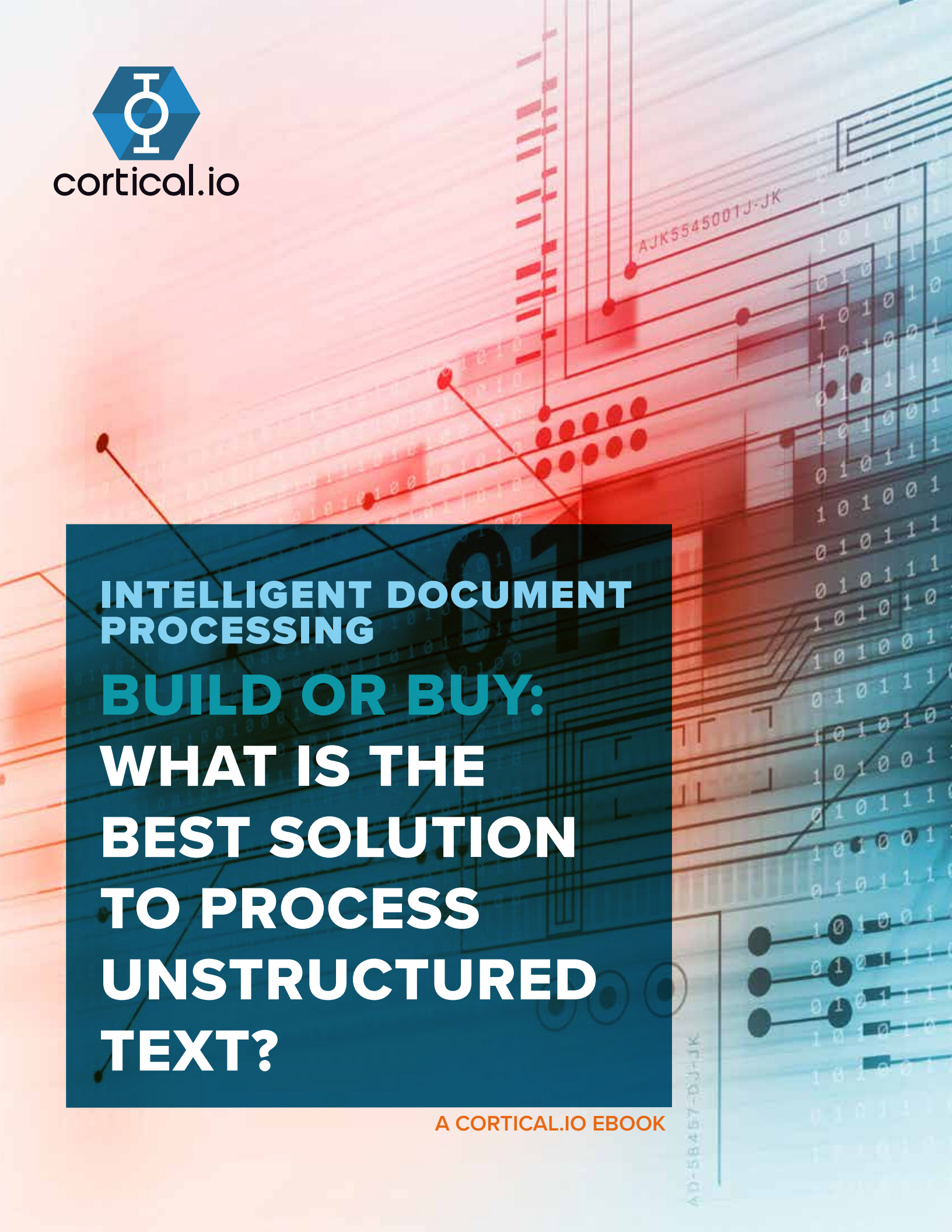


INTELLIGENT DOCUMENT PROCESSING

BUILD OR BUY: WHAT IS THE BEST SOLUTION TO PROCESS UNSTRUCTURED TEXT?

A CORTICAL.IO EBOOK

5 KEY TAKE-AWAYS

1. Building out in-house NLP capability is expensive. The costs of NLP development can be broken down into roughly four categories: personnel, infrastructure, time, and data.

