Requirement analysis using artificial intelligence in software engineering

R.D.Budake[1], S.D.Bhoite[2]

[1]Assistant Professor, Department of Computer Science, Karmveer Hire Arts, Commerce, Science and Education College, Gargoti, Maharashtra, India

[2]Associate Professor, CSIBER, Kolhapur, Maharashtra, India

Abstract: Application software is all about logic , problem solving and , creativity. It is based on user requirments. Requirments are bridge between end-user and the software development team. Planning, data collection, analysis, design, programming, testing, implementation, and maintenance processes are a few of the procedures used in software development. During the software development process, planning and requirement analysis carry a significant level of risks. A problem that begins during the requirement analysis phase of the SDLC will persist throughout the entire life cycle of the software, making it a crucial phase of the SDLC. When automated technologies are used in the requirement analysis process, then it reduces the cost and duration of software development. Natural Language Processing (NLP) helps to identify problems in user requirements. Software requirements are classified and identified using supervised classification methods like SVM, K-Nearest Neighbour, and Naive Bayes Algorithm with text vectorization techniques like BoW and TF-IDF. The main purpose of this chapter is to identify user requirement problems during requirement analysis and provide AI techniques to overcome these problems.

Keywords: A.I., ML, NLP, Python, SDLC

Introduction:

Virtually every facet of business is being transformed by artificial intelligence, and this trend is also present in the software development industry. Due to inadequate requirements, the majority of software development companies deal with issues with rework and project defects. NLP produces better readability, unique, and visually appealing output of the user's requirement. Apply NLP techniques to the input document to extract the syntactical and semantic meaning. The goal of NLP is to process written languages so that a computer can comprehend them. NLP transforms unstructured input into structured data, which is helpful when constructing SRS documents. Entity Relationship Diagrams (ERDs) are created from end-user requirements by utilizing NLP to extract entities, their characteristics, types, and

relationships [1]. In this work, researchers break down the process of building a machine-learning model into some steps. 1. Understand and identify the data: A machine learning model built on training data, and then applying model to make accurate predictions from new data. 2. Collect and prepare data: In data preparation task include data collection, cleansing, Aggregation, Labelling, and Transformation among others. 3. Determine the model's features and train the model: It requires Model Technique Selection, Training, Hyperparameter Setting, Validation, Development and Testing, Algorithm Selection, Model Optimization. 4. Evaluate model performance: It include performance matric and final determination.

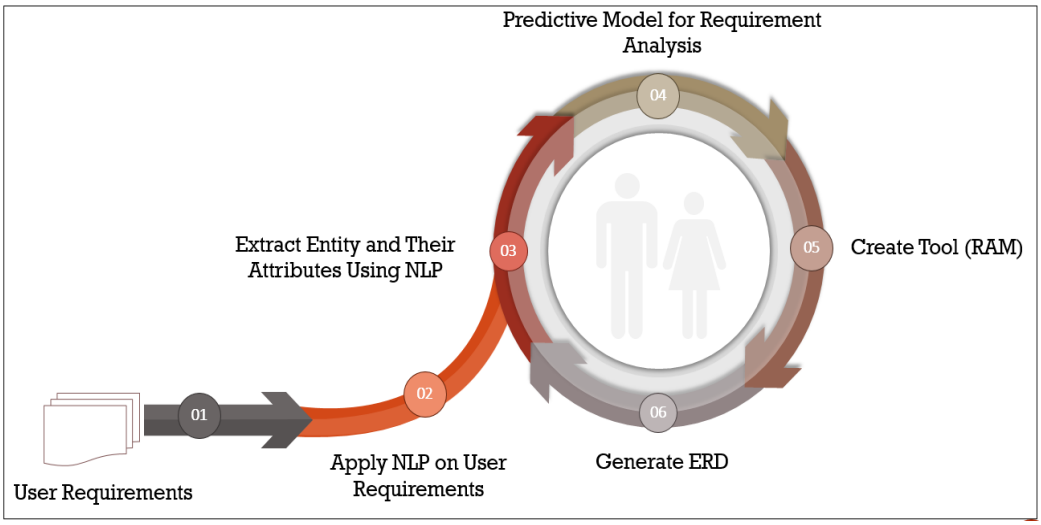


Fig.1. Life Cycle of Predictive Model for Requirment Analysis

Finally, researchers collect entities and attributes in order to generate diagrams such as ERDs.

