Using AI and NLP for Tacit Knowledge Conversion in Knowledge Management Systems: A Comparative Analysis

Ouissale Zaoui Seghroucheni *, Mohamed Lazaar, Mohammed Al Achhab

Posted Date: 19 December 2024

doi: 10.20944/preprints202412.1714.v1

Keywords: Knowledge Management Systems; NLP; Tacit Knowledge, Explicit Knowledge; Knowledge Conversion

 doi:10.20944/preprints202412.1714.v1

Article

Using AI and NLP for Tacit Knowledge Conversion in Knowledge Management Systems: A Comparative Analysis

Ouissale Zaoui Seghroucheni [1,*], Mohamed Lazaar [1] and Mohammed Al Achhab [2]

[1] ENSIAS, Mohammed V University in Rabat, Morocco
[2] ENSA, Abdelamel Essaadi University in Tetouan, Morocco
* Correspondence: ouissale_zaouiseghroucheni@um5.ac.ma

Abstract: A Tacit knowledge, often implicit and embedded within individuals and organizational practices, plays a crucial role in knowledge management. Converting this tacit knowledge into explicit forms is vital for organizational effectiveness. This paper conducts a comparative analysis of NLP algorithms used for document and report mining in knowledge management systems (KMS) to facilitate the conversion of tacit knowledge. The focus is on algorithms that help extract tacit knowledge from documents and reports, as this knowledge is typically represented in semi-structured and document-based forms in natural language. NLP strategies, including text mining, information extraction, sentiment analysis, clustering, classification, recommendation systems, and affective computing, are explored for their effectiveness in this context. The paper provides a comprehensive analysis of the suitability of various NLP algorithms for tacit knowledge conversion within KMS, offering valuable insights for advancing research in this area.

Keywords: Knowledge Management Systems; NLP; Tacit Knowledge; Explicit Knowledge; Knowledge Conversion

1. Introduction

Tacit knowledge conversion is a critical aspect of Knowledge Management Systems (KMS) and plays a pivotal role in enhancing corporate decision-making. Tacit knowledge, being deeply rooted in individual experiences, skills, and intuition, is often challenging to articulate and share. Its conversion into explicit knowledge—a more structured and communicable form—is essential for organizations to harness this valuable resource effectively.

In the context of KMS, the conversion of tacit knowledge ensures that personal insights and expertise are systematically captured, stored, and made accessible across the organization. This process prevents knowledge silos, where critical insights remain confined to specific individuals, and mitigates the risk of losing valuable knowledge when employees leave. A well-integrated KMS that facilitates tacit knowledge conversion creates a centralized repository, enabling seamless sharing and retrieval of insights, thus fostering collaboration and innovation [1].

From a decision-making perspective, tacit knowledge conversion enhances the quality and depth of organizational decisions. Experienced employees often possess intuitive understanding and nuanced perspectives that are not readily apparent in quantitative data or explicit knowledge sources. By converting this knowledge into formats that others can access and utilize, corporations can ensure that decisions are informed by practical, real-world experiences alongside theoretical or data-driven analyses.

Moreover, the integration of tacit knowledge into decision-making processes promotes agility and responsiveness within organizations. In dynamic corporate environments, timely and well-informed decisions are critical. By leveraging a KMS that incorporates tacit insights, companies can respond more effectively to challenges and opportunities, gaining a competitive edge.

 doi:10.20944/preprints202412.1714.v1

However, the process of tacit knowledge conversion is not without challenges. Tacit knowledge is often subconscious, making it difficult for individuals to articulate. Additionally, resistance to sharing due to concerns about job security or loss of intellectual ownership can hinder the process. To address these issues, organizations must foster a culture that values and rewards knowledge sharing while providing tools and practices—such as mentorship programs, collaborative platforms, and workshops—to facilitate the capture and dissemination of tacit knowledge [3].

2. NLP and Tacit Knowledge Conversion

2.1. Natural Language Processing

Natural Language Processing (NLP) is the intersection of artificial intelligence (AI), computer science and linguistics, combining the computational power of computers with the complexities of human language to enable machines to understand, interpret, and generate text in a manner that mimics human linguistic capabilities. NLP draws heavily from both fields: computer science provides the algorithms, data structures, and machine learning techniques necessary for processing and analyzing large volumes of language data, while linguistics offers the theories and frameworks needed to understand the structure, meaning, and context of language. Linguistics, with its subfields such as syntax, semantics, pragmatics, and phonetics, helps in decoding the rules governing sentence structures, word meanings, and the relationships between words in context. Computer science contributes the computational tools—such as statistical models, neural networks, and deep learning algorithms—that make it possible to process language in a scalable, automated way. Together, these disciplines facilitate tasks such as machine translation, speech recognition, text generation, sentiment analysis, and more. Moreover, the evolution of NLP has been marked by significant advancements in machine learning and artificial intelligence, particularly with the introduction of deep learning techniques like the Transformer architecture [4], which have revolutionized language understanding and generation capabilities. As NLP systems continue to improve, the collaboration between computer science and linguistics grows more critical, as it bridges the gap between human language and machine understanding. The study of NLP requires not only a deep understanding of how language works but also the computational power to handle vast amounts of data and complex models, making it an interdisciplinary field at the forefront of AI research [5,6].

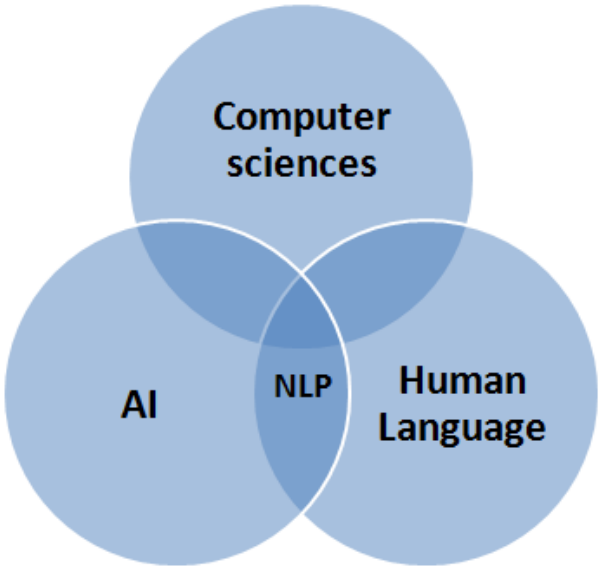

Figure 1. Relationship of NLP with AI and Human Language.

