button (see below, 'Exploring to achieve a goal'). Moreover, you can compare your own buttons with thousands of other websites and thus get a very robust idea of what 'a button' is. Later in this whitepaper we see how the Test.ai tool does this with web stores.

Recognizing objects becomes even more reliable if the AI takes the context into account. It can scan a complete page (via an advanced, 'convolutional' neural network) and, for example, estimate which buttons belong to a menu. And which buttons are likely to lead to the next page. And what you can probably do with a certain object (such as clicking something, selecting or typing in list boxes, checkboxes and links). The AI can recognize these trends on its own test object, but also on thousands of other test objects.

Exploring to achieve a goal

An AI can investigate and explore a test object by itself. Machine learning learns from data, including data generated by an AI itself. This concept is called reinforcement learning; the way in which an AI acts is reinforced by previous experiences and by the valuation of those experiences. More specifically, this means that we as a tester give our AI testing tool bonus points and penalty points for achieving certain goals. Clicking on a link or a button could, for example, yield 10 points, every elapsed second 1 penalty point and issue a specific text 100 points. The AI testing tool will then try to gain as many points as possible and therefore optimize the path to a certain goal.

It is up to the tester to set the goals properly. What do we do with a 404 page, is that a bonus point or a penalty point? That of course depends on what the tester wants to achieve with the test.

A logical question then is how an AI testing tool knows what the possible actions are. This could be set by a tester himself, manually or by loading the entire object model. But we can also let the AI testing tool figure out ourselves through object recognition.

Recognize trends in data

AI and machine learning are perfectly suited to use large amounts of data and to find useful trends in it, via techniques such as random forests[1] and in particular neural networks. Data - in the form of large amounts of text and code - is nowadays fully available and either be used directly or after data preparation.

With an existing application you can use the log files from production to discover the most important use cases. You can also cluster the users into a number of categories based on their actions.

You can also use data for more targeted testing. For example, you can train an AI on tickets in an issue management system to discover what often goes wrong and how much time it usually takes to solve something. If you have a good change registration, you can discover which changes have a high risk and what you should test for. A final, more complex application is the analysis of documentation, such as a requirements document or a functional or technical design. As testers, we already recognize ambiguities, omissions or error-prone subjects during the review. If trained with sufficient examples, an AI could also help with this. However, this is somewhat more difficult because this data contains much less structure than log files.

Finally, trend analysis via AI is very useful for reports and dashboards. Especially if our AI testing tool has done exploration by itself, countless paths will have been tried and we will undoubtedly have discovered some deviating, trend-breaking behavior and perhaps even errors. An AI testing tool can provide input for such reports, although it is not yet clear how human-readable this can be made.

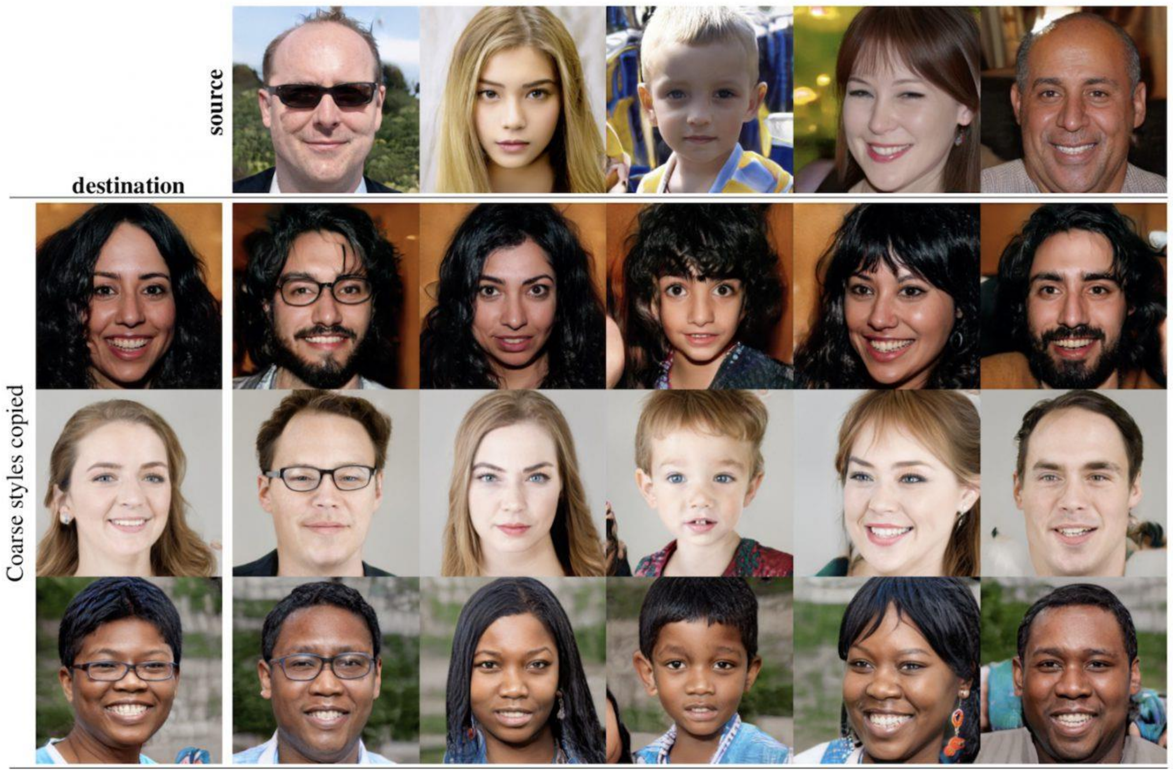
[1] Random forests is a technique for predicting an outcome (output) with constantly changing variables (input), to see which combination of variables is the best predictor. https://en.wikipedia.org/wiki/Random_forest

Text recognition

Of the three options mentioned in this chapter thus far, text often plays a role. A specific AI technique, Natural Language Processing, helps to interpret spoken or written text. This technique not only looks at the word itself, but can also look at the words before and after it to give the correct interpretation. That is also the reason that translation programs have become so good lately. And why translation this whitepaper to English has been done within half a day.

Data generation

AI is not only good at recognizing trends in data, it can also generate data itself using these trends or 'styles'. Think of creating images, such as the image below (in this example, the AI generates a new photo based on the photos on the left and above that is sort of an average of the two photos).



Source: https://www.lyrn.ai/2018/12/26/a-style-based-generator-architecture-for-generative-adversarial-networks/

For testers, a more specific application would be generating test customers or generating the right response from the stub of a surrounding application. Here too, of course, many examples are needed to train a model, such as a customer database or a large set of requests and responses from the other application.