Literature Review:

AI presently depends on artificial structures and is primarily concerned with replication, although automation of software engineering processes has a key advantage: when utilized appropriately, AI technologies enable human developers to expand their creative potential [2]. During their research, Yalla and Sharma used NLP techniques to automate the requirement and analysis phases of the Software Development Life Cycle (SDLC). Plain text is used as input for the NLP during the Requirement Analysis phase. They asserted that machine translation could translate textual artefacts into several languages. Textual information, or any other sort of information understood by both computers and people, must be mechanised. These authors provide a wide range of software engineering and natural language processing issues, as well as an integrated strategy that combines computer science and software engineering [3]. Each word in a phrase is assigned an abbreviation via the Part Of Speech (POS) method. Chunking and parsing are used to apply multiple possible analyses to the results, and then the Grammar Parsing procedure is used to assign a syntactic analysis to a string of words, resulting in the construction of a parsing tree. The complete study shows that the information collected from

the parsing tree results in the E.R. diagram in the end. [4]. Natural Language Processing (NLP) was used by Btoush and Hammad to extract E.R. components from natural language requirements. Natural language documents serve as the source of information for creating the E.R. data model. This method demonstrates how natural language processing techniques based on syntactic heuristics principles, such as tokenization, POS tagging, chunking, and parsing, may be used to extract composite attributes, cardinalities, weak attributes, and so on. [5]. Patil et al. discovered that machine learning approaches such as Naive Bayes, Support Vector Machines, Decision Trees, Random Forest, and deep learning techniques contributed positively to practically all phases of natural language processing. The accuracy indicates whether or not the machine understands natural language. Finally, they determined that the application of Machine Learning in Natural Language Processing had a significant favorable impact. [6]. Lilleberg et al. focused on support vector machines and word2vec for text classification to analyze the TF-IDF without stop words and TF-IDF with stop words. Word2vec brings extra semantic features that help in text classification [7]. Zijad, K. & Walid, M. applied a Support Vector Machine and lexical features for classifying requirements as functional (FR) and non-functional (NFR) in the dataset [8]. According to Nagarhalli, T. P., et al., almost all of these stages of natural language processing benefited from the use of machine learning techniques. The accuracy shows that natural language understanding by the machine or not. Finally, they conclude that, the use of Machine Learning in Natural Language Processing have had a very positive Impact [9]. Arshad Ahmad et al. presented to recognize or identify and classify the type of machine learning algorithms or techniques used for identifying software requirements on the Stack Overflow platform [10]. Mohd et al. used nine classification algorithms. The machine learning community develops these algorithms to detect which algorithm best suits the ambiguous software requirement specification [11]. Gokhan et al. suggested to extract quality attributes from non-functional requirements from in raw text using Support Vector Machines and classify quality attributes from raw text [12].

Design and development of system using NLP and ML:

There are several uses for information extraction. The endeavour to extract structured data from end user requirements during the requirement analysis phase of software development is a particularly relevant topic of current study. In this study, researchers feed the document's text, which represents the user requirement, and pass it to NLP. NLP has several phases to reach target representation. Following are the steps for generating Entity and their attributes using NLP.

Step 1. Input sentence to the NLP.

```
In [28]: ex="KH college offers Arts, Commerce, Science, Integrated Education and BCA Faculty with PG (M.A.) course under Shivaji University

In [29]: def preprocesssing(data):
             data = nltk.word_tokenize(data)
             data = nltk.pos_tag(data)
             return data

In [30]: data = preprocesssing(ex)
         print(data)

[('KH', 'NNP'), ('college', 'NN'), ('offers', 'NNS'), ('Arts', 'NNS'), (',', ','), ('Commerce', 'NNP'), (',', ','), ('Scienc
e', 'NNP'), (',', ','), ('Integrated', 'NNP'), ('Education', 'NNP'), ('and', 'CC'), ('BCA', 'NNP'), ('Faculty', 'NNP'), ('wit
h', 'IN'), ('PG', 'NNP'), ('(', '('), ('M.A', 'NNP'), ('.', '.'), (')', ')'), ('course', 'NN'), ('under', 'IN'), ('Shivaji',
'NNP'), ('University', 'NNP'), (',', ','), ('Kolhapur', 'NNP'), ('.', '.')]
```

Fig.2. Program for Token Identifier.