NLP has some unique aspects. The primary aspect is the lack of consistent, rigorously defined representations for the objects of our domain. Furthermore, different kinds of analysis appear to employ different underlying systems and resources to a considerable extent and are not seamlessly

 doi:10.20944/preprints202412.1714.v1

tied together at all. We can obtain some idea of the differences between design objectives by considering the question of what NLP is trying to extract from a text. Properties of text that are relatively easy for machine systems to extract seem to include properties that are manifestly not limited to texts in a single natural language, properties that are only indirectly related to the meanings that texts can be said to have by virtue of the beliefs of humanity or human-like speakers, and properties that are true of texts only relative to larger structures or processes.[7]

2.1. The Proposed NLP Pipeline for Tacit Knowledge Conversion

Our proposed NLP pipeline for tacit knowledge conversion is designed to bridge the gap between unstructured, implicit expertise and formalized, actionable insights. By leveraging advanced natural language processing techniques, the pipeline identifies, extracts, and organizes tacit knowledge embedded in textual data such as expert interviews, discussion transcripts, or informal communications. The pipeline integrates key stages including preprocessing, semantic analysis, and knowledge representation to ensure accuracy and relevance in the conversion process. Through methods like sentiment analysis, topic modeling, and named entity recognition, the system transforms implicit knowledge into explicit, structured formats suitable for reuse in knowledge decision-making, and foster knowledge sharing across teams.

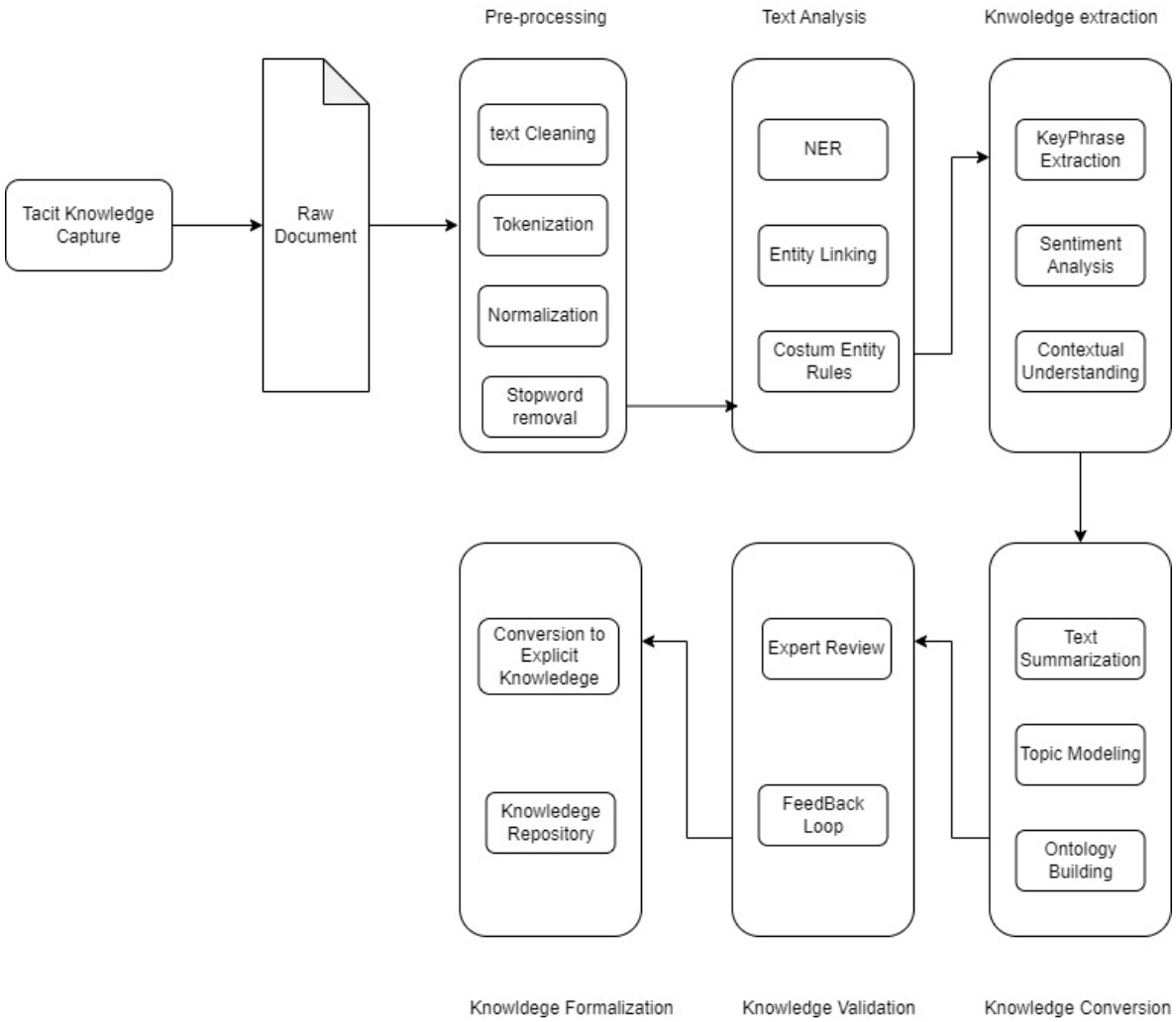

Figure 2. The proposed NLP pipeline for tacit knowledge convesrion.

✓ **Tacit Knowledge Capture**

The goal is to extract unstructured, informal knowledge from human sources such as interviews, meetings, expert discussions, voice recordings, and manuals. This is typically achieved through methods like speech-to-text conversion, transcription, and text mining, which transform the raw,

 doi:10.20944/preprints202412.1714.v1

spoken or written content into structured, analyzable data. These processes enable organizations to capture valuable insights from informal knowledge sources, making it easier to store, share, and analyze them for decision-making or knowledge management purposes.

