2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW) | 978-1-6654-4456-9/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICSTW52544.2021.00047

AutoKG - An Automotive Domain Knowledge Graph for Software Testing: A position paper

Vaibhav Kesri
ARiSE Labs at Bosch
Bangalore, India
Vaibhav.Kesari@in.bosch.com

Anmol Nayak
ARiSE Labs at Bosch
Bangalore, India
Anmol.Nayak@in.bosch.com

Karthikeyan Ponnalagu
ARiSE Labs at Bosch
Bangalore, India
Karthikeyan.Ponnalagu@in.bosch.com

***Abstract*—Industries have a significant amount of data in semi-structured and unstructured formats which are typically captured in text documents, spreadsheets, images, etc. This is especially the case with the software description documents used by domain experts in the automotive domain to perform tasks at various phases of the Software Development Life Cycle (SDLC). In this paper, we propose an end-to-end pipeline to extract an Automotive Knowledge Graph (AutoKG) from textual data using Natural Language Processing (NLP) techniques with the application of automatic test case generation. The proposed pipeline primarily consists of the following components: 1) AutoOntology, an ontology that has been derived by analyzing several industry scale automotive domain software systems, 2) AutoRE, a Relation Extraction (RE) model to extract triplets from various sentence types typically found in the automotive domain, and 3) AutoVec, a neural embedding based algorithm for triplet matching and context-based search. We demonstrate the pipeline with an application of automatic test case generation from requirements using AutoKG.**

***Index Terms*—Automotive Domain Knowledge Graph, Software Testing, Natural Language Processing**

I. Introduction

Software testing in the automotive domain is challenging due to the complexity of the systems which typically contain large amounts of dependencies between the various modules. A tester requires domain understanding to extract the relevant information for a requirement from the software documents to write correct and complete test cases, which is a time-consuming and costly process. Knowledge Graph (KG) can significantly assist and aid the tester in the following ways:

1) Handling complex dependencies: Storing information in a structured format enables the tester to leverage the inter-dependencies between states, signals, etc.
2) Exhaustive testing: As there can be multiple ways to test a particular requirement, exhaustive test cases can be generated using all the relevant sub-graphs of the KG.
3) Domain Visualization: In complex domains such as automotive it is a challenging task even for a tester to read the documents and extract the relevant information. A KG can help the tester by providing information at any granularity level.

While large scale KG such as ConceptNet [1] and NELL [2] have previously been generated from open source datasets like Wikipedia, WordNet etc., building a KG is still nascent in the automotive domain due to the following challenges:

- The content is largely technical and describes complex domain features such as Anti-lock Braking Systems.
- Features build upon a strict hierarchy of system components.
- Software documents have varied scientific entity classes, terms, symbols, and denotations.

There have been a few efforts in leveraging automotive domain specific KG for autonomous driving applications like scene understanding [3] and sensor fusion [4]. Another approach leveraged an ontology based text mining system for knowledge discovery from the diagnosis data in the automotive domain [5]. However, none of these have been built for the application of software testing in the automotive domain. An NLP based pipeline for generating test cases of automotive domain software using a Knowledge Graph has been previously proposed in [6], from which we have leveraged few components and further extended it by adapting a domain-specific ontology, along with a proposal to revamp the pipeline with recent state-of-the-art approaches in NLP.

In this paper, we propose a pipeline to tackle the sparse and complex nature of automotive domain text and generate a KG primarily for automatically generating test cases of functional and unit level software requirements. Our pipeline has been designed after analyzing several complex automotive software systems at Bosch. In the following sections, we explain the components of the pipeline and demonstrate how it can be used for the generation of test cases.

II. AutoKG pipeline

The AutoKG pipeline as depicted in Fig. 1, applies a series of NLP algorithms to mine information from the text corpus. In the following sub-sections, we describe these components in detail, the challenges tackled within each component, and the recommended methodology for processing automotive domain text.

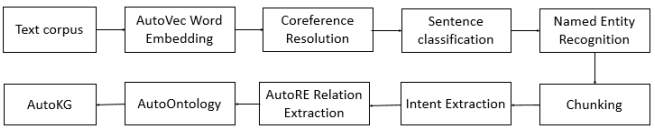

Fig. 1. AutoKG creation pipeline.