2. Qualified machine learning professionals are actually quite rare and highly sought after. For a bare-bones NLP team, businesses are already looking at yearly expenses in the range of hundreds of thousands of dollars.

3. Many models can take days, weeks, or even months to train and require very fast machines with expensive GPU or TPU hardware. Even if enough training data can be found to fine-tune the transformer model, its capacity to successfully solve a specific NLP task is not certain.

4. Buying a pre-built IDP solution to save on costs is only a good option if your use case exactly matches those foreseen by the vendor.

5. For most business use cases involving unstructured text with highly variable content, you will want to look into a highly flexible IDP solution that is easy to customize without requiring thousands of annotated data and delivers high levels of accuracy on unstructured text – something which Cortical.io's novel approach to natural language understanding makes possible.

INTRODUCTION

Intelligent Document Processing (IDP) refers to the process of using computational software and tools to extract meaning from text. Text comes in three different flavors: structured, semi-structured, and unstructured.

Structured text is so named because it possesses some sort of reliable schema. Examples include PDF forms with predefined fields and HTML or XML files with a predefined set of tags. In contrast, unstructured text lacks a strong structure and tends towards being free-form. Performing IDP on unstructured text might involve extracting relevant case law from a legal text or the pertinent details of a legal contract. Finally, IDP applied to semi-structured text entails the analysis of documents that have both structured and unstructured parts.

Structured and semi-structured text processing is largely a solved problem as it primarily involves using properly-configured, rules-based mechanisms to search through documents and pull out the relevant information. **However, unstructured text extraction and classification are much more complex problems which a rule-based system can't solve**. Instead, they require the use of techniques such as natural language processing, machine learning, artificial intelligence, and some creativity to discover contextually significant information. As such, this white paper will focus on IDP applied to unstructured text.

If you need an intelligent document processing solution to take your business to the next level, chances are you're familiar with the headache involved in researching the available options.

Between off-the-shelf solutions, custom products, and the prospect of building your own tool, it can be difficult to assess the costs, benefits, and tradeoffs associated with each approach.

THE CHALLENGES OF (IN-HOUSE) NLP

Some large companies are tempted to build an in-house team of AI and NLP experts to develop their own IDP application, thinking that doing so can save them money, allow them more customizable solutions, help them to achieve greater accuracy, and myriad other reasons.

While building your own Natural Language Processing (NLP) solution can be a great adventure, it is also fraught with multiple challenges related to the difficulty for a system to understand words it has never seen before, the high costs of developing, testing and fine-tuning a model, and last but not least, the time it would take to come near the NLP expertise of vendors that have been doing active research in that year for many years.

Vocabulary Coverage

State-of-the-art IDP techniques rely heavily on a form of machine learning called *Natural Language Processing* (NLP). NLP is a well-developed field that has been around for a few decades. In the early days, its methods were primarily statistical in nature, relying on things like word and n-gram counts, and models built on this data, to extract meaning from text.

More recently, deep learning has transformed NLP and brought with it new and complicated models that have achieved superior performance on many different textual datasets.

NLP as a discipline is *incredibly difficult.* Our brains have evolved naturally over millennia to utter and understand language; however, computers were

never designed with the processing of language in mind.

Human language is unique due to the incredible variety and structure of its vocabulary. The vocabulary of the English language, for example, follows a power law distribution, meaning that a few words are used very frequently (think “and”, “the”, “or”, “car”, etc.) whereas the rest are used relatively infrequently (for example, “walrus”, “verdant”, “churro”, etc.).

Why does this pose an issue for computers?

It’s simple. Nearly all modern NLP methods employ machine learning in some way. At its core, machine learning focuses on training models in an unsupervised manner to extract statistical patterns from data.

In order to do so effectively, a model must have access to enough training data to uncover these statistical patterns.

Usually, it is not too hard to get training data on common words as they show up everywhere.

However, words used infrequently, and especially domain-specific terms as are likely to occur in documents processed in a business context, are difficult to obtain sufficient training data for. As such, most of these words are frequently ignored by models, and this can severely limit the overall accuracy of a machine learning based approach if steps are not taken to mitigate it.