Because it is so abstract, we briefly discuss the technology behind data generation; the Generative Adversarial Network or GAN. Two neural networks are used here. One network (the generator) creates the data, another (the discriminator) determines how good the data is (compared to real examples) and then gives instructions to the generator. This cycle is repeated repeatedly until the generator is able to create data that cannot be distinguished from the real by the discriminator. So you can use a GAN, for example, to see if you can trace anonymized data to production data and thus do a privacy test.

3. Challenges when testing with AI

While there are plenty of opportunities to improve testing through the use of AI and machine learning, we also see challenges. In this chapter we describe the challenges that the workgroup considers most important. Further research will be needed to determine which topics are the most urgent. We not only look at the shorter term, but also have a longer-term horizon. After all, some of the challenges will automatically come our way, while with other challenges we should pay more attention to up front, so that we can prevent subsequent repressive and corrective measures.

Trust in AI

AI is something new for most people. Many people have a natural distrust of innovation, at AI this is increased because it is difficult to understand and therefore people cannot oversee the possible consequences. The media has a tendency to reinforce this image by giving much attention to the critical sounds of Elon Musk and Stephen Hawking, for example, about the fact that AI may soon do what we do NOT want it to do. In Hollywood there are plenty of films made about super intelligence and singularity – where AI surpasses human intelligence - with Terminator and Sky Net as cult examples. For the time being, super intelligence is decaces away at best and may also be pure fiction forever.

Still, this fear of AI causes resistance. This attitude leads to slower development and therefore competitve problems with regard to companies and countries that do embrace the technology. In European politics you see this fear result in guidelines. The GDPR states, among other things, that a machine may not automatically reject customers (which applies to both AI and 'normal' programs).

New technology naturally entails new risks. But in our opinion much of that fear is unfounded. How can you remove the unrealistic part of this fear? Knowing what AI really is helps to make the mystery tangible, such as through the national AI course that we mentioned in the introduction.

In addition to general knowledge, it helps to provide practical insight into the operation of AI, by gradually introducing AI and control the results. Let the AI first give suggestions, use it in parallel with the existing way of working. Apply it to simple and repeated work so that it can quickly add value. Only expand it once there is enough confidence.

Trust in AI for testing

The theme of 'trust' naturally also plays a role in the use of AI testing tools. We are convinced that AI will help with all kinds of testing tasks, as we will explain in Chapter 4. But to leave our important test work to a testing tool, we must be able to rely on such a tool. The first question that comes to mind is: ‘how do we know that the AI testing tool is good? Who will test the AI in a testing tool?’

How do you know if an AI testing tool is good (enough)?

Also for AI testing tools, there is little practical experience available from the organizations that use them. Some limited experience stories can already be found on the internet. In addition, some AI testing tool suppliers make great efforts to highlight the possibilities and qualities of their testing tools. Yet the best way to determine whether such a tool delivers added value in practice is simply to try it out. For example, via a small-scale pilot and then adjust the approach based on experiences.

However, this involves other practical challenges. Purchasing an AI testing tool through a supplier entails costs, while the limited insight into AI and machine learning means that the question is whether it will yield enough benefit. On the other hand, developing a testing tool yourself is a challenge, because various insights and skills are needed that were not needed with a traditionally programmed application (such as knowledge of machine learning models and statistics and skill in processing a large amount of data).

It therefore helps to set a clear goal before choosing an AI testing tool. And then adjust this goal based on practical experience. This immediately links up with the following topic: expectation management.

Unrealistic expectations

While people on the one hand may be very skeptical, on the other hand they may be over-enthusiastic. Customers of AI testing tools sometimes think that all their problems are magically solved, whether or not prompted to think this by suppliers of these tools.

In Chapter 5, we look more closely at practical applications, but it is clear that AI is currently primarily a tool for simple tasks, where you can train on a large number of ready-made examples. Think of improved object recognition or the automatic exploration of a limited number of functional paths.

The application of document analysis, the generation of a usable model or the contribution to non-functional tests are still in their infancy. Depending on the environment in which you are going to use AI testing tools, it is important that you do not start working on this or do so very carefully for the time being.

AI testing depends on data

AI, specifically machine learning, is the recognition and application of patterns in data. Analyzing large datasets result in a model that can be used to assess new data and carry out a corresponding action. The quality of this data used is therefore directly linked to the quality of the generated model.

Transparency of decisions

With an AI you may not always have full access to the input data, the model and the output. To achieve transparency, you'll want a 'glass box' approach that traces how the AI algorithm arrived at it's conclusions and actions. This glass box approach is feasible with certain types of machine learning. Other forms of machine learning (such as deep learning algorithms) are so complex that it is practically impossible to be fully transparent. Even if you keep a log of all the model's decision steps, you will come across an inscrutable mountain of millions of partial decisions.

An example of attainable transparency is indicating which group of pixels of an image are decisive for the AI to classify the image in question as a cat, for example. However, a picture is 2-dimensional and therefore easy to visualize. A model that trains on 100 variables is 100-dimensional and therefore much more difficult to visualize. Efforts are being made to do this, in this case by reducing the 100 variables to a maximum of 3 variables in order to be able to visualize it. For software, glass box testing techniques have been developed, for AI we are still discovering this. A term that is increasingly used in this context is 'explainable AI', abbreviated to XAI.

Paradigms give subjectivity

You can train an AI from a set of available data. By definition, this data provides a subjective representation, for example because it has been decided in the past what to record, when to record it and how to record it. In addition, this data always comes from a subset of reality. At a hospital in The Hague, for example, this is 'all people who have ever registered at this hospital', so that you do not include countless groups of other people in your model.

In addition, training an AI always happens from a certain perspective. The selection and evaluation of the data that you use for training the model is also a conscious and therefore subjective choice. The model creation is influenced by who makes these choices. Also consider culture, religion, education, gender, descent, wealth and so on.

Together this can lead to a trained model that gives biased results. A now classic example is the selection of employees at Amazon[2]: historically, almost all highly regarded managers are male, because there were virtually no female managers in the past. The Amazon AI therefore learned to prefer male managers based on this historical data, unless you ensure that gender is not explicitly included when training the machine learning algorithm. And then it is still possible that the effect of gender still indirectly plays a role via other variables.

Getting usable data takes a lot of time

In addition to the fact that AI is often not transparent and potentially unethical, there is another challenge when using a lot of data; usually this data is not immediately suitable for training. This is a major problem with Business Intelligence solutions. Data analysts and

[2] https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G

business analysts have been struggling with this for years and it is not a problem purely for testing with AI, but for applying AI in general.