978-1-6654-4456-9/21/$31.00 ©2021 IEEE
DOI 10.1109/ICSTW52544.2021.00047

A. AutoVec: Learning word embeddings from sparse automotive domain text

Word embeddings are crucial for semantically understanding the text and they have been used on a large scale to boost the performance of many NLP tasks such as Named Entity Recognition (NER), Syntactic Parsing, etc. Word embedding models can be classified into two categories:

- Static Embeddings: These embedding models (e.g. Word2Vec [7], GloVe [8]) hold a single representation of a word regardless of how many meanings it can have. This can potentially lead to ambiguity and poor accuracy in the case of polysemic words.
- Dynamic Embeddings: These embedding models (e.g. BERT [9], ELMo [10]) will provide a different representation for a word depending on the context it is found in. They are able to handle polysemic words and provide contextualized dynamic embeddings. Dynamic Embedding models typically require domain-specific fine-tuning to be performed at two levels:
  1) Language Model fine-tuning: This is done to understand the semantics of the domain. In the case of the automotive domain, the language model fine-tuning can be done with the various software specification documents available in AUTOSAR (http://www.autosar.org/).
  2) Task-specific fine-tuning: This is done to understand task-specific nuances of the domain to be used for downstream tasks such as NER, RE, Coreference Resolution, etc.

As the textual corpus found in the automotive domain is typically sparse, classical word embedding techniques such as Word2Vec and GloVe perform poorly. Further, these embedding models fail to handle out-of-vocabulary (OOV) words and polysemic words, which are commonly found in automotive domain documents. Dynamic embedding models such as BERT provide contextualized representations for a word and can even handle OOV words as it uses the WordPiece tokenization algorithm [11] to break down even previously unseen words. To generate meaningful domain-specific embeddings, it is also essential we incorporate external domain knowledge such as domain vocabulary, semantic relations, etc. We propose AutoVec, an embedding that is learned by jointly fine-tuning and training both the pre-trained dynamic embeddings along with hand crafted automotive domain-specific features. In the case where the word is OOV, the BERT embedding will be used directly. Based on the approach mentioned in [12], the domain-specific feature consists of:

- Domain vocabulary: Checks whether the word is an entity of the domain.
- Semantic category: The named entity class of the word.
- Direct relation: Encodes direct relations between the word (if it is a domain entity) and other entities of the domain.
- Indirect relation: Encodes common hierarchical relations between the word (if it is a domain entity) and other entities.

There are two methods to learn the AutoVec embeddings:

1) Joint fine-tuning of BERT + Training domain feature: In this setting, the domain feature will be concatenated to the final encoder layer output of BERT, after which joint fine-tuning and training is performed.
2) Independent fine-tuning of BERT + Training domain feature: In this setting, the BERT encoder is fine-tuned separately, after which the fine-tuned BERT embeddings are concatenated to the domain feature to train a separate model for a downstream task.

B. Coreference Resolution

Entities and events are typically mentioned in several ways across technical documents, which have to be resolved accurately while building a KG. This is achieved with a Coreference Resolution model. For BERT based models, coreference resolution is performed up to a length of 512 tokens of the input text. BERT coreference resolution architecture consists of a Transformer Encoder with a Coarse-to-fine head network [13]. The input is tokenized with a BERT variant of the WordPiece algorithm and passed into the encoder to generate contextualized representations for each token. The encoder representations are consumed by the coarse-to-fine network and iteratively refined using an attention mechanism.

C. Sentence classification

Prior to performing RE on the sentences, they are classified into predefined categories for better phrase and intent extraction. AutoKG is designed to handle the following types of sentences found in technical documents of the automotive domain and the classification model relies on the approach described in [14]:

- Non-nested Conditional: These sentences have evident pre-conditions and post-conditions without any sequence of sequence of cascading conditional sentences. These sentences typically are framed like: *If-then, When-then* etc.
- Nested Conditional: These sentences have multiple cascading nested conditional sentences, each building upon the pre-conditions and post-conditions from the previous sentences. These sentences typically follow the patterns like: *If-then-Elseif-then-Else then, If-then-Else-then*.
- Logical Operator influenced Conditional: These sentences can either be Non-nested or Nested Conditional, where the pre-conditions and post-conditions are influenced by logical operators like AND, OR, NOT. These sentences are typically framed like: *If...then none/any/all of the below conditions are satisfied*.
- Informative: These sentences do not have explicit pre-conditions and post-conditions, however, they contain important attribute information about the system. For example, *The system has 3 states: START, DRIVEAWAY, ACCELERATION*.

