Digital Assistance for System Requirement Discovery and Analysis using Machine Learning Natural Language Processing Algorithm

Nipa Phojanamongkolkij, Braxton VanGundy, Ian Levitt, and Heidi Glaudel

NASA Langley Research Center

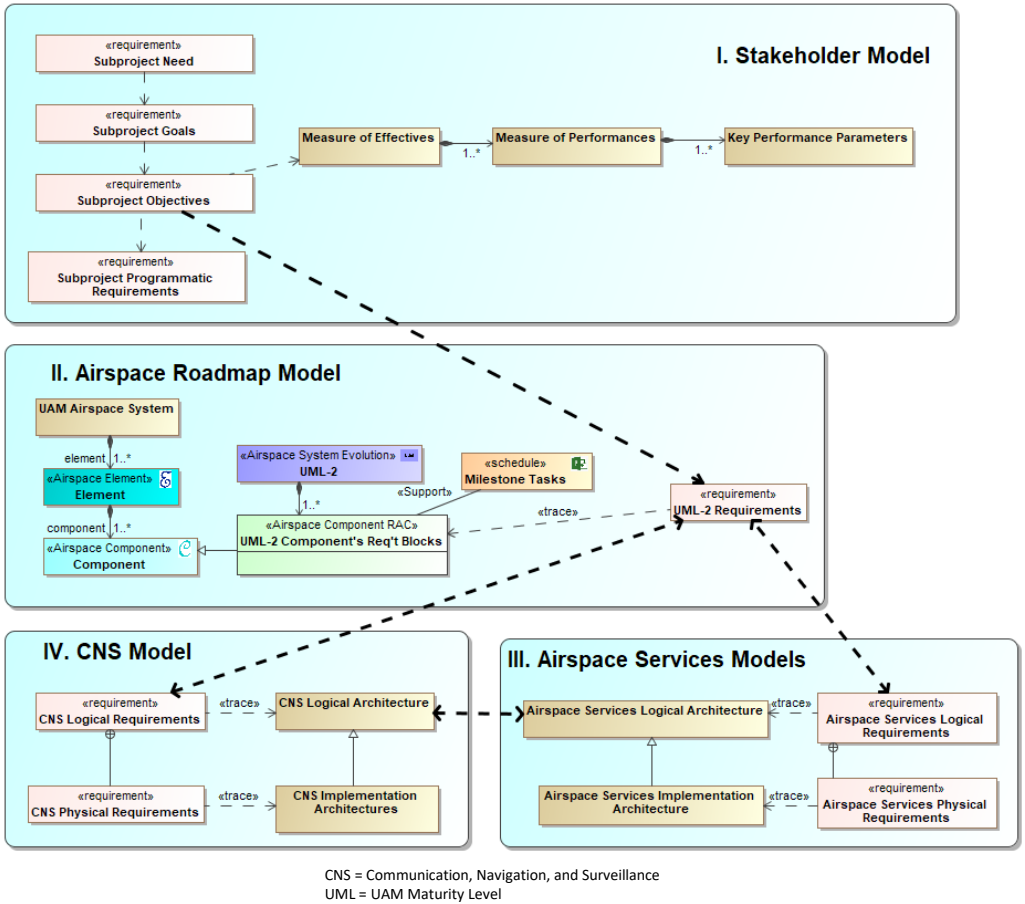
Outline

- Introduction
- Machine Learning Natural Language Processing Application
  - The Application – Look and Feel
  - What happen behind the scenes?
- Benefits
- Lessons Learned and Next Steps

Introduction

- NASA's Air Traffic Management-Exploration (ATM-X) Urban Air Mobility (UAM) Subproject is conducting research that **evolves UAM airspace** towards a highly automated and operationally flexible system of the future.
  - See https://www.nasa.gov/uam-overview/ for more information
- The complexity of UAM airspace evolution requires a **plan** to effectively organize, integrate, and communicate NASA's research and development.
  - This planning tool is called **the UAM airspace research roadmap**.
  - Implemented through Model-Based Systems Engineering (**MBSE**) methodology
  - Leveraging machine learning natural language processing (ML NLP, or just **NLP**).