Data sets are incomplete, contain unclear values and have been filled based on varying guidelines over the years. Moreover, the value of the data can be filled via different standards; just think of different units of measurement (such as length in meters or inches) or standards (such as postal codes in the Netherlands or Belgium). Cleaning up the data before it can be used by the AI must therefore be explicitly and carefully considered first.

Missing data provide an additional challenge. If it is not possible to delete the entire record, for example because half of the dataset would be dropped, clear and honest conversion rules will have to be devised to supplement this data.

AI projects will generally be affected by this, but this also applies to the use of data for AI testing. Consider, for example, issue records from which you want to recognize patterns.

Technical issue: overfitting

In the case of overfitting, the AI system has trained a model that very closely matches the examples of the training set, but the system is not very capable to assess new situations. For example: the AI perfectly recognizes the 20 cat and dog pictures that were trained with, but cannot correctly classify a new cat, or a kitten or rare breed that was not included in the training set.

For testing with AI, this can occur, for example, if you want to recognize an object, such as a button or link, within the application that you are testing. If you train the AI to recognize exactly the existing buttons and links, it may not recognize new buttons and links later. The same applies when training on issues records; Here too, the model must continue to generalize sufficiently to provide a valuable prediction of, for example, seriousness or lead time.

Interpreting AI test results

A feature of AI - and especially machine learning - is that it recognizes patterns in large numbers of examples and searches for paths based on a large number of explorations. This makes it challenging to interpret the results of such an AI testing tool.

The first part of this challenge is result prediction; if the testing tool does not (or no longer) recognize an object, has the object actually disappeared or has the testing tool trained for new, different examples?

The second part is the number of results that a testing tool returns. After an exploration of the application, it may return with thousands of explored paths and hundreds of situations that the AI considers worth viewing. This will especially be true if a self-learning AI testing tool is just started, it can be a huge job to assess all outcomes.

And even after we have made a distinction between relevant and non-relevant issues of the AI testing tool, we probably have a much larger number of measurements on which we have to give our one final assessment. Testers and their clients will find it a challenge to make this judgment, which partly explains the rise of the field of ‘decision science’; making business decisions based on data and models. More about such consequences for the testing profession can be found in the chapter ‘Future of testing’.

4. Use of AI in testing activities

Many different test activities are described in the testing profession. In this whitepaper, the TMap activities were taken as a starting point, but they have been extended to be able to do a more complete analysis. We have chosen to use the following classification of test activities:

Test management	Planning	Risk management	Maintenance	Reporting	Issue management
Test engineering	Preparation	Specification	Execution	Finalizing	
Testing support	Infrastructure	Test data management			

For each of the activities, we briefly describe what the activity entails and then how AI can help with this activity.

Test management

Planning

When planning the tests, the test manager will first focus on the change or changes in the IT landscape (software and hardware) to determine the **purpose**, **scope** and **risks** of the test activities and to estimate a lead time. This information can come from structured and unstructured documents and from voice recordings. AI techniques such as Natural Language Processing (NLP) can help to convert, analyze and even summarize these texts and speech recordings.

AI can then also be used if new information about the change becomes available, as a supplement to previously used information and therefore for adjusting the purpose, scope and risk.

Risk management

If the change is known, the test manager will investigate what the risks are so that the test effort can be divided based on those risks. An AI tool can interpret the requirements or design documents and extract the most important words and word combinations from the sentences. The AI technique used is the recurrent neural network (RNN), which contains a memory module (Long / Short Term Memory, LSTM). These words and word combinations are then given a risk score. This score must have already been built up in a baseline, either by assigning a score manually or by training on sample texts and the associated risk score.

The sources for arriving at the baseline need not only be scored requirements or design documents. Scored issues from an issue management system are also possible inputs. Of course, all data sources will have to be unambiguous and sufficiently complete.

Using the same technique, risks can be translated into business impact. However, this is a lot more difficult because an AI does not have a view of ‘real world’ scenarios, or a context. It only knows the things it is trained on and determining business impact is often tailor-made.

The final part of risk management is providing insight into the risks to the person who decides about which risks should be mitigated (and provides the budget to do that). An AI can summarize the risks as a basis for distributing the test efforts, and can also make the issues and associated product risks clear during test execution.

Maintenance

After the design of the testing process, it is important to monitor the test execution properly and to make adjustments where necessary. AI can help with this by providing insights into the state of affairs via smart dashboarding on various aspects such as coverage of the tests, proven quality of the test object, covered product risks and insight into the status of issues. Moreover, the progress of the testing can be logged, so that an AI can generate useful insights from this via data analysis. For example, it can provide suggestions about problem areas, so the test manager can adjust faster.

Moreover, such analyses and smart dashboarding can be used to evaluate whether the exit criteria for testing have been met. A condition to be able to do that, is that the exit criteria must be formulated very clearly and measurably.

Reporting

Reports are important, which is why we have decided to include reporting as a separate test task (whereas in TMap it is part of maintenance). Reporting uses information from the monitoring smart dashboards as input. These smart dashboards provide the interpretation of information to support decision-making.

Smart dashboards not only make the current situation transparent, they can also predict potential future changes to the quality of the information system based on data from test results and data from monitoring the live environment. This makes it possible to take corrective measures even before quality problems actually occur.

Issue management

TMap handles issue management in multiple places, therefore it is included here separately. In some cases, AI can independently detect deviations between expected and actual situation, and independently establish an issue based on this. A condition to do this, is that the expected and actual situation must be clearly described and comparable, for example when comparing an old and new version of the application. The AI can also make an estimate of the severity of the issue on the basis of the previously determined risk analysis as this is usually related to the risk.

An AI can also help with the assessment of issues entered by testers. Based on previous issues that have already been classified or even closed, the AI can see if an issue falls within a pattern and thus say something about the category, severity or lead time. Also, when the reporter of the issue has entered a category and severity, the AI can do quality control on the submitted issue. To do all this, the AI will have to run through the issue, using the same technique that was used to assess requirements and risks (RNN with LSTM).

By allowing the AI to look at the content of a report through this technique, another useful application is created: detecting duplicate registrations by recognizing patterns in issues and reporting the agreement percentages.

But we can go one step further if we use problem management in addition to issues management. The AI analyzes an issue, performs root cause analysis and provides tips on how to resolve issues. On the one hand by proposing (or perhaps even implementing) concrete adjustments to the software that resolve an issue, but on the other hand by making adjustments to the process so that the same error will no longer occur in the future.

Test engineering

Preparation

The test engineer will start preparing the tests. This means reviewing specifications to determine functionality and non-functionals. This information can be recorded in ordered and unordered documents, but also in voice recordings. The AI can help to extract information from the documentation via Natural Language Processing with the AI technique of RNN / LSTM. The goal in this case is to find out per object: what are the possible actions, for each status or context. The AI can summarize this information by making it a model filled with all objects and paths.