✓ **Preprocessing**

- **Text Cleaning:** Eliminates irrelevant elements from the text, such as filler words, special characters, and non-standard language, to improve data quality.
- **Tokenization:** Breaks the text into smaller units, such as words or sentences, enabling easier analysis and processing.
- **Normalisation:** Converts all text to lowercase for uniformity and reduces words to their base or dictionary form, ensuring consistency across variations like tenses or plurals.
- **Stopword Removal:** Filters out frequently occurring but non-informative words allowing focus on meaningful content.

Together, these steps are essential for preparing text data for natural language processing tasks, ensuring clarity, structure, and relevance.

✓ **Text Analysis**

- **Named Entity Recognition (NER)**: Identifies and classifies important entities within text, such as names of people, organizations, locations, dates, and other specific items, to provide structured insights from unstructured data.
- **Costum Entity Rules**: Assigns grammatical categories like nouns, verbs, adjectives, and adverbs to words, enabling a deeper understanding of sentence structure and syntactic relationships.
- **Entity Linking**: Identifies and links words or phrases that refer to the same entity, such as recognizing that "he" or "his" refers to "John" in a passage, enhancing text coherence and context interpretation.

✓ **Knowledge Extraction**

- **Key Phrase Extraction**: Focuses on identifying critical concepts, skills, or insights embedded within text, summarizing its main ideas and enabling efficient information retrieval.
- **Sentiment Analysis**: Evaluates the emotional tone, attitude, or intent behind the text, categorizing it as positive, negative, or neutral. This is particularly useful for analyzing feedback, reviews, or opinions to gauge reactions and perspectives.
- **Contextual Understanding:** Leverages advanced natural language processing (NLP) models to uncover underlying patterns, relationships, and context in knowledge-sharing statements, ensuring a deeper comprehension of the information's intent and relevance.

✓ **Knowledge Formalization**

- **Text Summarization**: Generates concise summaries of textual data, distilling expert knowledge into its most essential concepts and ideas, making it easier to comprehend and disseminate. Techniques include extractive methods, which pull key sentences directly from the text, and abstractive methods, which generate new, simplified content.
- **Topic Modeling:** Identifies recurring themes or patterns within a large body of text, grouping related information into coherent topics. This approach helps in organizing unstructured data, uncovering hidden trends, and facilitating thematic analysis.
- **Ontology Building:** Constructs formal, structured representations of knowledge, such as frameworks or knowledge graphs, that define entities, their attributes, and relationships. This enables the organization and integration of information, promoting semantic understanding and interoperability across systems.

➢ **Knowledge Validation**

- **Expert Review**: Involves subject-matter experts assessing the extracted knowledge to verify its accuracy, relevance, and completeness. This step ensures that the insights align with domain-specific standards and real-world applicability.
- **Feedback Loop:** Establishes an iterative process where experts provide detailed feedback on the performance of natural language processing (NLP) models. This feedback is used to refine and retrain the models, enhancing their ability to extract, classify, and contextualize knowledge more effectively in subsequent iterations.

➢ **Knowledge Conversion**

➢ **Conversion to Explicit Knowledge**: Transforms tacit insights into structured and accessible formats, such as formal documentation, standard operating procedures, user manuals, or

 doi:10.20944/preprints202412.1714.v1

instructional guides. This process ensures that the extracted knowledge is easily understood, shared, and applied.

- **Knowledge Repository**: Centralizes the formalized knowledge within databases, content management systems, or knowledge management platforms. These repositories enable efficient storage, retrieval, and dissemination of information, fostering collaboration and ensuring the availability of valuable resources for future use.

3. NLP algorithms: Comparative Analysis

Natural Language Processing (NLP) algorithms are fundamental tools that enable computers to understand, process, and generate human language. These algorithms serve as the backbone for a wide array of applications, such as speech recognition, machine translation, sentiment analysis, chatbots, and information retrieval. NLP algorithms are designed to perform various language-related tasks, including tokenization, lemmatization, part-of-speech tagging, named entity recognition (NER), and text summarization. Over the years, advancements in machine learning and deep learning techniques, especially with the rise of transformer-based models like BERT, GPT, and RoBERTa, have significantly enhanced the ability of machines to comprehend the complexities of human language. These models not only interpret syntax but also understand context, disambiguating meanings and recognizing relationships within text. The growing sophistication of NLP algorithms has made them indispensable in fields like healthcare, finance, legal analysis, and knowledge management [8].

However, with the wide variety of NLP algorithms available, selecting the most suitable one for a specific task can be challenging. To address this, we will conduct a comparative analysis of various NLP algorithms based on three key criteria: **accuracy**, **speed**, and **resource requirements**. Accuracy will be measured by the algorithm's ability to produce correct and meaningful results; speed will evaluate how quickly the algorithm performs, especially in real-time or large-scale applications; and resource requirements will consider the computational resources, such as memory and processing power, needed to run the algorithm efficiently. This focused analysis will provide insights into the best-suited algorithms for different stages of NLP tasks, ensuring optimal performance and scalability in real-world applications.

- Text cleaning

Text cleaning is a crucial step in data preprocessing, especially in fields like Natural Language Processing (NLP), where unstructured text is processed to extract meaningful information. The following table provides a comparative analysis of NLP algorithms for this task based on various criteria such as accuracy, resource requierement and speed.

Regular expressions, or regex, are character sequences that define patterns for searching and manipulating text. They play a crucial role in natural language processing (NLP) tasks such as text searching, data extraction, and format validation. Regex enables users to specify intricate string-matching rules, making them invaluable for organizing, cleaning, and efficiently parsing textual data [9].

Spelling correction is the process of identifying and rectifying errors in text to enhance its accuracy. In natural language processing (NLP), this involves techniques such as edit distance algorithms, like Levenshtein Distance, which calculate the number of edits required to transform one word into another. Contextual models, including transformer-based tools, analyze surrounding text to predict the most likely intended word. Additionally, dictionaries or lexicons are used to validate words and suggest accurate alternatives, ensuring the text adheres to standard spelling conventions [10].

Noise removal in NLP involves filtering out irrelevant or unhelpful elements from text to retain only meaningful information for tasks like analysis or modeling. Typical forms of noise include special characters that often lack semantic significance, stop words such as "the" or "is," which carry minimal contextual value, and numerical values that might be unnecessary in certain applications [11].

 doi:10.20944/preprints202412.1714.v1

Table 1. The performance of NLP Algorithms for text cleaning.