Systems Models for Roadmap Development



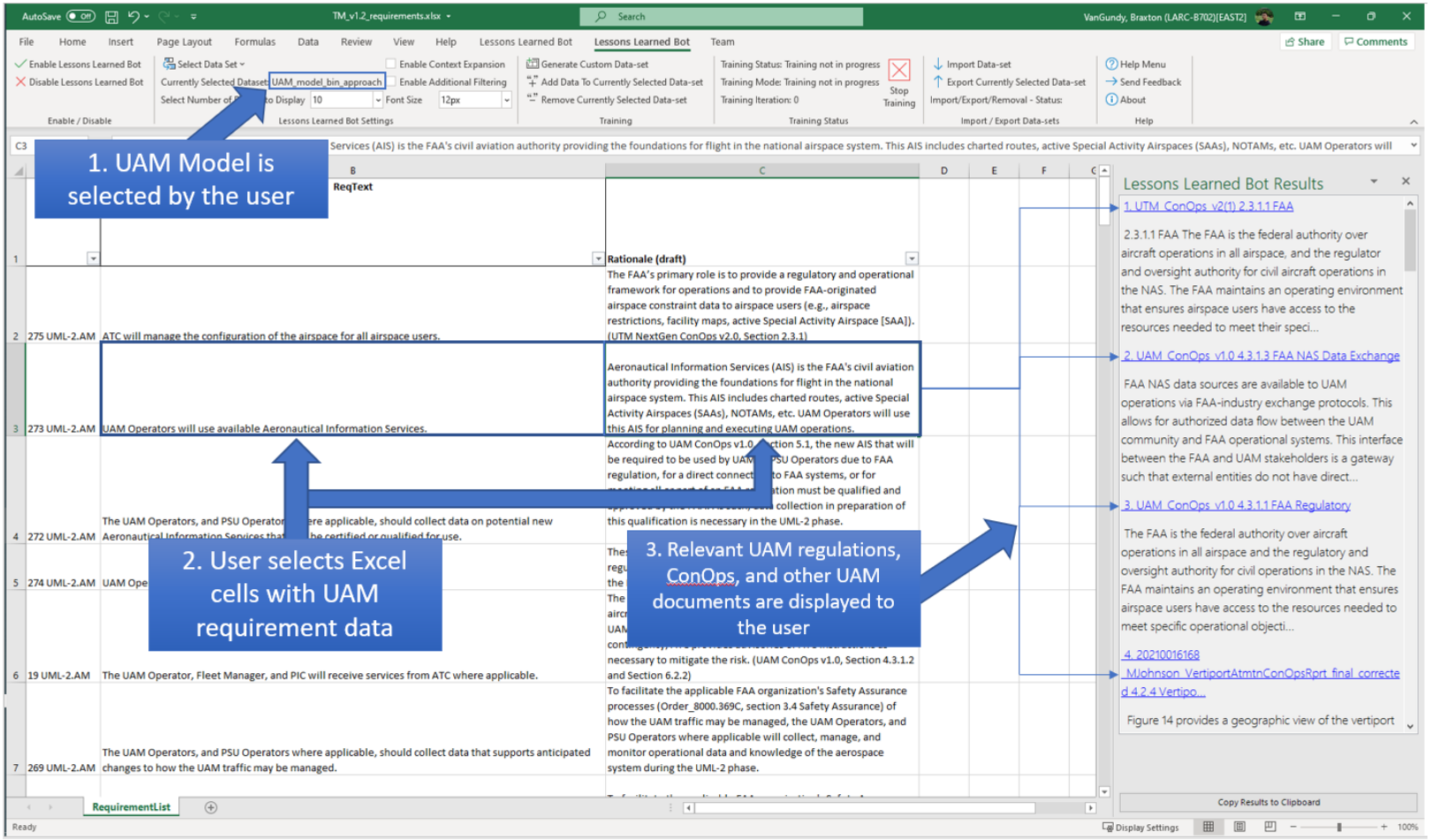
CNS = Communication, Navigation, and Surveillance
UML = UAM Maturity Level