Based on this model, combined with the previously found risk areas, an AI can then identify missing, incorrect or conflicting design decisions. And by incorporating the context, the AI can determine that different words are used for the same topic. In short; an AI could analyze all kinds of texts for the purpose of sorting and summarization, but also to support the review.

Specification

When specifying the test cases, an AI can help by choosing the right test specification technique. This becomes possible once the AI has been trained on a large number of pieces of documentation (input) and the optimal specification technique (output). In addition, you can give the AI a score of the coverage that a technique has achieved, based on the previously prepared model of the application. For example, the AI could see what it adds to specify a decision table test if a process cycle test has already been set up.

If a specification technique is chosen (or several of them), then it is important to apply it to the model. An AI can help to generate test situations with a test design technique and then convert them into logical test cases and physical test cases, including required test data, in the same or a different technique. The AI can then help to minimize any duplicate test cases (from the different test design techniques), by optimizing (i.e. minimizing) the number of operations within the entire test set. An optimal regression test can also be generated this way.

If a link is made with the issues registration, it becomes possible to recognize which tests cannot yet be performed. The cause may be that certain parts of the system are not yet available or reliable.

A very different application of AI in the context of test specification is the selection of test cases made during a development phase for the regression test. Since a regression test set is generally more limited than the original progression test set, the AI can make a selection of the most suitable test cases on the basis of various criteria (for example, a relationship with risk to be hedged).

For completeness; in this section in particular, machine learning and ‘programmed intelligence’ are intertwined. However, this fits in with the idea that AI and machine learning will have a place in the testing process, without directly taking over everything.

Execution

Especially the tests at a lower level, such as unit testing, can be created and carried out autonomously by an AI based on standards and best practices.

Functional tests can, according to specification, be performed by existing testing tools. No AI is needed for this, unless we want an AI to look for an alternative path once the test crashes based on specification.

An interesting variation on ‘testing according to test specification’ is ‘testing without test specification.’ This is actually an exploratory test, in which specification and execution of the tests are combined live. The AI discovers the possibilities, tries these possibilities out and tries to combine these possibilities until the goal is achieved. Or, if no goal has been set, until all possibilities have been tried (which is very unlikely) or the test execution is stopped.

To use this approach, an AI technique is needed to determine the location (in the form of a, usually convolutional, neural network) and to determine the route (again RNN with LSTM).

For both specified tests and exploratory tests, an AI can help to make a proposal for an issue, including categorization and classification. However, traditionally programmed testing tools are well advanced. There will not be much profit here in the case of loose, single issues. It can help if the AI reports the full scope of the issue based on trends and patterns, in the form of ‘on page A, B, G, H and I, this issue occurs.’

Finalizing

An AI can help in assessing the testing process. The AI analyzes all information about what is recorded and said during development, testing and during the retrospective and can extract trends that lead to suggestions about possible improvements.

In addition, a tester usually records the testware. An AI can support with version management, configuration management, saving of testware and retrieving testware.

Testing support

Infrastructure

In theory, it is possible that an AI helps to define and prepare the test environment, based on the tests to be performed. Based on which test cases are present, the AI analyzes what requirements there are for the test environment. An AI can set up and prepare the test environment based on those requirements. However, this application does not have very well-defined inputs and outputs. This does not seem profitable for a loose implementation, but it might be interesting as a standard service.

A more concrete application is the creation of stubs and drivers. This would be possible with an existing application; let the AI learn what requests and responses there are and then let it generate the result itself. With new applications, the previous step (data generation) is required first.

Test data management

AI can help with the definition and preparation of the data set, based on the tests to be performed. AI can generate synthetic test data based on an analysis of live data. This immediately prevents the problem of privacy-sensitive test data.

This requires fairly specific knowledge of data, but if at all possible, it will be via LSTM. Based on the model, it is possible to derive which data components are needed, which can then be generated. This will ensure that fictional data is created, via RNN.

5. Getting started

This chapter shows the ways to get started with AI in the testing process. Most of the times, the experience and trust in AI and AI testing tools still have to be built up, so the application of AI will commonly start small and be improved step by step. In this chapter we provide three ways to get started;

1. Use existing, ready-to-use AI testing tools
2. Use AI services and link these to your own testing tools
3. Developing AI applications yourself to support testing

Considerations

Chances are that eventually several AI (test) applications work together within the testing process. The possibilities are very diverse, therefore it is very important to consider what is the best place to start. In addition, we see three main questions that you can ask yourself.

What is the purpose?

Although in this period of hype it is attractive to 'do something with AI' or to use machine learning, it pays to think in advance what you would like to achieve with it. As described in the previous chapter, it appears that the most gains can be made in these topics:

1. Make your existing test scripts flexible and robust
2. Generate many automatic tests to find more errors
3. Ensure comprehensive unit testing
4. More efficient testing; the right test at the right time
5. Support for the testing process
6. Better reports and dashboards

Once there is an idea as to what goal should be achieved, then the next question is; do I need AI and machine learning for this? Can I not achieve the same with a simple query or a pivot table in Excel?

The goal may also be that you just want to see machine learning work in practice. In that case, the trade-off between costs and benefits is not relevant.

What is your current situation?

Adding value is done from where you are now. Maybe you already have a set of Selenium test scripts. Or your organization is entirely based on Java. Is the application you want to test a mobile app? AI of this has a major impact on the testing tools that you can use, because many AI testing tools - certainly the smaller ones and therefore cheaper ones - only work in specific situations.

Furthermore, the available technical knowledge determines what is possible. If this is not available at all, a testing tool from a supplier is more obvious. If you have programming knowledge and some basic knowledge of machine learning and statistics, then you could get started with DIY or AI services.

The third consideration is to what extent there is trust in AI, both trust by yourself and by your colleagues and management. What is the most convenient place to start, so that you immediately have added value and, moreover, it removes the initial hesitation? This determines whether you can get started ambitiously or whether you will first have to do some careful experiments. In the latter case, purchasing a license for an existing AI testing tool might be a bridge too far.

How much room is there for something new?

Whether you start big or small is of course also a matter of budget. Standard testing tools help you get started quickly and probably require less maintenance, but they have direct costs and may make sure that you learn less from it yourself, so that you ultimately apply it less effectively. The latter is the time component; for how long can you experiment before results have to be achieved.

A good way to create available time is to run the AI-supported tests in parallel with the existing test solution. After a start-up period, it should become visible that the AI variant approaches and ultimately surpasses the effectiveness of the existing test solution.