Personnel Costs

Personnel costs are perhaps the most straightforward and potentially the most significant. A lean NLP development team likely requires at least three employees, a machine learning researcher, a machine learning engineer, and a more general software/data engineer.

In general, the researcher would develop the vision for the NLP product, drawing on very technical statistics, ML, and NLP know-how to design a new algorithm to handle a given NLP task. They would also perhaps publish papers on the new methodology. Even if a new algorithm is not required, the researcher would be tasked with surveying the field for the relevant algorithms that apply to a given problem, reading the current literature on the subject, and guiding the resulting implementation.

One thing’s for sure, building out in-house NLP capability is expensive. The costs of NLP development can be broken down into roughly four categories: personnel, infrastructure, time, and data.

The researcher would then generally hand off the specific development tasks to the machine learning engineer, who would utilize their engineering experience to implement the algorithm in code and ensure it's scalable to a production environment. Finally, a data/software engineer would build out the more generic application backend around the algorithm and create code for gathering and warehousing training data. They would also be tasked with implementing any ETLs and preprocessing prior to handing the data off to the model.

For such a bare-bones team, businesses are already looking at yearly expenses in the range of hundreds of thousands of dollars.

Any general software or data engineer worth their salt will demand a salary of at least $100,000 / year, and more senior candidates with 3-10 years worth of experience could easily command salaries of $150,000 - $250,000 / year or more! Machine learning engineers, and especially researchers, will command even higher salaries due to their additional expertise and years of education. All-in-all, a lean machine learning development team will run somewhere in the range of $300,000 - $900,000 per year, and that's only the beginning.

It's important to note that this is the bare minimum in terms of developer expenditure; businesses which are looking to truly take their NLP products to the next level and match the state-of-the-art would likely require a much larger development team. One other aspect to consider is that these costs are assuming that you can even find the requisite talent and attract them to your organization.

Qualified machine learning professionals are actually quite rare and highly sought after. For example, 62% of respondents to this survey indicated that they couldn't find talent on par with the skills requirements needed in efforts to move to AI.

Infrastructure Costs

In addition to personnel expenses, running and training machine learning models requires vast computational infrastructure. Many modern-day deep learning models contain millions, or even billions, of parameters that must be tweaked, and this comes at a high cost.

Luckily, cloud computing services such as Amazon AWS, Microsoft Azure, and Google Cloud have ameliorated this problem to some degree by providing on-demand, scalable, cloud compute power that can be accessed and paid for on an as-needed basis.

However, these services do not come cheap, especially if you need to make frequent use of them for training and re-training models.

For example, one of the moderately sized AWS GPU instances, p3.8xlarge has an on-demand price of $12.24 an hour. That gets you 32 CPU cores and 244 GB of CPU memory, plus 4 NVIDIA Tesla V100 GPUs with 64GB of GPU memory. Depending on your model, you may require a more or less powerful machine or even require running machines in parallel.

But using this as a baseline, say you need to train your model for 100 hours – that's already $1224 for a single round of model training. There will inevitably be errors that need to be corrected, tweaks to hyperparameters that need to be made, and retraining that needs to periodically happen in response to model drift. As such, training costs can quickly add up over time. This is also not even close to the most expensive AWS instance either; for example, the p4d.24xlarge runs a whopping $32.77 per hour.

If you really want to turbocharge your investment, this is just the tip of the iceberg. Google Cloud offers TPU pods with up to 2048 TPU nodes that start at rates of hundreds of dollars per hour. Of course, the machines themselves are not the only infrastructure costs. Frequently, you'll also want to pay for cloud data storage, static IP addresses, as well as servers and load balancers for hosting your models. These all add up to significant amounts over time.

Many models can take days, weeks, or even months to train and require very fast machines with expensive GPU or TPU hardware.

Costs of Data

Data is another substantial expense associated with homegrown NLP projects. Especially now, in the age of deep learning, NLP models for IDP require extensive amounts of data to train. NLP models that are transformer based such as GPT-3, BERT, etc. are notorious for being among the largest (in terms of number of parameters) models in all of machine learning.