Algorithm	Accuracy	Speed	Resource Requirement
Regular Expressions	Moderate	fast	Low
Spelling Correction	High for common spelling errors	Moderate	Moderate
Noise Removal	Low to Moderate	Fast	Low

Comparative analysis : Regular expressions are fast and flexible but may require manual tuning and can be prone to errors for more complex text cleaning. Spelling correction works well but adds computational overhead. Noise removal methods (like lowercasing) are simple and effective, though they may miss important information.

- Tokenization

Tokenization is a fundamental step in Natural Language Processing (NLP) that involves breaking down text into smaller units, called tokens. Tokens can be words, phrases, or even characters, depending on the specific application. This step is crucial for text analysis as it helps transform raw text into a structured format that algorithms can process.

Whitespace tokenization is a straightforward method in NLP where text is divided into tokens based on spaces, tabs, and newline characters. It operates under the assumption that words are primarily separated by spaces, which makes it suitable for simple tokenization tasks. However, this approach struggles with punctuation and special characters and may not work well when words are written without spaces, such as "NewYork," where it would incorrectly treat the combined form as a single token. Consequently, while whitespace tokenization is efficient for basic use cases, it has limitations when handling more complex text structures [12].

The Treebank Tokenizer is a more advanced tokenization tool that employs predefined rules and models to accurately process text. It is particularly adept at managing punctuation, contractions, and other special cases, such as splitting "don't" into "do" and "n't." Unlike simpler tokenization methods, it ensures consistency with syntactic structures, making it especially useful in syntactic parsing tasks. It is commonly employed with Treebank-style annotations, which include part-of-speech tagging and syntactic parsing, to maintain precise and coherent tokenization in natural language processing tasks [13].

Subword tokenization decomposes words into smaller, more meaningful units called subwords, which helps improve handling of rare or previously unseen words in tasks like machine translation and language modeling. Techniques such as Byte Pair Encoding (BPE), SentencePiece, and WordPiece break down words into subword components, allowing for better generalization across languages. For instance, the word "unhappiness" could be split into "un" and "happiness" or further into even smaller parts like "un," "happi," and "ness." This method ensures more flexible handling of diverse vocabulary and improves model performance, especially in multilingual contexts [14].

Table 2. The performance of NLP Algorithms for Tokenization.

Algorithm	Accuracy	Speed	Resource Requierement
Whitespace Tokenzization	Low	Fast	Low
Treebank Tokenizer	High	Moderate	Moderate
Subword Tokenization	High	Moderate	High

Analysis: Whitespace tokenization is fast but lacks accuracy for more complex sentence structures. Treebank tokenization is more accurate but slower and less scalable. Subword

 doi:10.20944/preprints202412.1714.v1

tokenization, used in models like BERT, provides high accuracy but requires significant resources and is computationally expensive.

- Normalization

Normalization is a preprocessing technique in NLP that converts text into a standard or canonical form. It is essential for ensuring consistency in the data and improving the performance of downstream tasks like text classification, sentiment analysis, or machine translation. By reducing variability in text, normalization makes it easier for algorithms to analyze and extract meaningful insights.

Text preprocessing techniques like **lowercasing**, **stemming**, and **lemmatization** are essential for normalizing text in natural language processing (NLP). **Lowercasing** standardizes words by converting all characters to lowercase, helping reduce case-related inconsistencies. **Stemming** reduces words to their root forms using algorithms like Porter or Snowball, although it can result in non-dictionary forms . In contrast, **lemmatization** is more precise, transforming words into their base form based on context and meaning (e.g., "better" to "good" and "running" to "run"), and is particularly useful for tasks requiring semantic accuracy. While stemming is computationally faster, lemmatization is preferred for tasks where context and correctness are critical [15].

Table 3. The performance of NLP Algorithms for Normalization.

Algorithm	**Accuracy**	**Speed**	**Resource Requierement**
Lowercasing	Moderate	Fast	Low
Stemming	Low to Moderate	Fast	Low
Lemmatization	High	Slow	Moderate

Analysis: Lowercasing is a quick and simple process but can lead to loss of context. Stemming is fast and works well for many use cases but can lead to non-dictionary words. Lemmatization is more accurate as it returns valid words but is slower and requires additional resources.

- Stopword Removal

Stopword removal is a preprocessing technique in NLP that involves eliminating common words that usually carry little semantic meaning. These words, called **stopwords**, include terms like "the," "is," "in," "and," or "of." The goal is to reduce noise in the text, enabling algorithms to focus on more informative and meaningful words.

Predefined lists in NLP refer to sets of words used for specific tasks, such as stopwords or domain-specific vocabularies, which help streamline processes like tokenization and information extraction. **TF-IDF (Term Frequency-Inverse Document Frequency)** is a statistical method used to assess the importance of words within a document relative to a corpus, highlighting words that are frequent in specific documents but rare across others, making it useful for text mining, search engines, and information retrieval [16].

Table 4. The performance of NLP Algorithms for stopword removal.

Algorithm	**Accuracy**	**Speed**	**ResourceRequierement**
Predefind Lists	Moderate	Fast	Low
TF-IDF	High	Moderate	High

Analysis: Predefined lists are very fast but may miss important stopwords in context. TF-IDF is more context-sensitive but computationally more expensive.

- Named Entity Recognition (NER)

 doi:10.20944/preprints202412.1714.v1

Rule-based models in NLP rely on predefined linguistic rules to process language, often used for tasks like parsing and named entity recognition. While they can be highly accurate in structured settings, they lack flexibility and struggle with ambiguity. Statistical models, such as Hidden Markov Models (HMMs) and Naive Bayes, use probabilistic methods to infer relationships within data, offering more adaptability in tasks like part-of-speech tagging and machine translation. Deep learning models, particularly transformer-based architectures like BERT and GPT, have significantly advanced NLP by automatically learning from vast datasets, excelling in tasks like text generation, summarization, and question answering with impressive accuracy [17].

Table 5. The performance of NLP Algorithms for NER.

Algorithm	Accuracy	Speed	Resource Requierment
Rue-Based	Low to Moderate	Fast	Low
Statistical Models	Moderate	Moderate	Moderate
Deep Learning	High	Slow	High

Analysis: Rule-based systems are fast and simple but limited by predefined rules. Statistical models like CRFs offer better accuracy but require labeled data. Deep learning models such as BERT deliver the highest accuracy but require significant computational resources and are slower.