Step 2. Parse the sentence.

```
In [31]: pattern = 'NP: {<DT>?<JJ>*<NN>}'

In [*]:  cp = nltk.RegexpParser(pattern)
         cs = cp.parse(data)
         print(cs)
         cs.draw()
```

Fig.3. Program for Parsing the Sentence

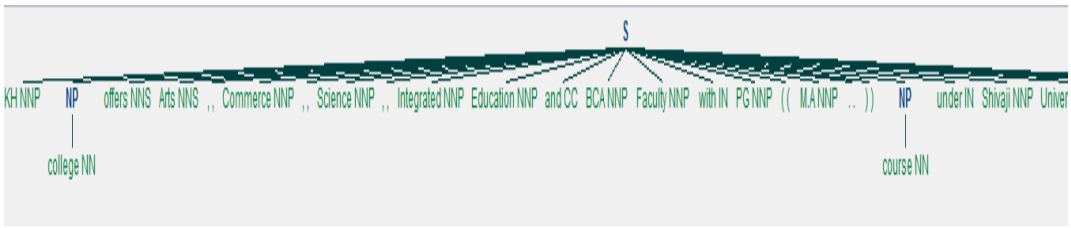


Fig.4. Output of Parsing the Sentence

Step 3: Input the sentences using Spacy.

```
import spacy
nlp=spacy.load('en_core_web_sm')

text='KH college offers department of Arts, Commerce, Science, Integrated Education and BCA Faculty with PG (M.A.) course under Shivaji University, Kolhapur.'

for token in nlp(text):
 print(token.text, '=>',token.pos_,'=>',token.tag_)
```

Fig.5. Program for Token Identifier using SpaCy.

Step 4: Display the structure of sentence using Displacy.

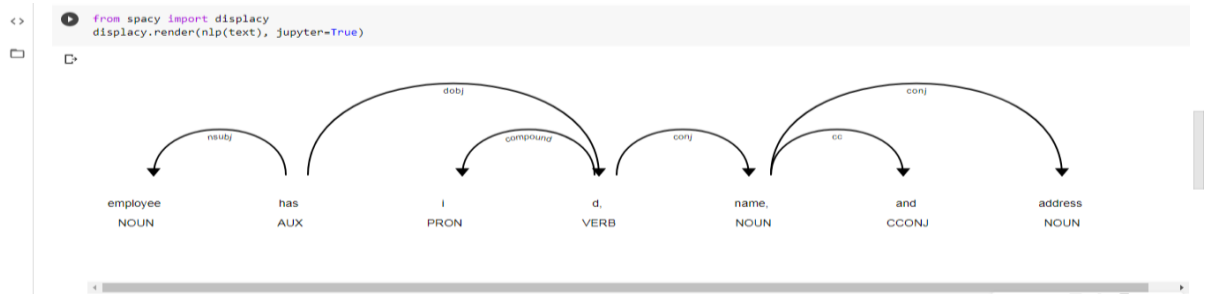


Fig. 6. Named Entity Recognition Using Displacy

Step 5: Collect Entiteis and their attributes using NLP.

```
cnt = 1
entity=[]
attribute=[]
for token in nlp(text):
  if token.tag_ == "NN":
    cnt = cnt+1
    if cnt == 2:
      entity.append(token)
      print("Our entity is",entity)
      print(token.text,'=>',token.dep_,'=>',token.head.text)
    elif cnt !=2:
      attribute.append(token)
      print("Our attribute is",attribute)
      print(token.text,'=>',token.dep_,'=>',token.head.text)
```

```
Our entity is [employee]
employee => nsubj => has
Our attribute is [name]
name => conj => d
Our attribute is [name, address]
address => conj => name
```

Fig. 7. Output of Entity and their Attributes

Step 6: Building a machine-learning model.