Therefore, **training them successfully from scratch requires incredible amounts of data, on the order of hundreds of gigabytes, terabytes, or even petabytes.**

In general, in order to compete with the state-of-the-art models in the field, you'll need data resources on this scale, or have access to highly proprietary data that can't be found anywhere else. In terms of training from scratch, gathering this much data is costly. It requires having an extremely large customer base which you can mine, countless hours of intelligent web scraping, or access to proprietary, paid datasets.

Acquiring such data on your own requires high expenditures of time and resources, and purchasing or licensing such datasets from a third-party company tends to be quite expensive. Many datasets can easily cost thousands of dollars per month to license, and you will likely need many such datasets to train your model to a sufficient level of accuracy. You may also be limited by the EULA on how you can use such datasets. For these reasons, training transformer models from scratch is something primarily pursued by technology leaders or academic researchers.

To make these large language models more accessible for business users, pre-trained versions have been developed that can be leveraged via a process of fine-tuning with relatively little extra data. "Relatively little extra data" means in the range of 5,000 to 10,000 training documents, a number which is still difficult to collect for most business use cases, not to mention the enormous internal subject matter expert resources it would require to review and annotate so many examples.

Most successful applications of such

Even if enough training data can be found to fine-tune the transformer model, its capacity to successfully solve a specific NLP task is not certain.

As a general principle, it's best to devote effort to those parts of the business which serve as unique differentiators. Unless you are a dedicated NLP company, this is unlikely to be your NLP technology.

models have dealt with extremely specific domains such as conversational AI (e.g., chatbots and question answering systems), summarization, machine translation, and text generation. The applicability of transformers and other language models to broader NLP tasks such as parsing, searching and classifying unstructured documents is an open research area that is still relatively unexplored.

Another thing to note is that the act of fine-tuning introduces a lot of the same constraints and expenses as discussed in the previous sections.

Fine-tuning the model requires understanding it at a deep technical level such that it can properly be trained. This reintroduces the need for an AI development team and doesn't shortcut the original problem, unless such a team is already in place. As a result, pre-training can only marginally reduce the amount of work involved in model development, and this for companies which already have expansive technology teams in place. However, for those who do not already have a development team, the cost of using pre-trained models is often greater than the value they bring.

Expertise

Finally, it's important to consider the time investment required to build a custom NLP solution. Beyond the material costs involved, it's a simple fact that industry leaders such as Google, Amazon and Microsoft, as well as NLP experts like Cortical.io have poured many years of focused effort into refining their systems to be state-of-the-art. **It would likely be impossible to replicate their results without investing a similar amount of time and focus,** and this is effort that could be devoted to the other areas of your business.

It is often better to tap into the expertise of established industry leaders for a fraction of the time and cost required to build a custom solution. These vendors also typically provide ways to customize their solutions to your unique use case.

LIMITS AND OPPORTUNITIES OF IDP SOLUTIONS

Using Pre-Built Solutions to Save on Costs

In response to the difficulties associated with building a custom classifier or search solution from scratch, many companies offer pre-built solutions for IDP which require no training and address a specific use case. **The main problem with these solutions is their inflexibility.** Because they have been trained for a single task, they cannot be readily adapted when new data is gathered, or a new type of document arises.

For supervised tasks such as document classification, adapting a pre-built solution for IDP requires having to annotate new data to cover additional document types and vocabulary variations. This leads to two problems: are there enough use case relevant training data available to reach good levels of accuracy? If yes, do your business users have enough time to annotate them? Unless you can answer both questions with yes, there's a good chance you'll find that the pre-built solution cannot help beyond the initial, foreseen use case.

Very often, a pre-built solution does not cover the unique, technical terms present in your own documents. This leads to an inability of the model to properly interpret the documents, leading to poor downstream business decisions based on incorrect understanding of the processed text. The result can be costly mistakes which dwarf the amount of money saved by purchasing a pre-built IDP solution.