- Part-of-Speech (POS) Tagging

Hidden Markov Models (HMMs) are probabilistic models that assume an underlying hidden state evolves over time, often used in sequential tasks like speech recognition or part-of-speech tagging. Conditional Random Fields (CRFs) are similar but focus on structured prediction, considering the entire input sequence when making predictions. Deep learning, particularly with models like RNNs, LSTMs, and transformers, surpasses HMMs and CRFs in capturing complex relationships in data, making it ideal for tasks that require understanding context and dependencies within large datasets, such as natural language processing [18].

Table 6. The performance of NLP Algorithms for Part-of-Speech (POS) Tagging.

Algorithm	Accuracy	Speed	Resource Requierement
HMM	Moderate	Fast	Low
CRF	High	Moderate	Moderate
Deep Learning	High	Slow	High

Analysis: HMMs are fast and simple but may not capture long-range dependencies. CRFs offer better performance, especially in sequence prediction tasks. BiLSTM-CRF or transformer-based models provide the best accuracy but are resource-heavy and slow.

- ✓ Coreference Resolution

Rule-based models in NLP rely on predefined linguistic rules for tasks like parsing and named entity recognition, but they lack flexibility and struggle with ambiguity. Machine learning models, such as decision trees and SVMs, adapt to data and are more versatile than rule-based models, making them suitable for a wide range of tasks. Deep learning models, including RNNs, CNNs, and transformers, automatically learn complex patterns from large datasets, outperforming both rule-

 doi:10.20944/preprints202412.1714.v1

based and machine learning models in tasks like text generation, translation, and summarization due to their ability to capture context and dependencies [19].

Table 7. The performance of NLP Algorithms for coreference resolution.

Algorithm	Accuracy	Speed	Resource Requierement
Rule-based	Low to oderate	Fast	Low
Machine Learning	High	Moderate	Moderate
Deep Learning	Very High	Slow	High

Analysis: Rule-based systems are fast and simple but not as effective for complex coreferences. Machine learning models (e.g., CRFs) perform well but require labeled data. BERT-based models offer very high accuracy at the cost of significant resources and time.

✓ Key Phrase Extraction

TF-IDF, RAKE, and TextRank are common algorithms in Natural Language Processing used for tasks like keyword extraction, summarization, and information retrieval. TF-IDF calculates the importance of words in a document by combining term frequency (TF) with inverse document frequency (IDF), highlighting terms that are significant within specific contexts. RAKE focuses on extracting multi-word key phrases by analyzing word co-occurrence patterns, making it efficient for large datasets. TextRank, inspired by the PageRank algorithm, builds a graph of words or phrases, using their relationships to rank their importance for tasks like summarization and keyword extraction. These techniques are fundamental in text analysis and content processing [20].

Table 8. The performance of NLP Algorithms for key phrase extraction.

Algotithm	Accuracy	Speed	Resource Requierement
TF-IDF	Moderate	fast	Low
RAKE	Moderate	Fast	Low
TextRank	High	Moderate	Moderate

Analysis: TF-IDF is quick and simple but may miss contextual relevance. RAKE is fast and can handle short texts but might miss key phrases in larger documents. TextRank performs well in terms of semantic relevance but is slower and more complex to implement.

✓ Sentiment Analysis

Lexicon-based approaches in NLP use predefined word lists or lexicons to perform tasks like sentiment analysis, relying on word matching to determine meaning. While simple and interpretable, they can struggle with context and ambiguity. Machine learning methods, on the other hand, learn from labeled data to make predictions and are more adaptable than lexicon-based models. Deep learning, a subset of machine learning, goes further by using neural networks with multiple layers to automatically learn complex language patterns, making it especially effective for handling tasks like text generation, translation, and sentiment analysis due to its ability to understand context and semantic nuances [21].

Table 9. The performance of NLP Algorithms for sentiment analysis.

Algorithm	Accuracy	Speed	Resource Requierement
Lexicon-Based	Moderate	Fast	Low

 doi:10.20944/preprints202412.1714.v1

Machine Learning	High	Moderate	Low
Deep Learning	Very High	Slow	High

Analysis: Lexicon-based methods are fast but lack accuracy in complex sentiment scenarios. Machine learning models (e.g., Naive Bayes, SVM) are more accurate but require labeled data. Deep learning methods (e.g., LSTM or BERT) provide very high accuracy but demand significant computational power and time.

✓ Text Summarization

Extractive summarization involves selecting important sentences or phrases directly from the original text to create a concise summary, often using techniques like TF-IDF, TextRank, or supervised learning models. It is computationally less intensive but may lack coherence as it simply extracts fragments. In contrast, abstractive summarization generates summaries by paraphrasing and rephrasing the original content, often employing deep learning models such as sequence-to-sequence networks or transformers. While abstractive methods are more complex and resource-demanding, they provide more coherent and human-like summaries. Both approaches have distinct advantages, with extractive methods being simpler and faster, while abstractive methods produce more natural summaries [22].

Table 10. The performance of NLP Algorithms for text summarization.

Algorithm	Accuracy	Speed	Resource Requierement
Extractive Summarization	Moderate	Fast	Low
Abstractive Summarization	High	Slow	High

Analysis: Extractive summarization is simpler and faster but may lack coherence. Abstractive summarization provides better and more coherent results but is computationally expensive and slower.

✓ Semantic analysis

Word embedding models, such as Word2Vec, GloVe, and FastText, map words into dense vectors, capturing their semantic meaning, and are widely used for tasks like sentiment analysis and machine translation. Recurrent Neural Networks (RNNs), including LSTMs and GRUs, handle sequential data by maintaining memory of previous inputs, making them suitable for tasks like language modeling and speech recognition. GPT (Generative Pre-trained Transformer), a powerful model for text generation, utilizes an autoregressive approach and pre-training on large datasets to produce highly coherent text. SBERT (Sentence-BERT) extends BERT by generating efficient sentence embeddings, which are particularly effective for semantic similarity and clustering tasks. These models represent key advancements in NLP, improving how machines understand, generate, and manipulate language [23].