D. Name Entity Recognition and Chunking

Named Entity Recognition is a sub-task of information extraction which seeks to identify and classify named entities mentioned in an unstructured text into pre-defined classes. NER is a building block for Knowledge Graph creation and typically appear as nodes in the KG. In the automotive domain, NER models are trained by consuming is generated by consuming different sources of information like annotated text, A2L files, etc. The individual tokens of the identified named entities are chunked for better downstream processing. We recommend using AutoVec embeddings as input to the Conditional Random Fields (CRF) based NER model proposed in [6].

E. Intent Extraction and AutoRE: Relation Extraction for automotive domain text

Intent Extraction is a key component in understanding the semantic meaning of a sentence in its surrounding context. Specifically, in the scenario of software testing, intent extraction helps in identifying pre-conditions, actions, and post-conditions, which is commonly referred to as test intent. In our pipeline, we utilize the Constituency Parse Tree (CPT) based intent extraction algorithm proposed in [6]. These conditions are further converted into triplets and populated in the KG.

Further, Relation Extraction aims to identify relationships between entities of a domain. This is challenging in the automotive domain due to the following reasons:

- Triplets have to be generated from a variety of complex sentence types.
- Triplets need to be classified into pre-condition, action, and post-condition categories relevant for test case generation.
- Triplets from noisy or irrelevant text need to be discarded.

We propose AutoRE, which builds upon the triplets generated from existing RE algorithms like Stanford OpenIE [15] and ClausIE [16]. It performs the following steps over existing RE algorithms:

1) Filtering noisy triplets using Part-of-Speech (POS): Since triplets often contain noisy text which does not convey any useful information, POS tag of the Subject, Predicate and Object tokens are used to decide which tokens are removed and also determining if some tokens from the Subject and Object should be moved into the Predicate. This is a 3 step process: First, tokens from the Subject are stripped until a token appears which is either a Noun, Adjective, or a Cardinal number. Second, trailing tokens from the Subject and leading tokens of the Object are merged to the Predicate until a token appears which is either a Noun, Adjective, or a Cardinal number. Lastly, the remaining tokens in the Object form the triplet's Object.
2) Tagging triplets using NER: As we are dealing with the automotive domain, only those triplets that give information about a named entity are relevant. After the above steps are applied on the triplet, we discard all those triplets where neither the Subject nor the Object contains a named entity.
3) Hierarchical grouping using AutoOntology: Unlike off-the-shelf algorithms which treat each sentence independently without considering context, AutoRE identifies which state, function, etc. the triplet originated from and uses this information to group triplets based on the hierarchy. Such a hierarchical based grouping is leveraged during the information retrieval phase.
4) Generation of missed triplets using Dependency Tree: As software description documents typically contain large amounts of complex sentences, it can often lead to domain specific triplets being missed. To tackle this, we propose writing syntactic patterns on the Dependency Parse Tree for the various named entity classes.

F. AutoOntology: Automotive domain ontology for knowledge representation and search

The ontology (Fig 2.b) has been designed by analyzing several automotive domain software functional components at Bosch, with the aim to encapsulate the hierarchical structure and semantic relationships of automotive domain software systems and assist in software testing. As the entities can also have associated attributes such as range, data type, etc. property-based graph databases such as Neo4j are ideal for the KG in the automotive domain. AutoOntology captures the domain with 7 hierarchical classes: Functional Component, State, Unit Component, Signal, Value, Math/Logic, and User Input.

- Functional Component: Entity corresponding to a domain specific feature (e.g. Cruise Control).
- State: Entity corresponding to a state belong to the state machine of the functional component (e.g. Acceleration state of Cruise Control).
- Unit Component: Entity corresponding to a domain specific unit level functionality (e.g. MUX multiplexer).
- Signal: Entity used to store a domain specific variable (e.g. Vehicle speed). These are typically used to store pre-conditions, actions, and post-conditions.
- Value: Entity corresponding to a value that can be held by a Signal (e.g. Boolean values, Integer values).
- Math/Logic: Entity with mathematical and logical operations (e.g. Square root, AND, OR, NOT).
- User Input: Entity which is externally configured by the user (e.g. run time environment settings).