Use existing testing tools

In the summer of 2018 we (members of the workgroup) started looking at testing tools with AI, talked to tool vendors, studied websites and followed webinars. Quite some time has passed and the tool vendors have not stood still. New testing tools have also come on the market. This chapter is therefore by definition a snapshot, we recommend that you conduct further research based on the information in this chapter.

We can organize testing tools in different ways, by test type, by how they work, in which environments they can run, how useful they are. The tools included in this chapter all have a real AI component and are applied daily in practice or have the option of doing so soon. The appendix of this document contains an extensive overview of this data.

From now on we will mention the name of the testing tool, but we mainly focus on the types of applications that we see.

Dashboards

Many testing tools with AI contain dashboards with overviews of test runs, the quality and what testing has taken place when, such as Eggplant.

A testing tool that specializes in this area is Sealights. Sealights is a test management tool with dashboards. They use AI to create information about the optimal test coverage. This AI is based on program code; testers conducting exploratory testing must record their work on the code folders and thereby record the coverage. It looks at which part of the code produces the most bugs. The aim seems to be to use very economical testing and perhaps to be able to predict in the long term where potential problems might arise.

Record playback as a basis for AI testing

There are quite a few testing tools that use record playback as a technique to test a web application. As a tester you record scenarios and the testing tool matches all elements, such as a button, a picture or a text field on as many things as possible, such as name, position of the element in the page, tab order, etc. That makes the test more stable.

The included test scenarios form the baseline, the basis for the tests. If a deviation is found in a subsequent run, the tool will indicate this. Often in the form of a visual comparison of the old and new situation. The tester checks the difference and indicates whether this is an error or a correct change. If the latter is the case, the tool records the change in the baseline. This allows the tool to learn over time what a correct and incorrect change could be.

These tools make it possible to choose the scenarios that you want to test. Testing tools that work in this way are ReTest, Testim and Mabl.

Crawlers

There are testing tools that go through the application like a spider and search for paths and then crawl though them. These tools generally cannot handle large applications and are suitable for apps or small web applications. The tools work online and sometimes also have a large platform to test the apps on iOS or Android or both. Every tool has its specialty.

Test.ai specializes in web shops. Because of this specialization it is possible that this tool can test web apps autonomously.

Sofy does more than the above tools and calls itself a test assistant. The app and paths are visually represented by a sankey diagram. In addition, Sofy detects which framework is being used and actively searches for known bugs with the framework on the internet at frequently used sites such as stack overflow. Bug reporting is at the touch of a button and a reproduction path and automatic test are added to the bug reporting tool.

New tools are AppTest.ai and Testrigor. Testar is a tool with which you can detect crashes, hangs and errors. You can expand it yourself by adding and programming checks. The intention is that the tool is further developed, so that the tool makes a model of the tested application and thus displays the most important tests.

Visual testing

There are tools that specialize in discovering differences between images. In the past this didn't work well because a tool often stopped due to a pixel difference. Applitools is a tool that tries to deal with visual differences between two test runs in a smart way. The idea is that the tool can eventually predict which change is correct. You can use this tool as an add-on to the existing tests.

Existing tools adding AI

There is a group of existing testing tools that add AI as an extension. Often these tools already have functionality such as functional, security or performance testing in their repertoire.

Functionize is such an existing tool that uses machine learning. They try to capture elements in a webpage in many different ways. Not just one identifier, but much more. The tool checks for all these identifiers and gives a score. With that they can try and figure out where the possible cause of an error lies, the root cause analysis. It is possible to use production usage monitoring. They have an extensive tool that processes recorded data from production to test data. This test data is managed in the tool and pushed to the test environment.

Appvance announced on its website that it offers similar functionality. They use external tools for recording user sessions and process them into test data and tests. EggPlant uses its existing tests for an extension with AI. As a user you have to map all fields and map the application. Eggplant then goes through testing with this.

These tools, just like the other tools, have a visual interface that makes working with the tool easier.

Code testing

Diffblue is a tool that can generate java unit testing based on java code. This can be useful if there are few unit tests on a large code base. They use AI for this.

How intelligent are these tools?

Assessing a tool is quite difficult, partly because the tools differ, but also because few tool vendors are very explicit about it about the inner workings and the word ‘AI’ is just used for marketing. Also, it matters what the tool is used for and what problems the tool should solve. Is a tool used as an extra tool on existing tests, or is the tool used for the total automation of the app or application?

Use existing AI services to enhance testing

All major cloud computing providers now offer cloud-based AI products. They are mainly large companies, because setting up these products properly comes with high costs. These services consist of both software and hardware. Together they ensure that you can send data to a cloud service. This service processes the data using pre-trained AI models and then sends you the results.

General applications

The possibilities are quite extensive. Wherever interpretation needs to be done and where standardized outcomes are possible, standard AI services can be used. For instance:

- Editing and analyzing data
- Analyzing visual material
- Converting images to text
- Analyzing text
- Converting text to speech, or vice versa
- Translating text

The condition is that large amounts of words and their coherence (or language) have been trained beforehand or that large amounts of images have been trained on the basis of labels. But this is exactly what cloud services are good at, they are trained with millions of texts and images. Offering your own words or images therefore ensures rapid recognition.

Applications in testing

The use of these techniques in testing may not seem obvious, but there are also several options there. How can we use this in our testing process?

- Upload data that is then converted to fill dashboards
- Upload bug reports to let the AI predict where future bugs will occur
- Upload code coverage, defect report and code changes, to let the AI determine which tests should be executed
- Upload software requirements and let the AI generate test cases
- Upload digital documents and use AI to structure and organize them
- Let the AI search for objects on the screen

An AI system requires a lot of data during the training session. Even when using standard trained AI models, you will have to train on your own examples.

The current main providers of these services are IBM (Watson), Amazon (AWS), Microsoft (Azure), Google (Cloud AI), Oracle and Salesforce.

Build your own AI

In addition to conducting testing tooling with AI, or standard AI services, it is also possible to develop AI components yourself to enrich your testing activities. This is particularly useful for long-running testing projects where test automation has already been set up. Much has already been invested in such processes and it is not realistic to switch to new tooling and technologies. Self-built AI components have the advantage that they can be modeled on the existing automation solution. As a result, they can help solve very specific problems. Moreover, it is not necessary to purchase expensive packages and tools for DIY: there are countless open source libraries available that allow free experimentation. Below we explain a number of important steps and considerations globally.

Problem definition: do I need AI?

First of all, it is important to formulate the problem that you want to solve as clearly and detailed as possible. The next step is to thoroughly check whether AI technology is the best way to solve it. The operation of AI is different (read: much more data-driven) than ‘regular’ programming and scripting, but this also entails completely different complexity. Data is the keyword. In their online training 'Machine Learning - Problem Framing', Google applies the following rule of thumb: successfully training a simple model requires a few thousand samples of usable data, complex models (such as neural networks) quickly require one hundred thousand samples, or more. If you don't have that much data available, first consider a ‘classic’ rule-based solution.