Analysis : GPT and SBERT stand out in terms of accuracy, with GPT excelling in tasks requiring contextual understanding and SBERT demonstrating superior performance in semantic similarity tasks. When it comes to speed, word embedding models like Word2Vec are the fastest due to their simplicity and lightweight nature, while SBERT, optimized for embedding generation, offers faster inference compared to GPT. However, resource requirements vary significantly: word embeddings are highly lightweight and efficient, making them ideal for resource-constrained applications, whereas GPT, with its massive size and transformer-based architecture, demands substantial computational power for both training and inference.

 doi:10.20944/preprints202412.1714.v1

Table 11. The performance of NLP Algorithms for semantic analysis.

Algorithm .	Accuracy	Speed	Resource Requierment
Word Embedding Models	Moderate	Fast	Low
Recurrent Neural Networks (RNNs)	High	Slow	High
GPT	Very high	Slow	High
SBERT	High	Fast	Moderate

After conducting a comprehensive comparative analysis of various NLP algorithms, we propose utilizing the algorithms listed in the table for tacit knowledge conversion. These algorithms were selected based on their proven effectiveness, adaptability, and performance across a wide range of tasks, from text preprocessing to advanced understanding and summarization.

Table 12. The proposed NLP algorithms for tacit knowledge conversion.

Task	The Proposed NLP Algorithm for tacit knwoledge conversion
Text Cleaning	Regex-based Cleaning
Tokenization	Hugging Face Tokenizers
Stopword Removal	SpaCy Stopwords
Named Entity Recognition (NER)	BERT-based NER
Entity Linking	DBpedia Spotlight
Keyphrase Extraction	KeyBERT
Sentiment Analysis	Transformer-based Models
Clustering	DBSCAN
Contextual Understanding	Sentence-BERT
Text Summarization	BART (Abstractive Summarization)

This section may be divided by subheadings. It should provide a concise and precise description of the experimental results, their interpretation, as well as the experimental conclusions that can be drawn.

3.1. SBERT for Tacit Knowledge Conversion

- SBERT

After conducting a thorough comparative analysis of various Natural Language Processing (NLP) algorithms and selecting the most suitable one for each specific task, we now focus on one of the most pivotal models in the field for tacit knowledge conversion: SBERT (Sentence-BERT). SBERT stands out for its ability to generate high-quality sentence embeddings that effectively capture the semantic meaning of text. This makes it an essential tool for tasks such as knowledge retrieval, information clustering, and, most importantly, the conversion of tacit knowledge into explicit knowledge. In this section, we will provide an overview of SBERT's architecture, highlighting its relevance in knowledge management systems especially in tacit knowledge conversion.NLP systems that handle large volumes of text require effective sentence embeddings, and one of the most advanced algorithms for this task is SBERT (Sentence-BERT). Using Siamese twin networks, SBERT minimizes the distance between embeddings of similar examples and maximizes it for dissimilar ones. This contrastive learning approach enables SBERT to capture both word order and semantics in text sequences. While earlier models focused on context-based word embeddings, SBERT

 doi:10.20944/preprints202412.1714.v1

combines both word-level and sentence-level understanding [24] It consistently outperforms previous models across a range of tasks, particularly when trained on larger datasets, showcasing improved scalability and performance.

SBERT has demonstrated remarkable success on benchmarks such as Semantic Textual Similarity and information retrieval tasks. Its ability to generate high-quality sentence or paragraph embeddings is crucial for Phase 1 of Nonaka's SECI framework, which focuses on converting tacit knowledge. Unlike traditional BERT models, which require repetitive queries for each sentence, SBERT efficiently pairs a query sentence with multiple document sentences without recalculating the query embedding each time, making it scalable for real-time applications. By modifying BERT's architecture to include pooling operations, SBERT creates sentence embeddings that preserve the semantic context of the entire sentence while remaining fast and scalable. As a result, SBERT is now regarded as one of the most effective methods for generating sentence embeddings in NLP tasks.

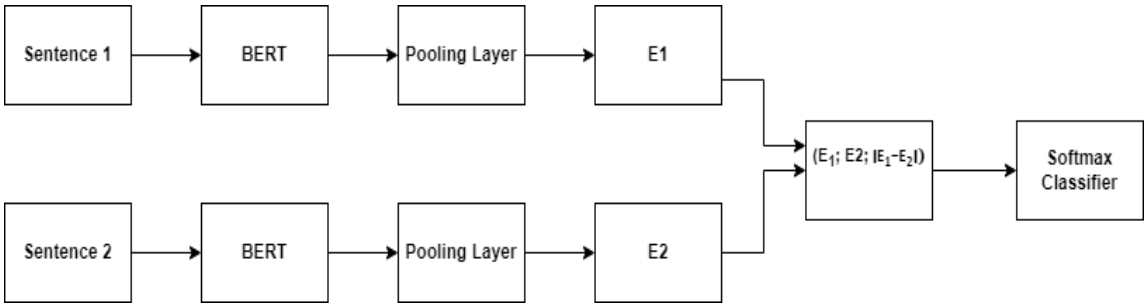

Figure 3. SBERT architecture.

- Input Representation:
  Two sentences S_1 and S_2 are given as inputs.
  Tokenize S_1 and S_2 using a shared tokenizer

- Shared Encoding via BERT:
  Pass S1 and S2 through identical pre-trained BERT models $BERT_{shared}$.

$$T1=BERT_{shared}(S_1) \quad (1)$$

$$T2=BERT_{shared}(S_2) \quad (2)$$

Outputs:

$$T1= [t_{11}, t_{12},\ldots, t_{1n}]: \text{Contextualized token embeddings for } S_1. \quad (3)$$

$$T2= [t_{21}, t_{22},\ldots, t_{2m}]: \text{Contextualized token embeddings for } S_2. \quad (4)$$

- Pooling Layer:
  Aggregate token embeddings into fixed-size sentence embeddings:

$$E1=Pooling(T_1) \quad (5)$$

$$E2=Pooling(T_2) \quad (6)$$

Common pooling strategies:
Use the [CLS] token embedding.
Compute the mean of all token embeddings.
Compute the max value across embeddings.

