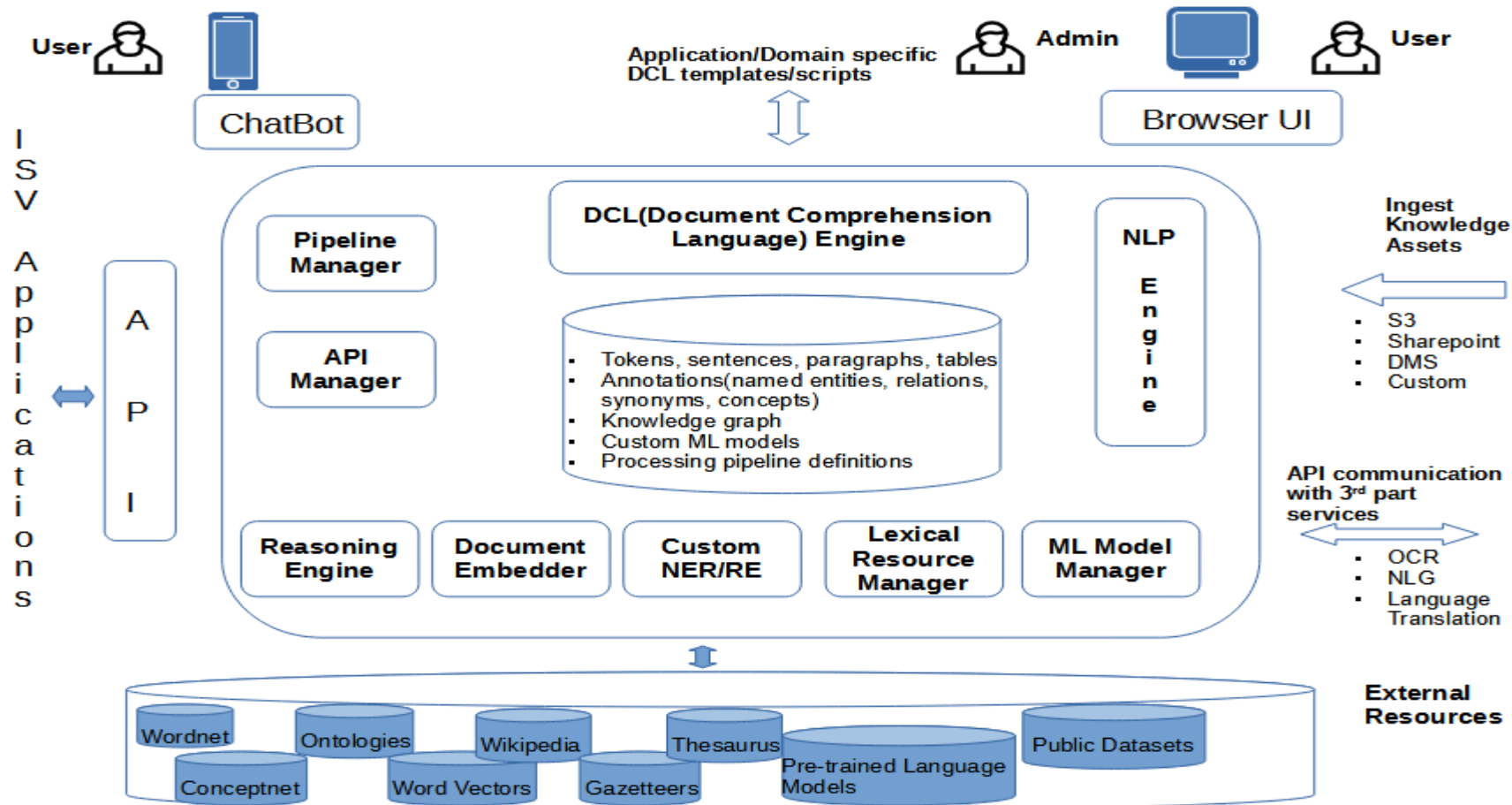# The technology behind LearnITy™ Knowledge Engine

Empowering organisations to perform *Deep Document Understanding* on their knowledge assets

