On Assigning Product and Software Codes to Customer Service Requests with Large Language Models

Sujatha Das Gollapalli,[1] **Mouad Hakam,**[1] **Mingzhe Du,**[1,2]
See-Kiong Ng,[1] **Mohammed Hamzeh**[3]
[1]Institute of Data Science, National University of Singapore
[2]College of Computing and Data Science, Nanyang Technological University
[3]Cisco Systems, Inc. Austin, TX, U.S.A.
{idssdg,mouad.hk,mingzhe,seekiong}@nus.edu.sg, mhamzeh@cisco.com

Abstract

In a technology company, quality of customer service that involves providing troubleshooting assistance and advice to customers is a crucial asset. Often, insights from historical customer service data are used to make decisions related to future product offerings. In this paper, we address the challenging problem of automatic assignment of product names and software version labels to customer **S**ervice **R**equests (SRs) related to Cisco Systems, Inc., a global company specializing in networking hardware, software, and services.[1] We study the effectiveness of state-of-the-art Large Language Models (LLMs) in assigning the correct product name codes and software versions from several possible label options and their "non-canonical" mentions in the associated SR data. We frame this assignment as a multiple-choice question (MCQ) answering task instead of conventional prompts and devise, to our knowledge, a novel pipeline of employing a classifier for filtering inputs to the LLM for saving usage costs. On our experimental dataset based on real SRs, we are able to correctly identify product name and software version labels when they are mentioned with over 90% accuracy while cutting LLM costs by about $40-60\%$ on average, thus providing a viable solution for practical deployment.

1 Introduction and Background

Companies such as *Arista Networks*, *Cisco Systems*, *Dell Technologies*, and *Huawei Technologies* offer a range of hardware devices and associated software related to routers, switches, modems, firewalls, and other networking purposes. For these companies, customer service which deals with providing technical assistance and troubleshooting support via emails, online chat, phone calls, etc. comprises a crucial aspect of business operations. Indeed, customer service quality directly impacts customer retention for a business and often insights based on customer interactions are used while making decisions regarding products (for example, edits to technical documentation, future product features, etc.) (Schijns, 2012; Brent Kitchens and Abbasi, 2018; Winkler and Wulf, 2019). However, scaling customer service is particularly challenging to these companies given the technical complexity in their rapidly growing and evolving product mix, high expectations of quality services from a growing customer base, and increasing use of digital touch-points. In this context, recent breakthrough research in the form of Large Language Models (LLMs) has particularly been of great interest to these companies due to the LLMs' advanced language understanding, generation, and reasoning capabilities that can significantly improve the automation of various tasks related to customer service (Wulf and Meierhofer, 2024; Krishnan et al., 2022; Cui et al., 2017; Su et al., 2025).

In this study, we investigate the use of LLMs for a specific task pertaining to customer support data related to *Cisco*, a company operating in the networking domain providing thousands of products and services to customers across the world. As new technologies emerge and applications evolve, *Cisco* receives increasing volumes of customer "**S**ervice **R**equests" (SRs) requesting assistance for setting up, troubleshooting, and for information via various channels such as emails, phone calls, online forums, or virtual chat. Service requests go through a lifecycle: (1) *opening* of an SR in response to a customer problem, (2) back and forth *communication* with the customer for any missing relevant problem details, as well as with the relevant internal product teams and knowledge experts, (3) *resolution* where the required support is provided by the tech engineer to the customer, and (4) *closure* where the SR is "marked" as closed after resolution. On average, **1-2 million** SRs are handled in *Cisco* every year, with resolution times ranging

[1]We use the name *Cisco* for future mentions in this paper.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 1092–1103
November 4-9, 2025

from a few weeks to several months depending on the complexity, priority, and severity of the SR.