Vectorizer	Model	Hyper parameter(α)	Test AUC
BoW	Multinomial Naive Bayes	0.005	0.95
Tf-Idf	Multinomial Naive Bayes	0.1	0.8

Vectorizer	Model	Hyper parameter(K)	Test AUC
BoW	K-NN Brute	3	0.77
Tf-Idf	K-NN Brute	15	0.75
Avg Word2Vec	K-NN Brute	9	0.6
Tf-Idf Word2Vec	K-NN Brute	11	0.65

Vectorizer	Model	Hyper parameter(K)	Test AUC
BoW	K-NN Kd_tree	5	0.72
Tf-Idf	K-NN Kd_tree	15	0.75
Avg Word2Vec	K-NN Kd_tree	9	0.62
Tf-Idf Word2Vec	K-NN Kd_tree	11	0.65

Vectorizer	Model	L1 (alpha)	Test AUC (L1)	L2 (alpha)	Test AUC(L2)
BoW	SVM	0.0001	0.91	0.0001	0.91
Tf-Idf	SVM	0.0001	0.79	0.05	0.74

Fig. 8. The Accuracy of the Above Models with Their Parameters

Step 7: Deploy and test the model.

This step display automatically identifies and classifies functional and non-functional requirements as well as subclasses after deploying a machine-learning model on an unbalanced dataset.

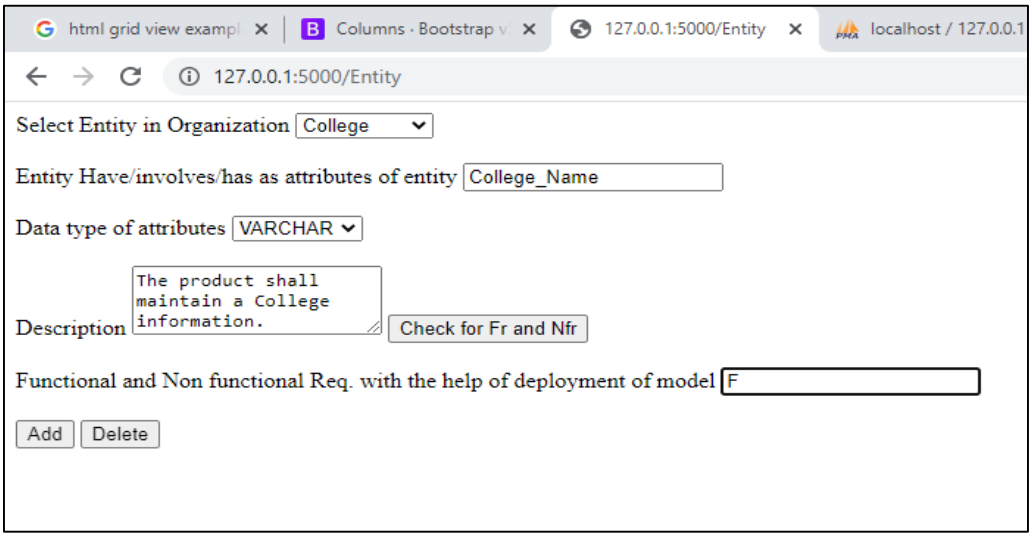
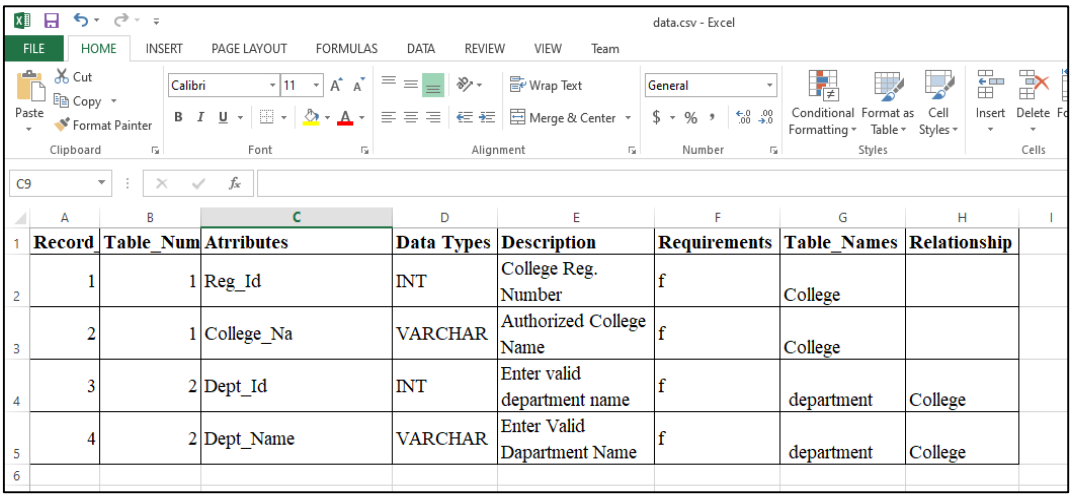


Fig. 9. Output of Entity with Functional or Nonfunctional Requirement

Step 8: Genetate ER- Diagram.