- Meta-model definition showing structure and high-level dependencies within and between models.
- Dependencies between models are through requirement traceability.
- **High cognitive demand** to perform Roadmap's requirement discovery and analysis to identify its rationale, reference sources, and dependencies among requirements.
- This challenge can be overcome by the application of **ML NLP methodology.**

Machine Learning Natural Language Processing Application

- NASA has been transforming our R&D, system engineering, and project management processes to fully leverage evolving digital technologies
  - to view digitized records virtually across diverse organizations, thus improving design efficiency, collaboration, and safety
  - to streamline our processes to take advantage of new digital tools to optimize process flows.
- Perfect opportunity to infuse a new way in the requirement discovery and analysis.
- Developed an application which is an intelligent data interface that rigorously searches a corpus of UAM technical manuscripts, FAA regulations, etc. in near real-time.
  - 20 pdf documents, each ranging from 20 to a few hundred pages long.

The Application – Look and Feel



What Happen Behind-the-Scenes?

- PyInstaller package
  - No need for the user to install a Python environment in order to run our application.
  - Allows for the seamless integration of Python Machine Learning capabilities directly into Microsoft Excel.
- Textract, PdfMiner, Xlrd packages
  - Supports the extraction of the following data types for training:
    - .docx | .pptx | .txt
    - .pdf
    - .xlsx and .xls
- PyDoE package
  - Python experimental design package.
  - Used to generate a matrix for the Application (called **Lessons Learned Bot or LLB**) auto-train functionality.

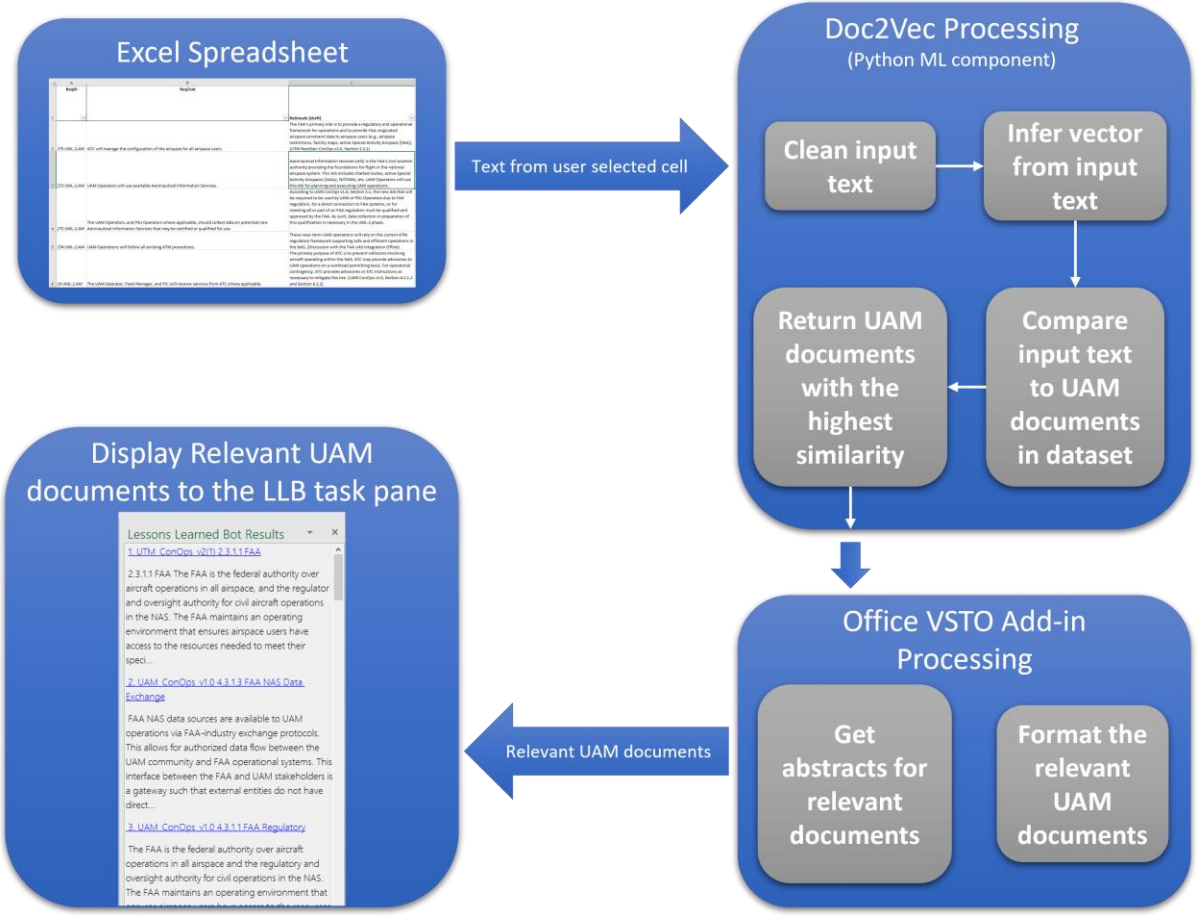
What Happen Behind-the-Scenes?