- Similarity/Task Computation:
  Compare the sentence embeddings E_1 and E_2 using a similarity metric or task-specific logic.
  For similarity: Compute cosine similarity Sim (E_1, E_2).
  For classification: Concatenate E_1 and E_2, and pass through a classifier:

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2024 doi:10.20944/preprints202412.1714.v1

$$\text{Output}=\text{Classifier}([E_1; E2; |E_1-E_2|]). \quad (7)$$

➢ Output:

Generate embeddings E1 and E2 that are:

- Fixed-sized.
- Semantically rich.
- Optimized for downstream tasks.

Tacit Knowledge Conversion involves transforming unstructured, implicit knowledge into structured, explicit forms that can be shared and utilized.

SBERT can be used to process and compare text data, cluster similar concepts, or identify implicit patterns from unstructured content like documents, discussions, or interview transcripts. Below is a conceptual SBERT-based architecture tailored for tacit knowledge conversion.

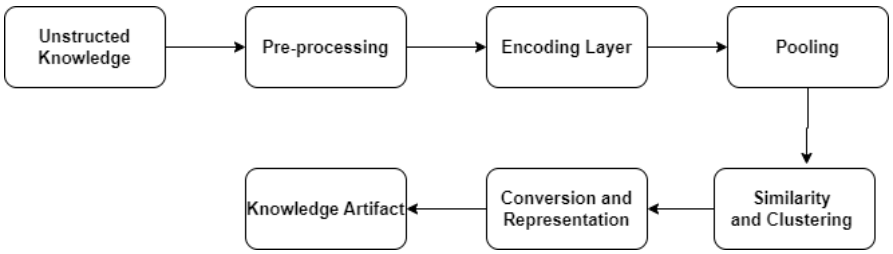

Figure 4. The proposed SBERT Architecture for tacit knowledge conversion.

➢ Input Layer (Unstructured Text Sources):

Unstructured text inputs can be collected from various sources, including employee feedback gathered through surveys, performance reviews, and suggestion boxes, as well as meeting transcripts derived from audio or video recordings, handwritten notes, or summaries. Additionally, research papers and case studies provide valuable text data, often sourced from academic databases or organizational archives. Informal communication, such as emails and chat logs, further contributes to unstructured text inputs, offering insights from casual interactions within teams or across organizations.

Let S_1, S2,…, Sn be the set of unstructured sentences, each representing an instance of tacit knowledge:

$$S_i \in R^d \quad \text{for} \quad i=1,2,\ldots, n$$

where:

S_i is a sentence, expressed as a sequence of words or tokens (w_1, w_2,…, w_m) for a sentence S_i, and d represents the dimensionality of each token embedding.

➢ Preprocessing Layer:

Tokenization and normalization involve preprocessing text data to enhance its usability for analysis. This includes removing noise, such as redundant phrases and formatting inconsistencies, to ensure cleaner inputs. Sentences can then be tokenized using advanced tools like SBERT's tokenizer, which enables the extraction of key phrases or thematic sentences for more focused analysis.

The sentence S_i is tokenized into subword units w_1,w_2,…,w_m and mapped into embeddings using a tokenizer. This generates token embeddings:

$$E(S_i)= [e1, e2,\ldots, em] \quad \text{where} \quad ej \in R^k \quad (8)$$

where:

e_j is the embedding of token w_j, and k is the embedding dimension.

➢ Encoding Layer (Shared SBERT Backbone):

Each sentence or segment is processed using a pre-trained SBERT model to generate dense sentence embeddings. These embeddings capture the semantic meaning of the text, enabling a more nuanced representation of the underlying information.

 doi:10.20944/preprints202412.1714.v1

$$T(S_i)=BERT\ (E(S_i))=[t_1,t_2,\ldots,t_m] \quad (9)$$

where:

$t_j \in R^k$ is the contextualized embedding of token w_j.

➢ Pooling

The sentence embedding e_i is generated by applying a pooling function on the contextualized token embeddings $t_1, t_2,\ldots,t_m$. Pooling can be done in multiple ways:

⇗ Mean Pooling:

$$e_i = \frac{1}{m}\sum_{j=1}^{m} \quad (10)$$

$e_i e_i$ is the final sentence embedding, which is the mean of all token embeddings.

⇗ Max Pooling:

$$e_i=\max\ (t_1, t_2,\ldots, t_m) \quad (11)$$

e_i is the element-wise maximum of the token embeddings.

Clustering and Similarity Analysis (Knowledge Aggregation):

To find similarities between sentences S_i and S_j, we compute the cosine similarity between their sentence embeddings ei and ej:

$$Sim\ (S_i, Sj)=\frac{ei.ej}{\|ei.ej\|} \quad (12)$$

where:

$e_i \cdot e_j$ is the dot product between the two sentence embeddings.

$\|ei\|$ and $\|ej\|$ are the L2 norms (magnitudes) of the embeddings.

Once the sentence embeddings are computed, clustering algorithms (e.g., k-means) are used to group similar sentences together. The k-means algorithm involves the following steps:

⇗ Initialize centroids

$C_1, C_2,\ldots, C_k$ (initial centroid locations)

⇗ Assign each sentence to the nearest centroid

$$Cluster\ (Si))=\arg_{c\in\{C1,\ C2,\ldots,Ck\}}\ \min\ \|\mathbf{e}i-c\|^2 \quad (13)$$

⇗ Update centroids based on assignments

$$Cj = \frac{1}{|Clusterj|}\sum_{Si\in Clusterj} ei \quad (14)$$

where:

$Cluster_j$ is the set of sentences assigned to centroid C_j.

➢ Conversion and Representation:

Tacit knowledge is transformed into explicit forms, such as rules, guidelines, or models, through techniques like embedding-based clustering and summarization. Additionally, multiple sources of explicit knowledge can be combined to create higher-order concepts, with embeddings used to pinpoint redundancies and uncover synergies among the different knowledge sources.

➢ Output Layer (Knowledge Artifacts):

 doi:10.20944/preprints202412.1714.v1

Structured knowledge artifacts can be created in various forms, including concise summaries, organized taxonomies, and comprehensive knowledge graphs, to systematically represent and manage information.