Metadata
Technology: Cisco DNA Center - On-Prem ✓ **Subtechnology:** Cisco Software-Defined Access (SDA Wired) ✓ **Problem Code:** Error Messages, Logs, Debugs **Problem Details:** SDA FE C9300✓ Device is Experiencing TCAM Exhaustion 97.0%. Layer2 TCAM Usage Total Mac Addresses for this criterion: 583 Total Mac Addresses installed by LISP: REMOTE: 32768 *(device log information is redacted)*
Case Notes
Total Case Notes for this SR = 22 **[Case Note:Log File]** File uploaded from Support Case Manager File Name: IP address_12012023_2351.log **[Case Note: Email]** Country: Singapore Software Version: 16.12.4 ✓ **[Case Note: Email]** Moving to 9300 ✓ platform team
To Predict
Software Version → ***'16.12.04'*** Product Name: → ***'C9300-48U'***

Table 1: Example SR is shown with its metadata fields, fields to predict, and associated case notes. Elements useful for prediction are underlined and indicated (✓)

1.1 Motivation

During the closure of an SR, two key pieces of information ("data fields") are **manually** added to the closure report forms by tech support engineers. These fields capture the precise "product code" referencing the hardware on which the customer faced the problem as well as the "version" of the software being used on the hardware, where applicable. These assignments may be "NONE" if the problem is generic or related to third-party software/hardware. A chief motivation for collecting this information in *Cisco* pertains to knowledge completion for future reference, staff training, and product improvements, as well as valuable macro-level insights that can be derived from the associations to enable informed strategic decisions. For example, missing documentation for a particular configuration option that had led to several SRs may be added in the next version of the configuration guide for the product together with the information shared during resolution by the tech support engineers for these SRs. Similarly, commonly seen (problem, resolution) pairs can be used to create troubleshooting FAQs for customers as well as problem-solving chatbots (Zheng et al., 2023; Yang et al., 2023).

Given the large and complex hardware-software product mix offered by the company, adding product name codes and software information is necessary for enabling rapid retrieval of solutions when similar problems are observed in future SRs. The current manual assignment of product names and software version information in *Cisco* is prone to error since these fields may not be directly available as part of customer shared problem descriptions and instead may require gleaning from the many different types of structured and unstructured data created for the SR during its life cycle. The possible labels for these fields are currently provided for selection via dropdown lists in *Cisco*, but errors often creep in due to the sheer length of these dropdown options.[2] On the other hand, product and software information that can be extracted precisely and programmatically from specific device log files is observed in only about 10% of the cases, requiring manual labeling for the rest.

In Table 1, we show part of the data created during the resolution of a real SR for illustration. The SR data can be broadly categorized into–(1) **Metadata** comprising of structured fields (submitted via a form) of which the relevant ones for our assignment tasks are "problem description, customer symptoms, and resolution summary", and company-specific codes indicating the **product technology and sub-technology**. A product technology can pertain to a broad topic such as "routers" with sub-technology referencing "wireless" or "wired" configurations. (2) **Case Notes** are unstructured text documents shared during the lifecycle of an SR. For the SR shown in Table 1, we have a total of 22 associated case notes that are of different types such as emails exchanged within the company, with the customer, logs containing output from commands, and notes made by tech support engineers during the resolution process.

In Table 1, we highlight the relevant fields (with ✓) that can help determine the precise product name and software version information in the provided metadata and case notes. In *Cisco*, related alphanumeric codes are used to represent the prod-

[2]Due to decades of operation and the hundreds of variations of products being managed in the market for a given purpose (e.g. routers), numerous options figure for a product name and software version selection in the dropdown lists.

uct type as well as the precise product name under that type in SR documents. For instance, few product codes under the technology/sub-technology group "Wireless/2800 Series Access Point" include *AIR-AP2802I-S-K9*, *AIR-CT8540-1K-K9*, and *AIR-AP2802I-Z-K9*.

For the example SR shown in Table 1, the software information is directly mentioned in one of the emails (from case notes), whereas the complete product name has not been mentioned anywhere. Unlike this SR, software information typically appears in "non-canonical" formats– (*15.5(1)SY1*, *15.5(1)SY*, *15.5(1)SY3* all referring to *15.5.1*) and is sometimes mentioned in filenames exchanged during SR handling ('*cat9k_iosxe.16.12.04.SPA.bin*').