Data selection

Data availability is therefore decisive when implementing AI functionality. In theory, we generally have no shortage of data automation in test automation; just think of all test and system logs, error reports, test data and evidence that is released during an average test run. The key question to begin with, however, is: how can we combine data from such sources in such a way that it says something about the problem and points in the direction of a solution? The more specific and narrower the scope of the problem is, the greater the chance that current AI technology can offer a solution: whoever wants to 'slim down' his test set on multiple fronts will need a data structure with corresponding AI model for each specific task. Setting up. Building an ‘all-knowing’, multi-tasking AI test assistant is certainly not within reach within the framework of self-building.

Data preparation for training

After mapping relevant data, it is important to put it together in a data frame - something you can see as a large table that can be read by an AI algorithm. Columns here represent the properties - features - of the data and the rows are individual samples. The 'Orchids' data set, the equivalent of 'Hello World' within AI / Machine Learning, is a nice illustration: the purpose of this AI functionality is to be able to determine the type of orchid, based on features such as the length and width of the sepals, and length and width of the petal.

The samples contain specific dimensions of the different leaves. Since algorithms 'think' in numbers, the conversion of all kinds of data (including text and image) into numerical values is an important step in shaping the data frame, also known as 'feature engineering'. Fortunately, we don't have to reinvent the wheel for this: almost all data science libraries contain tools for such conversions. The same applies to the AI / Machine Learning algorithms themselves, which can also be downloaded in libraries. Packages such as scikit-learn for Python and WEKA for Java contain out-of-the-box algorithms for classification, regression and clustering. A framework like TensorFlow is multi-platform and offers functionality for more advanced deep learning with neural networks - and all that is open source. If you want to make it really easy for yourself, consider Anaconda (www.anaconda.com): a one-stop package for Python that already includes all the necessary libraries and useful editors.

Training the model

With a data frame available and the necessary tools installed, real experimentation can begin: choosing an algorithm and training, testing and adjusting. It cannot be emphasized enough how important the quality of the training data is: the quality of the data has a much greater percentage influence on the accuracy of the trained model than the type of algorithm behind it. It pays to try different algorithms, because they all have their strengths and weaknesses. However, is it not possible to train the model accurately enough, regardless of the algorithm? There is a good chance that it will be a issue in the training data - and time to take a closer look at your feature engineering.

Acquiring AI knowledge yourself

As the sections above show, the test automation engineer who wants to work with AI will be dealing with data science, which is a profession in itself. Data science tooling such as the libraries mentioned earlier are open source, but contain numerous possibilities that are worth studying. Fortunately, there are excellent tutorials, lessons and courses available online these days that can also get us up to speed relatively quickly when working with AI technology - often free of charge and easily manageable.

The Machine Learning training courses from Kaggle[3] and Google[4] in particular are worth considering as a first step: in addition to the technical aspect, these training courses also pay attention to formulating cases for AI and preparing data. In addition, all exercises in Google's course run online through your browser, so installing tools is not necessary. Other interesting sources and training courses can be found in the appendices. The knowledge and tools for building AI components yourself are therefore readily available: are you the creative tester who will design the latest implementation for our field?

[3] Kaggle is a platform where machine learning competitions are organized. However, it is also an accessible one community, where newcomers can find training material to participate in the competitions later. https://www.kaggle.com/learn/
[4] Google is a well-known name in the AI world and makes this course available to everyone: www.developers.google.com/machine-learning/crash-course

6. A fictional but specific example

For this chapter, we will use an example a test object that consists of an online input system (via GUI) and some transaction processing in the background. The tester wants to support the activities with AI testing tooling.

The GUI is tested first. An existing self-exploring testing tool such as Testar can be used for this. The testing tool performs various tests in which the quality of the GUI is assessed. For example, whether buttons are logical and if there are no ‘broken links’.

The tester then uses a self-built AI-based testing tool (using standard machine learning algorithms) to do an analysis of the production data in order to generate synthetic test data on the basis of this that accurately captures each relevant unique situation from the production data. has a limited but optimal set of test data.

The next step is to generate test scripts with an AI-based testing tool that can enter the test data into the test object and then execute these scripts and evaluate the result. This assessment in particular is a complex activity that requires a great deal of knowledge, insight and experience. It seems likely that the AI testing tooling only makes suggestions for the time being and that the tester makes the final pass / fail decision.

As a final step, the result of the assessment is processed by a reporting tool that automatically displays the information for the various groups of stakeholders in an optimal presentation form for them.

At the time of writing (May 2019), not all of the tools outlined above are actually available. But at the time this workgroup was created (January 2018) we were unable to find any of these tools, so the developments are going very fast.

7. Future of testing

AI and machine learning are in full development. The technology is becoming increasingly powerful and more appealing applications are appearing. We also see this in the testing profession. The AI testing tools are becoming more advanced and frameworks for developing something themselves are becoming more accessible.

This leads to the question of what role the tester will play once the AI applications are fully developed within the testing area. In other words, ‘does AI take our job as a tester?’ Hopefully, the whitepaper has made it clear to this point that we should not be afraid of the latter.

It is clear that the testing profession will change with the arrival of AI, just like the current test automation has changed testing. And this will also affect the role and skills of the tester.

How will testing change?

Working with self learning testing tools

Although trained AI testing tools will take a lot of work off your hands, we will have to find a solution to the fact that trained models will not behave 100% as expected, as opposed to programmed testing tools. After all, machine learning learns from examples, rather than following fixed rules. This means that the testing tools and our own configuration must also be tested. On a case-by-case basis, we will have to assess whether the AI testing tool's testing effort outweighs the benefit that the AI testing tool brings.

In addition, test results must be interpreted differently. A traditionally programmed testing tool crashes if a specific object is not found. An AI testing tool will likely find a detour to complete the test and will report that ‘the test has passed 87%.’

Multiple options for tools

We asked the vendors of tools about their future vision of tests and tools. We asked them where they now think it's going. Naturally, the people we spoke with are convinced that the use of AI testing tools will increase in the coming period. But we also gained some additional insights.

Trends are that tools with AI will be used alongside traditional tools. Tools with AI are currently specialized for one specific task. In the future, it will become more common to use multiple testing tools in addition to and with each other and to have a range of tools that covers the total range of test activities.

Another trend is that smaller tool suppliers will offer their services together. For example, tools that can explore the application combine with tools that are good at smart image recognition. And possibly also that a tool such as Selenium IDE can be used in combination with one or more AI tools.