Record_	Table_Num	Attributes	Data Types	Description	Requirements	Table_Names	Relationship
1	1	Reg_Id	INT	College Reg. Number	f	College	
2	1	College_Na	VARCHAR	Authorized College Name	f	College	
3	2	Dept_Id	INT	Enter valid department name	f	department	College
4	2	Dept_Name	VARCHAR	Enter Valid Dapartment Name	f	department	College

Fig. 10. Retrieve data from the databsae and save it to CSV

```
from sqlalchemy import create_engine, MetaData
from sqlalchemy.orm import scoped_session, sessionmaker
from pydiagrams.SQLAlchemy import SQLAlchemyDiagram

# connect to database
engine = create_engine('sqlite:///RAM.db')
meta = MetaData()
meta.reflect(bind=engine)
session = scoped_session(sessionmaker(bind=engine))

# create the ERD
erd = SQLAlchemyDiagram(session, show_datatypes=True, show_indexes=True)
erd.create_diagram()

# save the ERD to file
erd.save('example.png')
```

Fig. 11. Fetch data from SQLite

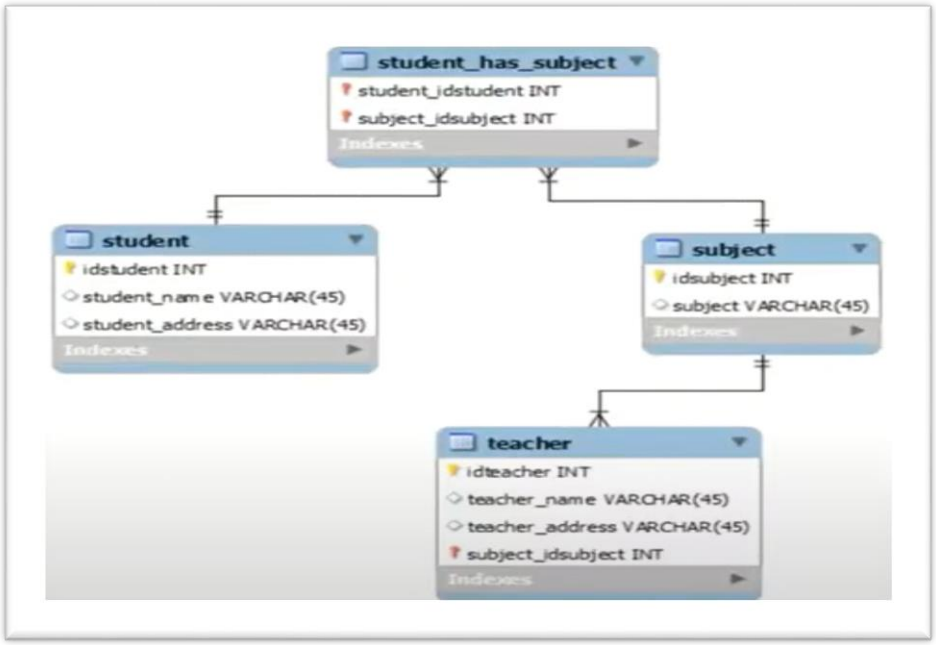


Fig.12. Display ERD

Result and Discussion:

The requirement analysis phase of the SDLC necessitates additional effort due to the collection of requirements, verification, and traceability. A combination of requirement-gathering techniques is intended to collect high-quality stakeholder requirements written down in English and distributed to other stakeholders. For ambiguity, subjectivity, incomplete logic, and volatile issues detection in requirement analysis, researchers used NLP techniques. The researcher applies ML algorithms to user requirements, to identify and classify the functional and non-functional requirements of software development. It is beneficial to developers, software designers, testers, and others involved in developing application software. The purpose of using NLTK & SpaCy is to extract entities, attributes, and methods from requirements written in

natural language by processing the requirements. To analyze requirements, it is recommended to use the spaCy and NLTK libraries of Natural Language Processing. When you use the AutoNormalize library in Python, it automatically detects and normalizes relationships in the dataset. To improve Natural Language Processing text by utilizing advanced algorithms such as k-Nearest Neighbor (K-NN), Naive Bayes (N.B.), and Support Vector Machine (SVM), as well as text vectorization techniques such as Bag of Words (BOW) and Term Frequency-Inverse Document (TF-IDF), Featurization & Machine Learning Models, ROC and AUC curves, Bi-Grams, and n-Grams in Python. It is useful for identifying and categorizing Functional Requirements (F.R.s), Non-Functional Requirements (NFRs), and their sub-classes in software development. In addition, the use of Natural Language Processing (NLP) and Machine Learning (ML) to generate ERD.

Conclusion:

Applying NLP to user requirements and extracting entity and their attribute is absolutely fantastic. Apply spaCy and displacy to a user requirements to make the output (phrase by sentence) more understandable, distinctive, and to visualise a dependency parsing or named entities. AI helps for classifying and identifying functional and non-functional requirements. It also assists in generating of ER-Diagram. This study aims to identify problems with requirements during the requirement analysis and provide NLP & ML techniques to overcome these problems as quickly as possible.

References:

1. R.D.Budake, S.D.Bhoite, “Extract Entity and Attributes from User Requirement by Applying on Natural Language Processing (NLP) Model”, STM Journal Recent Trends in Programming Language, Vol.8, ISSN: 2455-1821, 2021.
2. R.D.Budake, S.D.Bhoite, “Risk Analysis in Software Development Based on Artificial Intilligence (A.I.): Modern Approach”, Mukt Shabd Journal, ISSN: 2347-3150, UGC Care Group-I Journal, Vol.IX, June 2020.
3. Yalla, P., & Sharma, N. (2015). Integrating Natural Language Processing and Software Engineering. *International Journal of Software Engineering and Its Applications*, *9*(11), 127–136. https://doi.org/10.14257/ijseia.2015.9.11.12

4. Habib, M. (2019). On the Automated Entity-Relationship and Schema Design by Natural Language Processing. *The International Journal of Engineering and Science*, *8*(11), 42–48.
5. Btoush, E., & Hammad, M. (2015). Generating E.R. Diagrams from Requirement Specifications Based On Natural Language Processing. *International Journal of Database Theory and Application*, *8*(2), 61–70.
6. Patil, B. P., Kharade, K. G., & Kamat, R. K. (2020). Investigation on Data Security Threats & Solutions. *International Journal of Innovative Science and Research Technology*, *5*(1), 79–83.
7. Lilleberg, J., Yun, Z., & Yanging, Z. (2015). Support Vector Machines and Word2vec for Text Classification with Semantic Features. IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC). doi:10.1109/icci-cc.2015.7259377.
8. Zijad, K., &Walid, M. (2017). Utomatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning. IEEE 25th International Requirements Engineering Conference (RE), 490–495. https://doi.org/10.1109/RE.2017.8
9. Nagarhalli, T. P., Vaze, V., & Rana, N. K. (2021). Impact of Machine Learning in Natural Language Processing: A Review. 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV). doi:10.1109/icicv50876.2021.9388380
10. Arshad Ahmad, Chong Feng, Muzammil Khan, Asif Khan, Ayaz Ullah, Shah Nazir, and Adnan Tahir, Article 2020. "A Systematic Literature Review on Using Machine Learning Algorithms for Software Requirements Identification on Stack Overflow". Hindawi Security and Communication Networks Volume 2020, Article
11. Mohd Hafeez Osman and Mohd Firdaus Zaharin, (2018). "Ambiguous Software Requirement Specification Detection: An Automated Approach". 2018 ACM/IEEE 5th International Workshop on Requirements Engineering and Testing.
12. Gokhan Gokyer,Semih Cetin, Cevat Sener, Meltem T. Yondem, 2008. "Non-functional Requirements to Architectural Concerns: ML and NLP at Crossroads". The Third International Conference on Software Engineering Advances.978-0-7695-3372-8/08 $25.00 © 2008 IEEE.