Relationships are directed edges in the KG to encapsulate the strict hierarchy between the entities. The entity serving as the tail of the edge corresponds to the Subject, the Predicate is the edge itself and the Object is the entity connected to the head of the edge. Functional level testing can be performed by extracting the relevant conditions connected at the state level of the functional component, whereas unit level testing fetches only those relevant conditions affected directly by the unit level components.

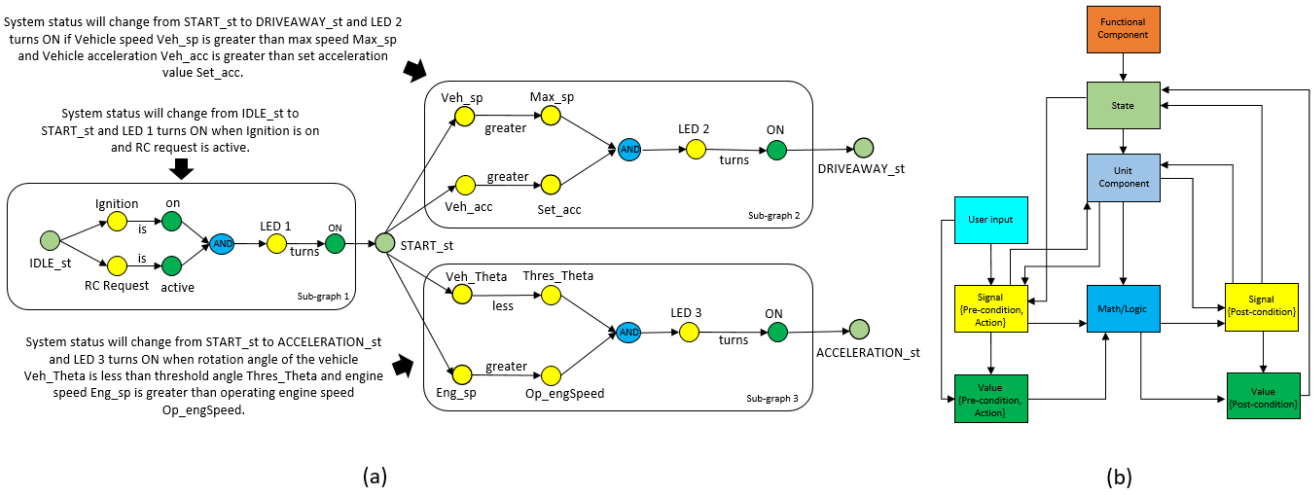

Fig. 2. a) Sample AutoKG b) AutoOntology.

TABLE I
TEST CASE GENERATION OUTPUT FOR A SAMPLE REQUIREMENT USING AUTOKG.

Requirement	For DRIVEAWAY_st, LED 2 will be ON if Vehicle speed is more than max speed and Vehicle acceleration is above set acceleration value.
Target Entities	DRIVEAWAY_st
Test Intent	Pre-conditions: [Vehicle speed is more than max speed; Vehicle acceleration is above set acceleration value] Post-conditions: [LED 2 will be ON]
Extracted Triplets	Pre-conditions: [(Subj: Veh_sp, Pred: greater, Obj: Max_sp); (Subj: Veh_acc, Pred: greater, Obj: Set_acc)] Post-conditions:[(Subj: LED 2, Pred: turns, Obj: ON)]
Context Sub-graph	Sub-graph 2 of Fig. 2.a
Dependency Sub-graph	Sub-graph 1 of Fig. 2.a
Test Case	Preconditions: [Sig_ign = True; Sig_RCReq = True; Veh_sp > Max_sp; Veh_acc > Set_acc] Post-conditions: [Sig_led2 = True]

III. TEST CASE GENERATION USING CONTEXT-BASED SEARCH ON AUTOKG

As the automotive domain contains a plethora of functions, states, signals, etc. which can quickly lead to an enormous KG, efficient querying to fetch relevant information for testing a requirement is vital. In this regard, we propose to reduce the search space using the test intent and target entities extracted from the requirement statement, which will serve as the context for generating the relevant test cases. Once the corresponding sub-graphs are obtained from the KG, further queries are executed to navigate and fetch dependency information for the requirement to generate a complete test case. The test case generation using a sample AutoKG (Fig. 2.a) can be seen in Table I. The test case generation consists of 5 phases:

- Phase 1: Capturing domain knowledge in the form of a Knowledge Graph from software documents.
- Phase 2: Tagging the requirement with the NER model for identifying the entity classes. Utilizing the AutoOntology heirarchy, the search space is reduced in a descending order starting from the entities at the highest level till the lowest level found in the requirement.
- Phase 3: Extract the test intent (pre-conditions, actions, and post-conditions) from the requirement using the CPT based algorithm. Each test intent component is mapped to a corresponding triplet in the KG by selecting the closest matching triplet based on the cosine similarity score between their AutoVec embeddings. The test intent extracted triplets are then used to find the relevant context sub-graph.
- Phase 4: Depending on the depth of testing to be performed, the connected dependency sub-graph is recursively traced back and fetched starting from the pre-conditions of the requirement for a particular test scenario.
- Phase 5: As the test intent of the requirement does not usually hold all the information to generate a complete test case, the information extracted from the dependency sub-graph (Phase 4) is sequentially merged with the context sub-graph (Phase 3) to generate a complete test case.

IV. CONCLUSION AND FUTURE WORK

In this paper we proposed improvisations for building an automotive domain KG from textual software documents that are primarily unstructured and sparse. The proposed pipeline builds a KG according to a domain-specific ontology, which is

then used for automatic generation of test cases for functional and unit level requirement statements. The future work that can be carried out are as follows: Firstly, the ontology can be extended to support higher levels of testing. Secondly, further work can be carried out to validate the information found in the requirement statements using the KG. This will help in identifying requirements that conflict with the information found in the KG.

REFERENCES

[1] R. Speer, J. Chin, and C. Havasi, "Conceptnet 5.5: An open multilingual graph of general knowledge," In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31, No. 1, 2017.

[2] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, and J. Krishnamurthy, "Never-ending learning," In Communications of the ACM, vol. 61, No. 5, 2018, pp.103–115.

[3] C. Henson, S. Schmid, A.T. Tran, and A. Karatzoglou, "Using a Knowledge Graph of Scenes to Enable Search of Autonomous Driving Data," In ISWC Satellites, 2019, pp.313–314.

[4] S. Schmid, C. Henson, and T. Tran, "Using Knowledge Graphs to Search an Enterprise Data Lake," In European Semantic Web Conference, Springer, Cham, 2019, pp.262–266.

[5] D.G. Rajpathak, 'An ontology based text mining system for knowledge discovery from the diagnosis data in the automotive domain," Computers in Industry, vol. 64, No. 5, 2013, pp.565–580.

[6] A. Nayak, V. Kesri, and R.K. Dubey, "Knowledge Graph based Automated Generation of Test Cases in Software Engineering," In Proceedings of the 7th ACM IKDD CoDS and 25th COMAD, 2020, pp.289–295.

[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," arXiv preprint arXiv:1301.3781, 2013.

[8] J. Pennington, R. Socher, and C.D. Manning, "Glove: Global vectors for word representation," In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp.1532–1543.

[9] J. Devlin, M.W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805.

[10] M.E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," arXiv preprint arXiv:1802.05365.

[11] M. Schuster and K. Nakajima, "Japanese and korean voice search," In 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2017, pp.5149–5152.

[12] A. Roy, Y. Park, and S. Pan, "Learning domain-specific word embeddings from sparse cybersecurity texts," arXiv preprint arXiv:1709.07470, 2017.

[13] M. Joshi, O. Levy, D.S. Weld, and L. Zettlemoyer, "BERT for coreference resolution: Baselines and analysis," arXiv preprint arXiv:1908.09091, 2019.

[14] C. Kadar and J. Iria, "Domain adaptation for text categorization by feature labeling," In European Conference on Information Retrieval, Springer, Berlin, Heidelberg, 2019, pp.424–435.

[15] G. Angeli, M.J.J. Premkumar, and C.D. Manning, "Leveraging linguistic structure for open domain information extraction," In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, vol. 1, pp.344–354.

[16] L. Del Corro and R. Gemulla, "Clausie: clause-based open information extraction," In Proceedings of the 22nd international conference on World Wide Web, 2013, pp.355–366.