A manual assignment of product name and software version information is both time-consuming and error prone. Due to the myriad formats in which the product name and software version information is expressed in texts, along with other confounding patterns common in networking domains such as IP addresses, dates, and system logs messages, solutions based on pattern matching alone are inadequate. Moreover, texts such as "unavailable post upgrade from *2.2.2.9* to *2.3.3.7*" and "problem was first seen in *AIR-CT8540* (see SR 659356)" are common and need a deeper, semantic understanding for an accurate assignment.

Contributions: We present our LLM-based solution using a multiple-choice question (MCQ) answering formulation for predicting product names and software versions from the SR metadata and case notes in *Cisco* as an alternative to the currently adopted manual process. Our evaluation experiments with a range of open-source and proprietary LLMs show that with appropriate prompts and custom matching functions, we are able to predict product names with an accuracy of about 70-80% and software versions with over 96% accuracy in best settings. We address cost optimization for LLM calls by training a binary classifier to filter out the uninformative case notes available for an SR in order to send only relevant documents that can help with the prediction task to the LLM. Overall, in our best settings, we filter out uninformative tokens in LLM API calls with high precision (∼90%) and enable cost savings of ∼40-60%.

2 Methods

Our task is a combination of two NLP problems: named-entity recognition (NER) and document classification. For a parallel, consider the task of labeling financial documents with the full names of the financial organization that each document is about primarily. For example, consider a document that is primarily about the company "Tesco PLC" and its market competitors, the true label for the document would be "Tesco PLC". However, the document's content may not reference the true label precisely, but only via its abbreviation such as "Following Wal-Mart's model, both France 's Carrefour, and TSCO.L...". Our task is to predict the true classification label "Tesco PLC" for the document based on document understanding and the mentions of "TSCO.L". For *Cisco*, these target labels are various product names and software versions where mentions tend to be partial or ad hoc variations of the full names, and the documents are noisy and in heterogeneous formats such as unstructured notes, emails, metadata etc.

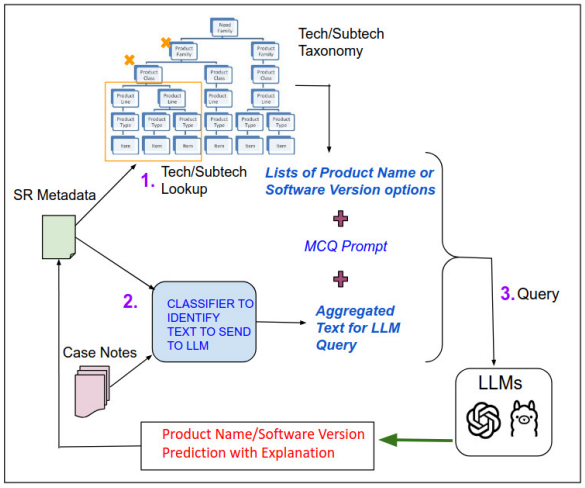

Figure 1: Workflow of our proposed approach.

Our overall pipeline for predicting product names and software version information in SRs is illustrated in Figure 1. In zero-shot settings, we leverage the language understanding capabilities of LLMs via a straightforward question answering prompt ("QPrompt") such as "What is the product name being discussed in the document below?" (Borst et al., 2022; Du and Cardie, 2020; Hu et al., 2025). Product names in *Cisco* are alphanumeric codes assigned through naming conventions internal to the company, unlikely to be "pretrained knowledge" for LLMs. Product name codes are indicative of organizational and grouping information. For example, *'AN5900-35X'* could be a specific product under the product group *'AN5900'* whereas *"AB5-JX-BRM"* and *"AB5-JX-BRM-XL"*

could be siblings under a common technology.

Despite indicative hints, consistent naming conventions across the several product verticals are lacking in *Cisco*. For instance, the code *AIR-CT8540-**1K**-K9* refers to the product "Cisco 8540 Wireless Controller supporting **1000** access points" whereas *SNS-3595* is a product code under the technology "Security - Access Control - AAA and Policy Management". A lack of a clear naming schema and product code name conventions makes it challenging to include this information into LLM prompts for extracting product names from SR data. In experiments, we find that providing sample product codes with the question prompt improves prediction performance significantly for product names although the overall performance is still low compared to the multiple-choice question formulation described next (Table 4).