Finding a Flexible and Efficient IDP Solution

The challenges discussed in this white paper suggest an optimal approach for businesses looking to acquire IDP capabilities. The ideal solution must have the following qualities:

1. **Must be easily customizable to a specific use case.**
2. **Must be able to be trained and re-trained in response to new data.**
3. **Must be able to handle vocabulary that is not contained in training data.**
4. **Must be capable of being trained without large datasets.**
5. **Must not require extensive in-house technical and AI expertise to train and utilize.**
6. **Must be fast and not require extensive computing resources to achieve scalability.**

Because all Cortical.io IDP products include a very capable embedding model which uses Semantic Folding, they are superior to other commercial products including the word and document vectors created by approaches such as word2vec, GloVe, and FastText.

One reason for the superiority of semantic fingerprinting is that it preserves word sense and eliminates noise. Moreover, semantic fingerprints are stable across different languages, allowing direct comparison of text across any combination of languages without the lossy step of machine translation. More details on exactly how the fingerprinting works can be found in the appendix.

Cortical.io provides a number of products that use the power of semantic fingerprints to search, extract, and compare key document information as well as filter, classify, and route messages such as emails and social media posts. These products are designed for subject matter experts (SMEs) and business users who can easily customize models to their specific use case with a few annotated documents – in the range of 100s. No data scientists nor AI experts are needed to tune the models. These products can also be easily integrated into existing software systems as well as CRM and CLM software and business intelligence tools.

Cortical.io has developed an IDP method which satisfies all of these criteria. In contrast to many other machine learning approaches, Cortical.io uses an unsupervised fingerprinting method to learn representations of words and documents.

The Contract Intelligence tool has been successfully applied to a number of industry use cases. As one example, a major audit and management consulting company needed to assist its many enterprise customers in disclosing the leases on their balance sheets due to changes in US and international regulations. This was a difficult task to automate because lease agreements are non-standardized; the same type of information can be expressed using many different terms, leading keyword-based systems to provide subpar results.

The company was able to quickly integrate Cortical.io's Contract Intelligence tool into their existing contract-processing workflow and use SMEs to train the system in a matter of days. The system was then evaluated and found to successfully extract and classify relevant information from the leasing agreements, much faster and more accurately than would be possible using manual labor. It also gave the SMEs full control over the fine-tuning and tweaking of the system, allowing them to adjust its training as new leasing agreements were gathered.

Cortical.io's other IDP product is Message Intelligence. This tool is designed to filter, classify, and route messages in real time, making it ideal for processing documents as they are received. It can be integrated with any REST API, allowing it to easily be dropped into any existing web backend.

The Message Intelligence product has been used successfully by many major corporations to streamline their processing of real time data. As one example, Cortical.io partnered with a global biopharmaceutical company to develop a prototype based on Message Intelligence which identified on-and-off-label medication usage in a

The Cortical.io application achieved a very high level of accuracy on the task based on standard NLU metrics (92-100%) in classifying on-label vs. off-label usage.

static set of over 2.2 million Reddit posts discussing medications.

The application automatically and accurately filtered for trade and generic names of medications. From there, it filtered out off-topic and ambiguous posts, and classified the remaining ones based on condition type and on vs. off label usage. The results were then summarized in a visual dashboard. In this way, the company was able to understand better the uses and side effects of its medications within a large customer population.

All of the Cortical.io products, including Contract Intelligence and Message Intelligence, are blazing fast as representations are pre-computed and thus don't affect the overall query response time. Because the representations are sparse and binary, they can be compared via simple Boolean operations which are implemented at the hardware level of all machines. This makes processing incredibly efficient and means that huge volumes of documents can be processed extremely quickly.

In fact, it has been shown that **Cortical.io products can help teams improve responsiveness and meet critical deadlines by reducing processing times by up to 80%.**

CONCLUSION

IDP is a difficult and varied task that operates on all manner of unstructured, structured, and semi-structured text. While pre-built solutions do a great job when documents are organized and simple, they normally provide inferior results on documents which are highly unstructured and domain specific.

The downsides of applying pre-built solutions to complex, unstructured documents which vary in form and content have led many organizations to tread the path of building custom, in-house solutions. However, as this white paper has shown, such an approach brings with it many drawbacks of its own, many of which are insurmountable for all but the largest enterprises.

Luckily, Cortical.io provides a suite of products which apply IDP successfully to even the most complex and unstructured documents, at speed, at scale, and with an unmatched level of accuracy.