This is a Pseudocode that demonstrates the process of clustering unstructured text data using Sentence-BERT (SBERT) embeddings and the KMeans clustering algorithm. Here's a step-by-step explanation:

```
documents = ["Tacit knowledge is difficult to express.",

             "Effective teams often learn by doing.",

             "Collaboration fosters innovation."]

from sentence_transformers import SentenceTransformer

model = SentenceTransformer ('paraphrase-MiniLM-L6-v2')

embeddings = model.encode (documents)

from sklearn.cluster import KMeans

Num_clusters = 2  # Adjust based on data

Clustering_model = KMeans (n_clusters=num_clusters)

clustering_model.fit (embeddings)

Cluster_labels = clustering_model.labels_

Clusters = {i: [] for i in range (num_clusters)}

For idx, label in enumerate (cluster_labels):

    Clusters [label].append (documents [idx])

For cluster, sentences in clusters.items ():

    Print (f"Cluster {cluster} :")

    For sentence in sentences:

        Print (f"  - {sentence}")
```

 doi:10.20944/preprints202412.1714.v1

4. Conclusion

This paper has explored the potential of Natural Language Processing (NLP) algorithms for converting tacit knowledge into explicit knowledge within Knowledge Management Systems (KMS). The comparative analysis of various NLP techniques, including text mining, information extraction, sentiment analysis, and recommendation systems, has highlighted their effectiveness in extracting knowledge from semi-structured and document-oriented sources. These methods have shown promise in supporting organizations by transforming implicit knowledge embedded in documents and reports into structured, actionable insights. In future work, we aim to extend this research by developing a chatbot-based tool designed to facilitate the real-time conversion of tacit knowledge. This tool will leverage NLP techniques to interact with users, process document-based knowledge, and convert it into explicit forms that can be easily accessed and shared within the organization. Through the integration of conversational interfaces and NLP algorithms, we envision a more dynamic and accessible approach to knowledge management that enhances organizational learning and decision-making processes.

References

1. Zaoui Seghroucheni. O, Al Achhab.M , Lazaar. M. (2023). Systematic Review on the Conversion of Tacit Knowledge.2023 7th IEEE Congress on Information Science and Technology (CiSt)
2. Farnese. M.L, Barbieri. B , Chirumbolo. A , Patriotta. G.(2019). Managing Knowledge in Organizations: A Nonaka's SECI Model Operationalization. FRONTIERS IN PSYCHOLOGY
3. Mohajan, Haradhan. (2016): Sharing of Tacit Knowledge in Organizations: A Review. Published in: American Journal of Computer Science and Engineering.
4. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. A., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All You Need. Advances in Neural Information Processing Systems (NeurIPS)
5. Jurafsky, D., & Martin, J. H. (2023). Speech and Language Processing (3rd ed.). Prentice Hall.
6. Goldberg, Y. (2017). Neural Network Methods for Natural Language Processing.
7. Rajvardhan Patil, Venkat Gudivada, (2020). Review A Review of Current Trends, Techniques, and Challenges in Large Language Models (LLMs). MDPI journals
8. Kanwar .M.A, Talha A.K, Syed.M.A, Asif.A. 4, Sharfuddin. A.K5, Sadique.A. (2024) An Exhaustive Comparative Study of Machine Learning Algorithms for Natural Language Processing Applications. 1st International Conference on Industrial, Manufacturing, and Process Engineering.
9. Gibney. D, Sharma V. Thankachan. (2021). Text Indexing for Regular Expression Matching. Algorithms journal.
10. Arthur Flor de Sousa Neto, Byron Leite Dantas Bezerra, Alejandro Héctor Toselli. (2020). Towards the Natural Language Processing as Spelling Correction for Offline Handwritten Text Recognition Systems. Applied Sciences Journal.
11. Khetam Al Sharou, Zhenhao Li, Lucia Specia. (2021). Towards a Better Understanding of Noise in Natural Language Processing. Proceedings of Recent Advances in Natural Language Processing
12. Kai Jiang, Xi Lu. (2020). Natural Language Processing and Its Applications in Machine Translation: A Diachronic Review. 2020 IEEE 3rd International Conference of Safe Production and Informatization.
13. Rebecca Dridan, Stephan Oepen. (2012). Tokenization: Returning to a Long Solved Problem A Survey, Contrastive Experiment, Recommendations, and Toolkit. Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics
14. Taku Kudo, John Richardson. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. EMNLP 2018
15. Divya Khyani, Siddhartha B S, Niveditha N M, Divya B M. (2020). An Interpretation of Lemmatization and Stemming in Natural Language Processing. Journal of University of Shanghai for Science and Technology.
16. Andrea Gasparetto, Matteo Marcuzzo, Alessandro Zangari, Andrea Albarelli. (2022). A Survey on Text Classification Algorithms: From Text to Predictions. Information Journal.
17. Nada Shahin, Leila Ismail. (2024). From Rule-Based Models to Deep Learning Transformers Architectures for Natural Language Processing and Sign Language Translation Systems: Survey, Taxonomy and Performance Evaluation. Artifcial Intelligence Review.
18. Gulizada Haisa, Gulila Altenbek. (2022). Deep Learning with Word Embedding Improves Kazakh Named-Entity Recognition. Information Journal.
19. Mohammad Mustafa Taye. (2023). Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions. Computers Journal.

 doi:10.20944/preprints202412.1714.v1

20. Jianguo Jia, Wen Liang, Youzhi Liang. (2024). A Review of Hybrid and Ensemble in Deep Learning for Natural Language Processing.
21. Helena Gomez-Adorno, Gemma Bel-Enguix, Gerardo Sierra, Juan-Carlos Barajas, William Álvarez. (2024). Machine Learning and Deep Learning Sentiment Analysis Models: Case Study on the SENT-COVID Corpus of Tweets in Mexican Spanish. Information Journal.
22. Divakar Yadav, Jalpa Desai, Arun Kumar Yadav. (2022). Automatic Text Summarization Methods: A Comprehensive Review.
23. Pinaki Sahu. (2023). Exploring Word Embedding Tools: GloVe, FastText, Word2Vec and BERT. AICyberInnovate Spectrum Magazine.
24. OMAR. M, CHOI. S, NYANG. D, MOHAISEN. D. (2022). Robust Natural Language Processing: Recent Advances, Challenges, and Future Directions. IEEE ACCESS.
25. Nils Reimers, Iryna Gurevych. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP 2019).