We harness the technology/sub-technology organizational hierarchy by creating lists of all possible product names (or alternatively, software versions) using historical SR data, applying custom matching functions to canonicalize their mentions, and collating them to form options for multiple-choice question (MCQ) prompts. Intuitively, MCQ options provide hints to the LLM on the correct label choice for a document despite it not including a precise mention. For the previous example using organization names, an MCQ formulation "which of the following companies– Tesco PLC, Carrefour ..." may enable the LLM to reason out the potential connection between *Tesco PLC* and the mention *TSCO.L* to make a more accurate prediction. Our prompt templates are listed in Table 2.

Filter Classifiers: The case notes for an SR refer to "free text" notes created and exchanged during the process of resolving an SR and include different types such as emails between customer and tech support engineers, back and forth communications about scheduling meetings, action plans, etc. The numbers of case notes range from a few tens to hundreds, and though mostly uninformative, few notes contain vital exchanges related to product name or software version details of the SR. For illustration, the histogram of number of case notes for SRs as well as samples of uninformative case notes in our experimental dataset are shown in Figure 2 and Table 3, respectively, whereas sample useful case notes were highlighted in Table 1.

Given that commercially-available LLMs incur a cost depending on the number of input tokens per API call, it is not cost-efficient to send uninformative documents to the LLM. In addition, these documents may instead add "noise" and affect predictions by the LLM. To address these concerns, we train two classifiers (for product name and software version, respectively) to filter out the uninformative documents for a given SR before aggregating and sending the rest to the LLM. The training data for

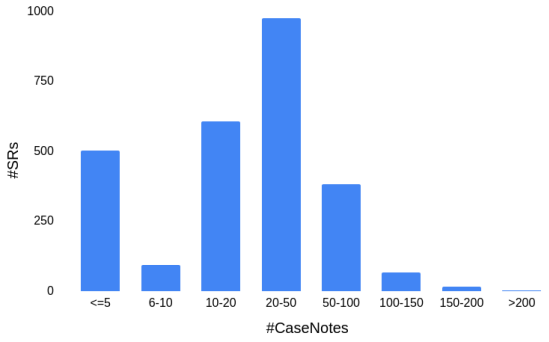

Figure 2: Histogram of number of SRs versus number of associated case notes in our dataset.

our filter classifier for product name/software version was created based on the following intuition– for a given SR, if none of the elements in the potential list of product name/software version labels are mentioned in a case note, it is unlikely to be useful for predicting product name/software version information (negative class), and conversely, if at least one of the potential labels appears in a case note, it may be useful for the prediction task (positive class). Due to the large number of uninformative case notes present in the SR data, the above process results in highly skewed datasets. We employ sampling on the negative class ("uninformative notes") and investigate smaller classification models for training filter classifiers.

Matching functions: Human-generated product name mentions in SR data do not always strictly follow naming conventions and occur as ad hoc variations (e.g., "AB5 JX BRM" or "Axx Byyy5 JX BRM" instead of "AB5-JX-BRM"). Similar to product names, software mentions often occur in non-canonical forms ("yyyyy.3.1.3(c).A.bin" or "3.1-3c" instead of "3.1.3c") in SR data. In consultation with domain experts, we adopted the longest common sequence as the matching function for product names.[3] Software strings are first normalized using various rules to extract the numeric sequence and alphabetic sequence **separately** and the

[3]LCS matches greater than 70% are considered equivalent. Domain experts considered product group prediction to be of value if the full mention is unavailable (Table 1).

Q Prompt	Which *product/software version* is principally discussed in these documents?
MCQ Prompt	Which *product/software version* from the given list: `{choice_list}` is principally discussed in these documents?
Extra Prompt	+ Answer "NONE" if you cannot find the answer in the given documents.
System Prompt	You are a helpful *Cisco* tech support engineer . . .

Table 2: List of LLM prompt templates. `{choice_list}` is derived from the relevant product tech/subtech hierarchy.