APPENDIX

THE BEAUTY OF SEMANTIC FOLDING - HOW IT WORKS

Semantic folding is rooted in neuroscience and the Hierarchical Temporal Memory model originally developed by Jeff Hawkins.

This model describes how the cortical modules organize in the brain when exposed to a sequential stream of sensory input data and represent each sequence as a unique pattern that is stored within memory. Semantic fingerprinting bases its data representation on this concept by building sparse, distributed, binary representations of text where each bit in the representation corresponds to the presence in the text of a conceptual idea.

As an example, the semantic fingerprint of the word “cat” would look something like this

The presence of 1s in the representation captures the various properties that a cat possesses, and these are organized in a hierarchical manner from the most basic such as “has 4 legs” to the most specific such as “eats rodents”. Already, one of

the primary advantages of the semantic folding model can be seen, which is that it provides a conceptual framework that captures key high and low level attributes of a word or document. By computing the attributes that are shared between representations, similarities can be easily estimated.

Contrast this with other representations such as word2vec, GloVe, and FastText which are real-valued (not binary), dense (not sparse), and are based on word statistics, not high-level conceptual ideas. What's more, semantic fingerprints are fault-tolerant. If some small subset of the bits are lost or corrupted, the overall meaning of the document is still captured. Redundancy is natively built into the representation.

The abstraction of the semantic fingerprint also helps it to deal with vocabulary coverage. As stated in the first section, most conventional representation methods cannot understand the meaning of a word unless it is present in the training dataset.

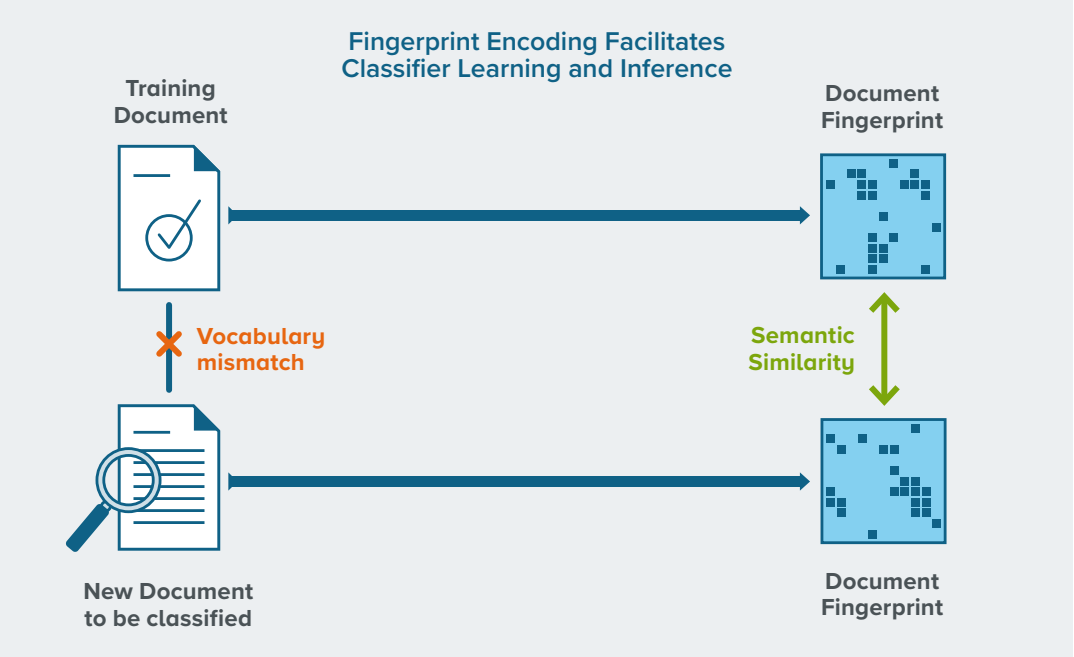
However, classifiers using semantic fingerprints can infer the meaning of terms they have not seen during training due to the semantic fingerprint similarity.

