Proposal: Using Azure Search, Azure Machine Learning and Azure Storage for Cognitive Search

# Authors

Tyler Ayers, Cloud Solution Architect, Microsoft Deutschland GmbH

# Executive Summary

This proposal shows how Azure Search, Azure Machine Learning and Azure Storage can be leveraged to provide a self-learning, cognitive category-based search across any number of large and complex documents and information repositories. The cloud is ideally suited to this scenario for the following reasons:

* Automatic scaling of solution based on amount of data, no manual scaling of hardware necessary
* Utilize cloud connectors between storage repositories like OneDrive, Azure Storage and Web Pages / Bots for searching, no integration necessary
* Latest features and modern algorithms/capabilities at a very low price and operation effort because they are provided at scale through the Azure platform

# Current Situation

Currently product and user documentation is stored in hundreds of Microsoft Word files, many of which are many hundreds of pages long. This makes it difficult for the user to read and find answers to concrete questions regarding software programs and processes used by the banks, and means that many more support calls must be handled to help users with common questions and problems.

# Proposed Solution

The proposed solution is divided into two parts: first the document processing, categorization and indexing, and then the searching.

## Part 1: Document Processing, Categorization & Indexing

The solution flow starts with a user who creates or updates a large user document. The easiest way is for the user to just save the document to OneDrive or a different cloud storage platform. A trigger in **Azure Functions** can then react on the upload, and start the document processing.

The document processing **Function** includes splitting and saving the document into individual chapters and sub-chapters that can be displayed in a chat or other search page. The document parts can be converted into a format that can be displayed in a chat, such as Html and Markdown.

Each document part is also sent to **Azure Machine Learning** or another categorization service that reads the text, and returns the categories and key words for the document part. This information is saved as metadata with the document part back to **Azure Storage** for indexing.



*Figure 1: Document processing flow to split, convert, categorize and index the document parts.*

### Document Processing

The document processing in the PoC was adapted to fit the document structure of the agree21 SEPA handbook, and so uses Heading1, Heading2, Heading3, and bold titles as orientation to split the document. To cover other formats of documents, customization features could easily be introduced to either let the user decide how the document should be split, or recognize the document based on previously-defined categories to best format the resulting parts.

Additionally, images in tables are not yet realized in the PoC version, along with nested lists. These features would not be much effort to realize in further versions.

### Machine Learning Categorization

The categorization and key word generation in Azure Machine Learning can be taken in levels. The first level of basic text key word analysis was tested for the Azure PoC. Machine Learning generally relies on existing data sets to use for classification purposes – for the PoC this data was not available, so the simplest classification was used.

**PoC Level**

In the PoC level basic text analysis capabilities in Azure ML were used to get a set of key words from the text, and use those key words for indexing of the document part. This is simple and effective, but does not improve automatically with time.

**Next Levels**

These further levels of machine learning could be used to further improve the categorization and key word processing, especially together with datasets of existing categorization data.

* N-Gram Feature Extraction: This functionality in Azure ML is used to generate feature vectors of the most commonly used words in the texts. These feature vectors can then be used to find the best key words that similar texts used. For this feature a dataset of existing classifications is needed.
* Feature Hashing: feature hashing, similar to N-Gram extraction, can be used to generate feature vectors to match key words with an existing dataset.
* Add custom Python or R scripts to recognize Fiducia-specific vocabulary and add additional intelligence to the key word selection process.

More information on Text Analysis in Azure Machine Learning can be found [here](https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-text-analytics-module-tutorial).

## Part 2: Searching with Azure Search

Searching for information through a bot or web page is done by first contacting **Azure Machine Learning** by web service to get the key words and category of the request. The key words can either be extracted through simple text analysis algorithms (as in the PoC level), and/or using trained models and an existing dataset of user request data to get better results.



*Figure 2: User search flow*

After retrieving the key words, the bot service contacts Azure Search by web service to do the search. In the PoC there was only one document that was indexed into around 1200 parts. There were no overlapping parts from other documents. That’s why in the next levels some additional features could be added.

**PoC Level**

Document parts were indexed only based on key words, with no filtering or scoring of the documents.

**Next Levels**

* Azure Search Filtering: add filtering of the search documents based on a product or topic (for example agree21, SEPA, etc..) This would allow the user to filter results when key words overlap between topics (same key words in similar documents). In this case the user could filter for the product he/she is interested in, and get the correct results. Filtering is a built-in feature of Azure Search, so little effort is needed to use it.
* Azure Search Scoring: add scoring of searches using other metadata on the document such as user feedback review (for example the data when the bot asks the user if info was useful), or a low-score of the entire text content. Scoring is a built-in feature of Azure Search, so little effort is needed.

# Conclusion

The **PoC Level** descriptions of the proposed solution above show that document processing, text categorization using machine learning, searching and integrating in a bot or web interface is possible with very little effort, mainly thanks to the built-in integration between the Azure platform features of Storage, Machine Learning, Web Apps, Search and Bots. The PoC level shows a basic but effective classification and search solution that would prove useful from day 1.

In addition, the **Next Level** points above demonstrate where further steps could easily be taken to further improve the effectiveness of the categorization and search features. This mainly would come through integrating datasets of user and support data, deploying machine learning algorithms on those datasets to train classification models, and then using those models to classify new documents or process search requests.