Case Note 1 Hello [Person Name], This email is to let you know that the severity of Case Number [CaseNumber] has been changed: Previous Severity: 4 New Severity: 3 Reason for change: . . .
Case Note 2 Next Plan of Action: For time being, if the RP connection is coming UP and there is a proper RP communication, Please try the following action 1. Need to bring down the RP Port 2. Make the reachability . . .
Case Note 3 Thank you for sharing the requested logs, I will review and update the findings . . .

Table 3: Examples of SR case notes, unlikely to help with our prediction tasks.

software version matching function incorporates an **exact** match for the numeric parts and edit distance for the non-numeric parts of the string. This combination ensures that "3.1.3" does not match "3.1.1" or "3.1.35" but will match "3.1.3a" if the edit distance of '1' for alphabetic characters is acceptable to domain experts (Kleinberg and Tardos, 2005). Matching functions play a crucial role in identifying ad hoc variations of product name and software mentions and are used not only to compile canonical options lists for MCQ prompts but also to identify documents with mentions ("positive" class) for training the filter classifiers, as well as during evaluation for comparing predictions and gold labels.

Recently, retrieval augmented generation (RAG) solutions that merge external knowledge (for example, retrieved from a database or web search) with LLMs' intrinsic knowledge are being widely adopted for domain knowledge-intensive tasks (Gao et al., 2024). Note that our core task is fundamentally extractive, framed as "What product name (or software version) is this SR addressing?". In a typical RAG workflow, this question would need to be transformed into an effective query to retrieve relevant documents. However, in our context, the necessary information is already expected to be within the input texts related to the SR, precluding the "retrieval" step. In addition, for matching against a pre-defined, structured list of product names or software versions (our MCQ options), we needs flexible matching functions for identifying relevant lexical and syntactic patterns making generic retrieval mechanisms (based on vector similarity) unsuitable. Our classifier-enabled augmentation approach based on matching functions customized to our domain is motivated by these challenges.

3 Experiments

Dataset: For studying this task, *Cisco* provided a dataset of about $1,287$ SR metadata records and their associated case notes (27,136 text documents). After employing our matching functions to identify canonical forms, we extracted 218 unique product names and 375 unique software versions for the provided data organized into a two-level hierarchy of 44 and $\sim$ 200 product technology, sub-technology categories, respectively. Correct product names and software versions were provided by *Cisco* only for a small subset of SRs that we use for evaluating product name and software version prediction. "Gold" product name and software version labels were available for 55 SRs and 30 SRs, respectively. Though these numbers are considerably small, for zero-shot settings and a relative evaluation of LLMs and prompts, we posit that our findings are still representative. The full set of $1,287$ SR case notes were used for creating data for product name (**PName**) and software version (**SWV**) filter classifiers, with subsets of "gold" SRs used to create the validation splits. After sampling (10%) for the majority ("negative") class, our resulting datasets are summarized as follows:

	Train (+/-)	*Test* (+/-)	*Val* (+/-)
PName	1859/5192	409/1305	56/61
SWV	463/6588	125/1589	20/234

LLM Prediction Performance: We experimented with a range of proprietary as well as open-source state-of-the-art LLMs including those from GPT-series (OpenAI), Claude (Anthropic), DeepSeek-R1 (DeepSeek), Llama (Meta), and the Qwen series for predicting product names and software versions from SR data. Precise model names can be found in Appendix A.1. We also included experiments with smaller models in Appendix A.4 to highlight their significantly lower performance compared to SOTA LLMs in zero-shot settings.

All result tables for product name/software version prediction in this paper show **averaged accuracy** values using the two matching functions described previously.

Prompt	**PName**	**SWV**
QPrompt	0.3090	0.8965
QPromptEgs+NONE	0.4181	0.7931
MCQPrompt	**0.7818**	0.5172
MCQ+NONE	0.7272	**0.9655**

Table 4: Prediction accuracy with GPT-4o

A summary of performance with GPT-4o LLM using different prompts is provided in Table 4. Since product name strings are not "general knowledge", not surprisingly, the basic QPrompt does not perform well for product names in contrast with software version labels. Specifying exemplars for indicating to the LLM the nature of product name strings ("QPromptEgs+NONE") significantly improves the product name performance but with a suitable MCQ prompt that includes option lists to choose from, we obtain significantly higher performance for both product names and software versions. The prediction performance using best prompts across LLM models (from Table 4) is summarized in Table 5. Among the LLMs considered as well as older models from OpenAI, GPT-4o performed the best for product names whereas several SOTA LLMs performed equally well for software version identification.