- Doc2Vec package from GenSim
  - A NLP Machine Learning approach for finding related documents using their corresponding vector representations.
  - Based on a single-layer neural network model
    - Input -> Hidden Layer -> Output with a Softmax Classifier.
  - A total of 19 hyper-parameters for training. Examples are:
    - Vector Size - Size of Neural Network Input / Output Nodes, or the dimensionality of the feature vectors
    - Window - Sliding window size of words in a sentence to be considered as a context
    - Min. Count - Minimum threshold frequency (count) of words to be discarded during the pre-processing of the training dataset. (Less frequent words)
    - Sample - Threshold value to compute the probability of words with high frequency (with less added information) that will be discarded. (Too frequent words)

What Happen Behind-the-Scenes?

- Auto Training vs. Set Default Parameters (or Out-of-the-Box)
  - Two user training modes built into the LLB.
  - Auto Train -> Model generator runs through various hyper-parameter setting scenarios, evaluates the generated model, and selects the best one. Produces best model but is computationally intensive.
  - Out-of-the-Box -> A set of static default parameters is used to train the models. These parameters were verified in the literature to work adequately over a wide range of models but may not perform as good as Auto Train does. Only requires 1 training run vs. multiple runs with Auto Train, therefore it is much faster.

Document Recommendation Process



*VSTO = Visual Studio Tools for Office

Benefits

Today's practice

- Systems engineers (SE) performing reviews of this corpus.
- SEs identify requirement's rationale and references.

New practice

- The team, consisting of SEs and the NLP expert, uses this corpus to train the Doc2Vec model.
- After validation of the newly trained model, the team can package this add-in application and deploy it to users.
- SEs and/or general users can then install and use the application to search through the UAM body of knowledge efficiently for rationale and references

Benefits:

- Proven a significant time saving (~12x faster for one pdf) over today's practice.
- Reusable trained model of the UAM body of knowledge beyond the current study to survey literature relevant to UAM research questions (as opposed to just requirements).
- Extendable to incorporate the Advanced Air Mobility (AAM), NAS, and National Transportation System (NTS) knowledge corpus.

Lessons Learned and Next Steps

- **Lessons Learned**:
  - Pre-processing pdf before training
    - One record per page: does not produce a good result because:
      - a page could have different research concepts especially at the end of one section and the beginning of the next
      - a page could show an incomplete concept when it takes several pages to articulate the concept.
    - One record per section: produces a better result and is used in our application.
    - Manual process in dataset preparation because each pdf comes in various section layouts. Scripting will need to be customized for each pdf.
  - Grouping the corpus into different themes and building a model for each theme.
- **Next Steps**:
  - More documents to be included in this corpus.
  - A new use case for this NLP tool to find “similar” requirements.
    - A trained model for a corpus of “requirements” for identifying requirement’s dependencies.