The opposite is specialization in a way test.ai now applies it. This tool focuses exclusively on web shops. The goal is to offer a testing service for web stores that can work completely independently in the long term. There are tools with AI that specifically target the app market. It is also important for apps to be able to run quickly and on as many platforms as possible.

With larger testing frameworks, people are also starting to get interested in offering AI. Functionize, Appvance and Eggplant are examples of American test platforms that have been doing this for some time and each in its own way.

Testing becomes more creative

With the arrival of tools such as Selenium, repeated tests were removed from the tester and manual testing was able to focus more on creative testing situations. This is reinforced if AI testing tools take over even more of the standard tests from us.

Moreover, AI will help to write and perform many of the (technical) unit tests, so that the tester can focus more on functional and more complex non-functional tests. The manual, creative testing shifts to verifying requirements and delivering value to the organization.

How will the role of testing change?

The work of a tester will continue to exist. However, the work required to provide insight into quality and risks will change. This will also change the required knowledge and skills. We see the following developments.

Clear formulation of objectives

This whitepaper has shown that an AI can support us in all kinds of points in the testing process. There is even a possibility that the AI itself will be exploring autonomously. This means that we must clearly state what we actually expect from the application and what we would like to have tested. This means more attention to requirements, acceptance criteria and test goals. Because in the most extreme scenario, the AI could explore without any guidance and we would spend months evaluating the results. A good set of expectations in advance can prevent this.

However, this does not mean that we must stop exploring the application ourselves, for example through exploratory testing. It might become even more important; the human tester does a good exploration of the playing field, we leave the large-scale and repeated testing to the AI.

Managing the uncertainty of the testing process

As the first section of this chapter showed; applying self-learning AI (or machine learning) ensures that we do not get absolute 'right' or 'wrong' results. As a tester, we will have to do our own interpretation of a certainty percentage that the AI returns. So if the AI finds a test case 87% successful, how do we report this and what follow-up action will we take? This will be different per project, which means it is up to the human tester to determine this. Quite often, the tester will step in and perform some tests manually.

The test manager will also experience this uncertainty. If most of the tests have passed at least 90% and some tests have passed at 75%, are we done? This is an interpretation that the test manager must do and nicely returns to the previous theme; make sure there are clear expectations in advance. Moreover, it requires the ability to interpret large numbers of results. And that brings us to the next point.

Knowledge of statistics

Interpreting large amounts of results and certainty percentages requires the tester and the test manager to have some statistical knowledge. Even if ready-made tools claim to be happy to do the interpretation on your behalf, it is important to have a well-founded opinion about all figures that an AI returns. Should the tester and test manager then become a part-time data scientist or data engineer? That's hard to say, but some basic knowledge will certainly help.

Better knowledge and understanding of testing tools

A very diverse set of tools is available. Regardless of whether they are 'real AI', they all have different ways in which they contribute value, in different environments. We already see the trend that a large number of tools are linked together to achieve the test goals. AI-supported tools will be added to this.

The expectation is that the automated part of the testing process will increase, compared to manual test and test support activities. Knowledge of which tools you can use, in which combination and especially for what purpose, becomes even more important.

In other words: there will always be a role for someone who thinks about quality, who asks critical questions and who takes care of the communication about goals and priorities. To quote Jeremias Roessler[5]; the chance that we will one day automatically *build* software is greater than that we will *test* it fully automatically. Because when an AI builds an application based on a model itself, it is important that this is *not* tested according to the same model.

[5] https://www.sealights.io/webinars/when-will-ai-take-my-job-as-a-qa-manager/#fvp_WhenWillAITakeMyJobAsAQAManager

Testing óf an AI

This whitepaper deals with testing with AI, not testing the applications that make use of AI. Up to this point we have therefore avoided testing óf AI as a subject. Nevertheless, we have to mention it briefly here, because this has an impact on testing with AI.

- Testing of an AI application will ensure that data plays a more central role in our testing processes. When training and testing a customer application with AI, we will have to consciously choose the data used. The required data of a test case is as wide as the number of variables on which the model has been trained. We can then use AI to determine whether we have the correct data, or even generate it via AI.
- Accepting the end result will be strongly based on the amount of right and wrong predictions that the AI application makes. This means that statistics will play an important role in the test area, in addition to the number of successful and failed test cases that we have used as an important metric.
- Manage more on the process side; in a project where AI recognizes patterns in data, there is constant interaction between the modellers who optimize the model and people with domain knowledge, who assess the outcomes and exceptions. Applying AI smart dashboards can help to make this complexity transparent and can help to plan the right quality controls.
- Broadening the view on quality: as a tester we will also have to include ethics and morality. In addition, we must keep an eye on unwritten / common sense requirements, because an AI has no context.

This whitepaper is just the beginning

While we were working on this whitepaper as a workgroup, testing with AI continued to develop rapidly. We have rewritten some parts five times. This whitepaper describes the ‘state of the art’, but is therefore absolutely a snapshot.

The workgroup enthusiastically continues to follow this field. The testing of AI and the combination with testing with AI will also continue to receive attention via the TestNet workgroup.

Do you want to join us in following and shaping these new developments? You are very welcome! Contact werkgroepen@testnet.org and we will be happy to work with you. Do note that our main meetings are in Dutch, but there are still plenty of ways in which we facility community members who prefer English.

Appendix 1: sources

AI and testing in general

- Jason Arbon has written many different articles and gave webinars and interviews.
  https://www.centercode.com/blog/2019/04/ai-and-the-future-of-testing
  https://www.linkedin.com/pulse/links-ai-curious-jason-arbon
  https://huddle.eurostarsoftwaretesting.com/resources/artificial-intelligence/ai-will-soon-test-everything/

- Test talks with Joe Colantonio
  Many interviews about testing with ai. An example:
  https://www.joecolantonio.com/testtalks/178-third-wave-test-automation-joe-colantonio/ from 2017

- Book about both testing with AI and testing of AI, including about evolution towards forecasting based on information from tests and monitors:
  'Testing in the digital age; AI makes the difference', 2018, Tom van de Ven, Rik Marselis, Humayun Shaukat. ISBN: 978 90 75414 87 5

Testing tools

- A number of existing tools
  See especially the links in appendix 2
  https://www.ministryoftesting.com/dojo/lessons/ai-in-gui-based-software-testing?utm_source=linkedin.com&utm_medium=social&utm_campaign=artificial-intelligence-ai-and-machine

- DIY
  Machine learning library Weka: Getting Started with Weka - Machine Learning Recipes #10