In our dataset, there are SRs for which product name (or software version) cannot be extracted based the available information alone. For instance, the SR pertains to a generic problem (agnostic to specific devices) or related to a third-party product

Model Name	**PName**	**SWV**
Claude-3-7-Sonnet	0.6909	**0.9655**
GPT-4o	**0.7818**	**0.9655**
QwQ-32B	0.7273	**0.9655**
Llama-70B	0.6000	0.8620
DeepSeek	0.6545	0.7931

Table 5: Prediction Performance across LLMs

or software. In these cases, the "correct" prediction should be "NONE" or "N/A" (empty). We summarize the aggregate performance (on NONE and non-NONE SRs) of best models across LLM groups on the two prediction tasks using the best-performing MCQ prompts in Tables 4 and 5. Overall, when the labels are available (non-NONE cases), in best settings, all latest LLM offerings (GPT-4o, Claude-3.7-Sonnet, QwQ32B) perform competitively (over 85% accurate) but their aggregate performance is affected by errors on the NONE cases. However, an explicit instruction regarding NONE handling affects performance differently for software and product name prediction. Details of these experiments, exact prompt texts, and other computational setup information are included in the appendix in view of space constraints (Table 7-11).

Filter Performances: We investigated models from RoBERTa (Liu et al., 2019), T5 (Raffel et al., 2020), Flan-T5 (Chung et al., 2024) (base and large variants) for training our filters. On this binary classification task, the FlanT5-large model fined-tuned using instructions including potential labels "Question: Given the list of labels, does the passage contain any of the labels? Answer True or False. Labels= []" performed the best. Performance using standard classification metrics (Bishop, 2007) on the test datasets with FlanT5-large is shown in Table 6 with comparison experiments listed in Table 16 of the Appendix.

Filter/Class	**Precision**	**Recall**	**F1**
PName (+)	0.65	0.63	0.64
(-)	0.89	0.89	0.89
MacroAvg	0.77	0.76	0.76
SWV (+)	0.80	0.86	0.83
(-)	0.99	0.98	0.99
MacroAvg	0.90	0.92	0.91

Table 6: Performance of filters on Test Set

The performance of the software version filter is significantly better compared to that of the product name filter. Similar to the earlier task, we attribute this lower performance to the nature of product name sequences ("alphanumeric" codes) which are challenging to distinguish from other textual tokens in SR data contexts. As such, in product name and software version filters, we are able to identify uninformative case notes in the test dataset with high precision–89% and 99%, respectively.

The results in Tables 4 and 5 use the case notes identified as informative ("True/+") by our filters. On average, we are able to filter out upto 40% of total tokens while predicting software versions (for 30 SRs) and upto 60% of the total tokens while predicting product names (for 55 SRs). Though the specific percentage savings may be an artifact of our datasets (since SRs have varying numbers of case notes as shown in Figure 2), these results highlight the effectiveness of using filters to obtain considerable savings in *Cisco* where millions of SRs are handled annually (Howell et al., 2023).

Deployment notes: Our LLM solution is expected to significantly decrease the earlier manual effort during SR closure in *Cisco*. A GPT-4o based solution is now being test-bedded for evaluating time and cost savings for manpower and LLM usage in realistic conditions. We note that though large-scale, open-source LLMs perform comparably on our evaluation datasets as seen in experiments, current commercial products have the advantage of ease of use via APIs, and coupled with smaller filter models eliminate the need for in-house compute-intensive environments.