Instead of a vector, we can think of the semantic fingerprint of a document as an n x n matrix where the rows are fingerprints corresponding to the document's words or sentences. This is powerful, because each atomic unit of the document is contained in its overall

Because the representations are binary, this computation is lightning fast, and because they are sparse they can be stored with minimal memory overhead by only storing the indices of the few set bits.

representation. When creating a document representation from word vectors, the vectors need to be averaged. This causes an extreme loss of information such that the resultant document representation is often useless. There are ways around this; for example, modern models such as BERT and XLNet can more intelligently create document vectors. However, they are limited in terms of the sequence length they can consider, which results in some information loss. They are also computationally expensive and have a large footprint which can limit their applicability when trying to process many documents at once.

Semantic fingerprints do not suffer from these limitations. The union of word and sentence representations into a documentation does not result in any information loss. They also retain their computational efficiency at scale, which is key to Cortical.io's ability to process hundreds or thousands of documents in a timely manner.

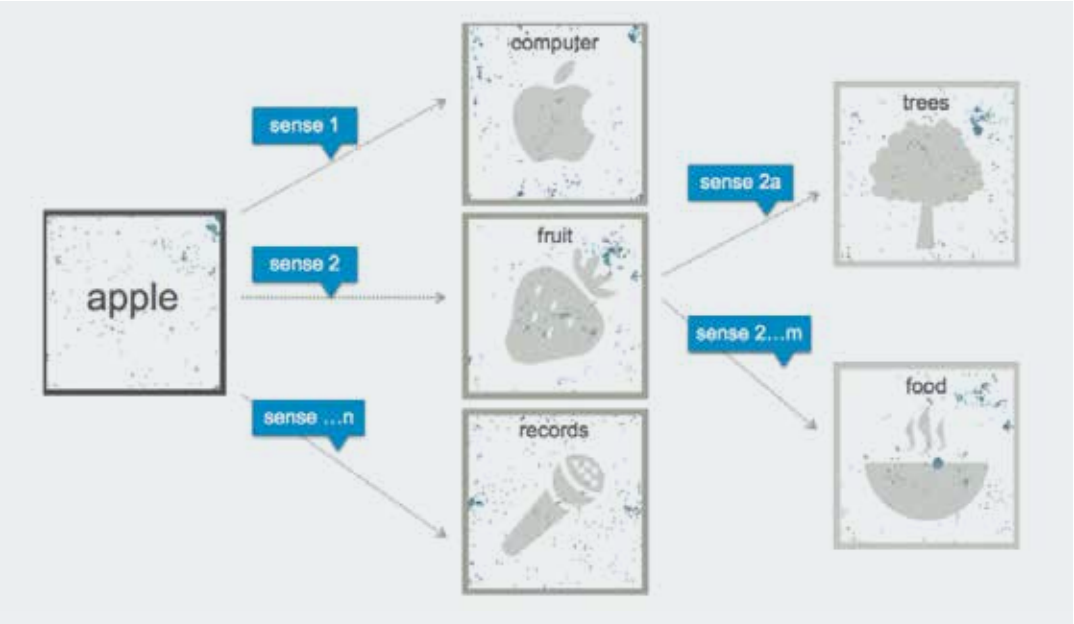


Fingerprints also can be easily extended to incorporate the presence or absence of specific keywords in a document, something that's not possible with conventional vector representations. This can be incredibly important for processing documents with complex, technical terminology that may not be represented in many training datasets.

Finally, another major advantage that semantic fingerprints have over conventional representations is that they can automatically disambiguate word senses. This is because the representations are based on concepts, not word statistics.

For example, the word “apple” would have three distinct semantic fingerprints, one corresponding to each of its senses–the computer company, the fruit, and the record company. The concepts represented in the fingerprint for a computer company, for example, “manufactures electronics” and “develops operating systems” would be totally distinct from those in the representation for a fruit such as “grows on trees” and “tastes sweet”.

In contrast, conventional word vectors condense all the senses of a word into a single representation, thus convolving their meanings. Thus, the conventional vector for “apple” would have some attributes that correspond to fruits and some that correspond to computer companies. As a result, this representation would present some inaccurate information for any given context that it was used in.



Disambiguating the senses of the word “apple” with semantic fingerprints