Opinie/Visie

- Computable
  https://www.computable.nl/artikel/nieuws/overheid/6630922/250449/kabinet-komt-voor-de-zomer-met-ai-actieplan.html
- EuroSTAR
  https://huddle.eurostarsoftwaretesting.com/title-will-ai-kill-software-testing-heres-wont/
- NRC (Robbert Dijkgraaf)
  https://www.nrc.nl/nieuws/2018/10/19/gevangen-in-onze-verbeelding-a2672745
- Salves
  https://www.testforward.nl/newsroom/#artificial-intelligence

- Sogeti
  https://labs.sogeti.com/to-test-is-human
- TED
  Grady Booch: Don't fear superintelligent AI
- Tegenlicht
  https://www.facebook.com/37586539901/posts/10156280444544902/
  https://www.facebook.com/37586539901/posts/10156289342059902/
- TorontoTech
  Chris McKillop of turalt presents Email, empathy, ethics & AI

Appendix 2: overview of tools

Below you will find information about tools in two tables. The first table gives an overview of what the tool can do for the user and what not. The second table shows what the web address is and for which type of application / framework the tool is suitable.

Tabel 1

Name of the tool	**what does the tool do for the user?**	**what should the user still do?**
applitools	Visual validation, recognizing smart visual differences, recognizing objects. New is applitools with selenium IDE.	Make tests with recordings. Validate deviations with the baseline. Add tests to existing tests with other tooling such as Selenium.
appvance	Test framework with functional, load and performance tools. The tool generates test cases for automated tests based on recordings from production or from log files. Analytics on dashboards.	Recording the baseline and validating differences.
functionize	Test framework with functional, security, load and performance tools. Functionize uses data from production logs and functional tests. They process this data in a tool (adaptive event analysis system). This tool makes the data ready for models. They push the models to the Elastic Cloud environment, where the tests are run on all possible platforms. They are able to assess static and dynamic scenarios with ai. Analytics on dashboards.	Starting and managing test sets and runs. Add testing. Validate differences found by the tool.
eggplant	Test framework with functional testing. If the application is mapped in the tool, it can run existing functional tests on it (made with Eggplant functional. Can search its own way and generate tests. Choose test cases / scenarios.	User must visualize the application. Assessing test runs. Adding tests yourself is possible and tests also give a higher rating so that the chance that they are run is greater.

	Analytics on dashboards.	
testim	The AI tool analyzes all DOM objects of a page and extracts the objects and their properties. Based on this, the best location strategy to locate a certain element is made. This makes the elements more findable and tests more stable and, the test continues to run. Testim keeps scores of elements in the GUI. If these objects do not change or do not change much, they get a different score than objects that change a lot. (tests that fail / succeed) Analytics on dashboards.	Make tests yourself.
mabl	Mabl collects more than one identifier for each test step. They say that they can easily handle small changes in the application without breaking the test. The test shows that by showing auto-healed, which you can also turn back if the result is incorrect. Differences are visible at every step through screenshots. Machine learning extracts the differences in the static parts of the application	User must make tests himself with record playback, add assertions etc.
retest	Run all tests again with the tool and with the chosen intensity. (Monkey test.) A report is automatically generated based on this test. The tool also looks at as many features as possible in the DOM and tries to match smartly with and without AI.	User records tests with the visual testing tool.
sofy	Calls itself a test assistant. Testing apps runs through the app itself. Deviations can be copied to Slack or jira at the touch of a button with a generated test and written reproduction path. Looks for problems with known mobile frameworks on stack overflow and reports on them with possibly workarounds.	User assesses the test runs. Creates bugs with the tool.
testrigor	The tool goes to work independently and goes	Assessing differences and adjusting the generated tests.

	through the pages and recognizes input fields and presses buttons. After a new deployment of the application, a retest can be done.	
test.ai	Tool for web shops. Walks the paths in the app and tests them. Checks for visual changes and assesses them. Ensures that testing continues to run and does not break in the event of a small difference.	Assessing differences in test runs. Add tests.
apptest.ai	Testing of Android apps. After uploading the.apk, the tool crawls through the application. Can be run on mobile devices of choice, real or emulators. The software delivers the results in five categories: - Summary is a summary of device info and test results; - Activity Map provides an extensive visualization of all pages with links and connections; - Screen is a snapshot of all app pages with highlight on action button and other available button (s) - Performance provides a graph of CPU and memory during testing; - Log is a list of tag, message and bug options.	Assessing differences in test runs / devices.
testar	It is a tool with which you can detect crashes, hangers and errors. You can expand it yourself by adding checks and programming. The intention is to further develop the tool	Starting the tool. Check which errors have been detected. The functionality is still limited.
diffblue	Works on the programming code in Java. Scans the code and creates unit tests where they are most needed.	User is developer. This must assess and run the tests.
sealights	Analysis platform for coding and testing. Philosophy: economical testing, covering risks. Dashboards and	Tool provides insight. The tester is expected to perform manual testing based on the changes in programming code,

	overviews provide insight into which tests need to be done, both automatically and manually. Tool also includes a commit assistant that prohibits checking in code without testing.	so that the tool can register it.

Table 2

tool website	tool suitable for	webservice / local
applitools.com	as an add-on to existing tools, works on many different systems and development environments Applitools is new on IDE.	SDK installation communication with web service. To get it working, some coding is needed, examples on the website.
appvance.ai VS	Web applications and apps	webservice
functionize.com VS	Web applications and apps	webservice
eggplant.com (voorheen testplant) VS	Various parts of the tool are aimed at management, apps, network, testing and generating testing.	local installation on server
testim.io VS	Chrome extension record playback. Web applications and apps (beta)	Webservice Right now you have to start testing again if you want to use Testim. Work is being done on integration with other existing tools such as Selenium.
mabl.com VS	Chrome-extension Record playback.	Webservice
retest.de Duitsland	Applications written in Java. Work is being done on a tool for (web) apps.	Local installation with local ai. NB an AI can be downloaded in addition to an example tool.
sofy.ai bedrijf Quantyzd VS	Sofy focuses on apps on IoS and Android.	The app is uploaded to the cloud and sofy gets started.
testrigor.com VS	Web applications and apps Intended for small applications and apps.	webservice
test.ai VS	Webshop apps and application	webservice

apptest.ai Zuid Korea	Apps Android	web service, upload your app to the apptest.ai cloud
testar.org NL/SP	The tool works on Windows, MacOs and Android. The tool uses the OS and requests widget trees from the OS api of a target application.	Local installation with download
sealights.io Israel	Plugin on the CI tool does dynamic analysis of the build, testing etc.	webservice
diffblue.com Qxford GB	The tool can be found at https://github.com/diffblue. Work is being done on supporting other programming languages.	