4 Related Work

Apart from chatbots, LLMs are being employed in various tasks related to customer support data such as automated text correction, summarization and question answering (Wulf and Meierhofer, 2024; Palen-Michel et al., 2024), product reviews evaluation (Roumeliotis et al., 2024; Azov et al., 2024), as well as knowledge and workflow representation (Su et al., 2025). Recently LLMs were used for NER in datasets from customer order descriptions (Oh, 2024), recipes (Agarwal et al., 2024), clinical contexts (Lu et al., 2024), and the financial news articles (Shah et al., 2024). Though question answering techniques for extracting metadata were previously investigated (Du and Cardie, 2020; Borst et al., 2022; Song et al., 2024; Hu et al., 2025), as far as we know, there is no multiple-choice question answering formulation for similar tasks in published works. Since our dataset is proprietary, we also showcase our approach's advantage over conventional prompts on a recent public dataset for the comparable task in the financial domain (Appendix A.3).

5 Conclusions

We investigated the use of SOTA LLMs for assigning product names and software version labels for customer service requests (SR) data, previously requiring human intervention. With appropriate prompts, existing commercial LLMs are able to extract the required information from SR data with high accuracy. However, a naïve use results in high LLM costs due to substantial amounts of irrelevant data associated with SRs. To this end, we designed filter classifiers that ensure that only SR data pertinent to prediction is included in the LLM prompts. In on-going work, we are exploring practical deployment of our developed pipeline within *Cisco*.

Limitations

Though our solution design was motivated by a context exclusive to *Cisco*, we expect the general result trends and ideas such as MCQ formulation and input filtering for LLMs to carry across comparable datasets as illustrated by our results on the NER task from the financial domain (Appendix A.3). Filter classifiers, despite being not very high performing (particularly for product name), did not adversely affect the final prediction results on our SR datasets. Since errors on "positive" class result in informative case notes being mistakenly discarded, there is a need to improve datasets both for training the underlying classifiers as well as for evaluating the effect of incorrect classification on final predictions. We highlighted the contradictory behavior of LLMs regarding handling "NONE" cases for software version versus product name assignments requiring further investigation regarding prompt sensitivity of LLMs (Bowman, 2023). Studies on MCQ-answering behavior of LLMs are on-going (Huang et al., 2022; Balepur et al., 2024; Wang et al., 2025).

Our industry-partner *Cisco* keen on exploring state-of-the-art LLMs for their in-house tasks, provided us fully-anonymized datasets replacing all personally identifiable information such as customer names and contact details with pseudo-

identifiers. Generally speaking however, data privacy is a crucial concern while using proprietary LLMs on data containing sensitive customer information. Another caveat of our proposed solution is the assumption of a "static ontology" as we use the pre-existing product and software technology/sub-technology information to construct options for our MCQ prompts. While including a "NONE" option among the choices aims to capture the missing instances and highlight when the ontology updates are due, future research is needed for developing methods that can seamlessly handle dynamic ontology updates.

Acknowledgments

This research is supported by A*STAR, CISCO Systems (USA) Pte. Ltd and National University of Singapore under its Cisco-NUS Accelerated Digital Economy Corporate Laboratory (Award I21001E0002).

References

Ayush Agarwal, Janak Kapuriya, Shubham Agrawal, Akhil Vamshi Konam, Mansi Goel, Rishabh Gupta, Shrey Rastogi, Niharika Niharika, and Ganesh Bagler. 2024. Deep learning based named entity recognition models for recipes. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 4542–4554, Torino, Italia. ELRA and ICCL.

Guy Azov, Tatiana Pelc, Adi Fledel Alon, and Gila Kamhi. 2024. Self-improving customer review response generation based on LLMs. In *Proceedings of the Seventh Workshop on e-Commerce and NLP @ LREC-COLING 2024*, pages 40–57, Torino, Italia. ELRA and ICCL.

Nishant Balepur, Abhilasha Ravichander, and Rachel Rudinger. 2024. Artifacts or abduction: How do LLMs answer multiple-choice questions without the question? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10308–10330, Bangkok, Thailand. Association for Computational Linguistics.

Christopher M. Bishop. 2007. *Pattern Recognition and Machine Learning (Information Science and Statistics)*, 1 edition. Springer.

Timo Borst, Jonas Mielck, Matthias Nannt, and Wolfgang Riese. 2022. Extracting funder information from scientific papers - experiences with question answering. In *Linking Theory and Practice of Digital Libraries*, pages 289–296, Cham. Springer International Publishing.