LKE Application Architecture

User

ChatBot

ISV Applications

A P I

Pipeline Manager

API Manager

Application/Domain specific DCL templates/scripts

Admin

User

Browser UI

DCL(Document Comprehension Language) Engine

- Tokens, sentences, paragraphs, tables
- Annotations(named entities, relations, synonyms, concepts)
- Knowledge graph
- Custom ML models
- Processing pipeline definitions

NLP Engine

Ingest Knowledge Assets

- S3
- Sharepoint
- DMS
- Custom

Reasoning Engine

Document Embedder

Custom NER/RE

Lexical Resource Manager

ML Model Manager

API communication with 3rd part services

- OCR
- NLG
- Language Translation

External Resources

Wordnet
Ontologies
Wikipedia
Thesaurus
Public Datasets
Conceptnet
Word Vectors
Gazetteers
Pre-trained Language Models

NLP Engine

- A custom NLP engine provides the strong foundation of LKE
- Supports ingestion of born-digital documents of multiple formats – PDF, Word, Powerpoint, Excel, Text
- Extracts and indexes all components of documents – page, paragraph, sentence, phrase, word, header, footer, table, image
- Performs tokenisation, POS tagging, parsing (constituency and dependency), named entity recognition and linking, relation extraction
- Strong support provided for custom NER and custom RE using gazetteers, rules, ML (CRF), pre-trained DL models
- Indexing using Tf/Idf and BM25
- Supports multiple languages – English, French, German (other language support WIP)

Document Comprehension Language

- DCL (patent pending) is a DSL (domain specific language) purpose built for the domain of deep document understanding
- The various document understanding (DU) requirements of an organisation (extraction, validation, comparison, etc.) are expressed using DCL
- It is a declarative DSL expressed using XML notation and contains different tags for implementing the DU operations such as information extraction, document comparison, etc.
- DCL makes use of the various annotations generated by the NLP engine while processing the documents
- Work in progress to use Controlled English as the language for DCL (as a substitute for XML based syntax) so that business users may easily express their DU requirements without requiring much help from IT

Vectorisation

- Supports multiple pre-trained word-embeddings (Word2Vec, Glove, FastText, and many others) for multiple languages
- Multiple word vector aggregation mechanisms supported – average/max/..
- Multiple OOV policy supported – ignore/embedding average/..
- Supports vectorisation using models from model hubs (BERT, Sentence Transformer, etc.) as well as via APIs (Google Vertex, OpenAI daVinci, etc.)
- Facility to train your own embedding models
- Support for multiple distance functions – cosine/euclidean/dot product/…
- Efficient implementation of Nearest Neighbours and Locally Sensitive Hashing

Lexical Resources and Reasoning

- Wordnets/thesauri/dictionaries/.. for multiple languages supported
- Support for ingesting Wordnets in multiple formats – PWN/LMFXML/DebVis
- Wikidata used for NE linking, training data augmentation
- Utilises *ConceptNet* for common sense reasoning
- Support for ingesting Ontologies in various formats – OWL/XML, SADL, Turtle
- Domain ontologies are used for information extraction (OBIE), search query expansion, performing logical reasoning (Description Logic)

Processing Pipelines

- Both interactive and batch processing of documents are supported
- Processing pipelines are defined declaratively and represented in XML notation. Pipeline definitions are stored in DB and may be reused across multiple document repositories
- Pipelines may include all tasks: ingestion, NLP processing, DU operations, report generation, sending results to downstream applications via API
- Documents may be ingested from S3, DMS, Sharepoint, and custom sources
- Pipelines may run with multiple types of executors on single m/c or clusters
- Pipelines may be run from the GUI or from the CLI
- Job status may be seen from GUI

Technology Stack

- Built on reputed Open Source software components powering large enterprise deployments
- Based on the Java platform making LKE available on almost all OS
- Uses MySQL as the database software enabling handling of very large datasets
- Supports multiple distributed computing platforms for handling large application loads
- The above technical choices make it simple to scale the solution horizontally via *clustering* and *sharding*

Packaging and Deployment

- Web-based application with role based GUI for deployments on customer owned infrastructure (on premise or private cloud)
- *LKE Saas* – Subscription based software as a service offering in the form of a collection of APIs made available on public cloud infrastructure
- Licenced software components packaging partial LKE capabilities to be used as engines by 3rd party product developers (ISVs)
- Installation options
  - OS native
  - Docker containers

Security Certified

Thank You