How to build great AI

Even though artificial intelligence (AI) is in full bloom and more and more businesses are developing their own solutions, there are several things around it that aren't quite clear. One of the most pressing ones is what constitutes a good AI. Though there are neat examples of AI-based software showing huge potential (IBM's Watson first comes to mind), there still isn't a precise recipe to end up creating good AI - let alone great AI!

Maybe the reason that has software development companies and teams feeling somewhat confused is how new AI algorithms are. Perhaps it's AI's staggering potency or sophistication. It could be that there's a lack of standardization that makes it easy for developers to quickly derail in AI development.

This doesn't mean that software development services have to start from scratch and experiment all the way through. If experience serves right, then there are certain basic steps and considerations anyone can take into account that will lead to great AI solutions more often than not. It's all about how you approach it.

Understanding the basics

If you are one of those people that think of artificial intelligence and imagine a powerful sentient machine capable of replicating every human thought, you're not alone. Yet, while we are definitely going towards that horizon, today's reality is far more pragmatic. In other words, you won't be able to build a human-like software but you'll be able to develop a powerful AI-based tool if you take the following 3 aspects into account:

> **Purpose:** all great AI tools begin with a purpose. Why do you want to create this tool? What's its main goal? It can be a lot of things, so the answer will depend on your industry, your niche, your processes, and your customers. Maybe you want to develop an efficient chatbot for your clients or automate your calls. Maybe you want to include a predictive solution to maintain your supply chain! Be it because you're imitating a use you already saw in another company or an entirely new one, you definitely have to start with a purpose. Otherwise, your tool won't make any sense.
>
> **Operation:** the trickiest part is to define how your AI will learn, from which data, and what will happen with that information. For instance, you can work on an AI tool that monitors the supply chain to aid with maintenance. You can program it to control the wear and tear of specific components and determine that, once they get to a certain point, the program has to trigger a warning and schedule maintenance. Naturally, you'll have to feed the AI with certain starting points and rules (what constitutes wear and tear, when does it have to report it, what to do with certain situations that weren't defined) and program it in a way that it will learn how to write new rules based on new information over time.

Tools: Once you have all of the logical parts surrounding your AI solution, it'll be time to decide how will the software work with the data. In other words, this is the part where you have to define the proper language, frameworks, APIs, encoding systems, and everything that will play a part in the final tool. If you've done your work correctly and were detailed enough in the 2 previous aspects, then the choices here will be crystal clear and your job will be easier.

A 4-step journey towards great AI

The 3 aspects above implicitly delimitate a roadmap that might take you and your software development providers towards a successful AI project. In fact, following the next 4 steps might give you the conceptual framework you need to build a great AI tool.

1 - Define the problem you want to solve

As said above, the first step is always to identify the need you want to address with your AI solution. From automating your communications and predicting failures in your supply chain to anticipating consumer demand and offering smart recommendations, there's a wide array of possibilities here.

That's why it is so important for you to be as detailed as possible when defining your AI's purpose. For instance, it won't be enough to say "I want to build a chatbot." You need to define what kind of chatbot you'll want, where will you use it, what kind of assistance will it provide, and if it will work alongside other tech solutions you might have in place - or might develop in the future.

2 - Identify the type of data you'll collect

AI software works with some sort of input. The data you'll use as such will certainly have an impact on how you'll develop your platform, so it's also essential to determine which kind of data you'll be using. Though there are massive amounts of data you can gather through digital and offline channels, data in itself can be classified into two big groups: structured and unstructured data.

Structured data is the one that can be easily recorded into a field within a set of options. Examples of this data include names and surnames, dates of births, ages, addresses, and any other quantifiable information that can be easily classified into one specific option. Unstructured data, on the other hand, isn't that easily classifiable and doesn't have a defined form. This includes images, audios, videos, files, emails, documents, and the like.

Whether you use structured or unstructured data (or both) will define the algorithm's logic and how they will deal with the information collection and classification. It's not the same to program a solution that has to read quantifiable information to define an action than creating an AI software capable of "understanding" a text's intent or an audio's "mood."

3 - Picking the algorithms and the programming language

Now that you have all planned out, it's time to build the AI solution itself. Though you might be tempted to build everything from scratch to better fit your needs, you definitely don't have to - and maybe you shouldn't do so! Why? Because there are some powerful AI platforms out there than can be integrated into your own software that will save you from all the work. Examples include IBM Watson, Google Cloud AI Platform, and Azure Machine Learning Studio, among others.

Of course, those platforms will provide you some tools you can use to integrate sophisticated algorithms into your own tools. For that to work at its best, you'll have to pick one programming language that fits what you need. Here are some of the most popular options:

Python: given its syntax simplicity and short development times, many development teams use Python to create AI solutions. The fact that this language has plenty of libraries created specifically for AI and machine learning only turns into a better choice for all kinds of AI projects.

R: the power of R resides in its efficiency at analyzing and managing all kinds of data for statistical purposes. The existence of packages such as Gmodels, Class, and RODBC for machine learning makes it easier to implement powerful algorithms that allow reaching stronger AI products in shorter times.

Lisp: it's odd that one of the oldest languages around is such a good fit for AI. However, Lisp has consistently proven to be of great assistance when prototyping, especially thanks to its interactive evaluation of expressions and the recompilation of functions while the program is running.

Prolog: this language's eligibility for AI projects is based in 3 main features - efficient pattern matching, tree-based data structuring, and automatic backtracking. The combination of the 3 provides this language with enough flexibility to create advanced AI systems.

Java: easy to use and to debug, simplified work for complex projects, graphical data representation, and package services are among the features that turn Java into a great alternative for AI programming.

Any of these 5 alternatives are great to create an environment that can perform the advanced functions you expect from your own AI solution. Of course, you'll have to go through the last step first.

4 - Training the algorithms

After picking the language and platform that better suit your needs, you'll have to train the algorithms to work accordingly to offer the results you are looking for. In other words, you'll have to feed the type of data you defined and create the rules necessary for the algorithm to understand what to do with it. Training is a time-consuming task that will take a lot of effort.

That's because the training needs for you and your developers to refine your platform's model accuracy. This is a vital concept for AI solutions, as it defines how trustworthy your AI will be. Basically, model accuracy is the percentage of predictions the AI algorithm got right. If you were working in a tool that predicts when a supply chain component will break down, then the number of times it does so correctly divided by the total number of predictions will determine its accuracy.

Understanding that will let you create a minimum acceptable threshold for the AI to start working. After that, the training can go on naturally in real case scenarios.

Some final words

Creating great AI is a complex challenge. It involves knowing which languages to use, how to implement the available platforms, and how to conceptualize these kinds of projects. There's a reason why so many companies that want to tackle an AI-based tool resort to the outsourcing of software development - they are looking for experts and professionals that have the necessary expertise to create these products.

However, this doesn't mean you can't develop your own solution or that you'll necessarily have to use software development outsourcing to get there. On the contrary, you can use a lot of the tools readily available to come up with your own platform. The recommendations above will bring you closer to success, though you have to keep in mind that they are just the beginning.

Building great AI will have you conceptualizing, training, and adjusting your algorithms until you reach a satisfying end product. As challenging as that sounds, it's the only way in which you'll benefit from this tech.