Samuel R. Bowman. 2023. Eight things to know about large language models. *Preprint*, arXiv:2304.00612.

Jingjing Li Brent Kitchens, David Dobolyi and Ahmed Abbasi. 2018. Advanced customer analytics: Strategic value through integration of relationship-oriented big data. *Journal of Management Information Systems*, 35(2):540–574.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Lei Cui, Shaohan Huang, Furu Wei, Chuanqi Tan, Chaoqun Duan, and Ming Zhou. 2017. SuperAgent: A customer service chatbot for E-commerce websites. In *Proceedings of ACL 2017, System Demonstrations*, pages 97–102, Vancouver, Canada. Association for Computational Linguistics.

Xinya Du and Claire Cardie. 2020. Event extraction by answering (almost) natural questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 671–683, Online. Association for Computational Linguistics.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

Kristen Howell, Gwen Christian, Pavel Fomitchov, Gitit Kehat, Julianne Marzulla, Leanne Rolston, Jadin Tredup, Ilana Zimmerman, Ethan Selfridge, and Joseph Bradley. 2023. The economic trade-offs of large language models: A case study. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 248–267, Toronto, Canada. Association for Computational Linguistics.

Zhilei Hu, Zixuan Li, Xiaolong Jin, Long Bai, Jiafeng Guo, and Xueqi Cheng. 2025. Large language model-based event relation extraction with rationales. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 7484–7496, Abu Dhabi, UAE. Association for Computational Linguistics.

Zixian Huang, Ao Wu, Jiaying Zhou, Yu Gu, Yue Zhao, and Gong Cheng. 2022. Clues before answers: Generation-enhanced multiple-choice QA. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Jon Kleinberg and Eva Tardos. 2005. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., USA.

Chitra Krishnan, Aditya Gupta, Astha Gupta, and Gurinder Singh. 2022. *Impact of Artificial Intelligence-Based Chatbots on Customer Engagement and Business Growth*, pages 195–210. Springer International Publishing, Cham.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Qiuhao Lu, Rui Li, Andrew Wen, Jinlian Wang, Liwei Wang, and Hongfang Liu. 2024. Large language models struggle in token-level clinical named entity recognition. *arXiv preprint arXiv:2407.00731*.

Jangmin Oh. 2024. Developing a model for extracting actual product names from order item descriptions using generative language models. *IEEE Access*, 12:122695–122701.

Chester Palen-Michel, Ruixiang Wang, Yipeng Zhang, David Yu, Canran Xu, and Zhe Wu. 2024. Investigating llm applications in e-commerce. *Preprint*, arXiv:2408.12779.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. 21(1).

Konstantinos I. Roumeliotis, Nikolaos D. Tselikas, and Dimitrios K. Nasiopoulos. 2024. Llms in e-commerce: A comparative analysis of gpt and llama models in product review evaluation. *Natural Language Processing Journal*, 6:100056.

Jos Schijns. 2012. Gaining customer insights by analyzing and understanding cross-platform and multichannel customer behavior. *International Journal of Business Research*, 12:131.

Agam Shah, Abhinav Gullapalli, Ruchit Vithani, Michael Galarnyk, and Sudheer Chava. 2024. Finerord: Financial named entity recognition open research dataset. *Preprint*, arXiv:2302.11157.

Hyunju Song, Steven Bethard, and Andrea Thomer. 2024. Metadata enhancement using large language models. In *Proceedings of the Fourth Workshop on Scholarly Document Processing (SDP 2024)*, pages 145–154, Bangkok, Thailand. Association for Computational Linguistics.

Hanchen Su, Wei Luo, Yashar Mehdad, Wei Han, Elaine Liu, Wayne Zhang, Mia Zhao, and Joy Zhang. 2025. LLM-friendly knowledge representation for customer support. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pages 496–504, Abu Dhabi, UAE. Association for Computational Linguistics.

Haochun Wang, Sendong Zhao, Zewen Qiang, Nuwa Xi, Bing Qin, and Ting Liu. 2025. LLMs may perform MCQA by selecting the least incorrect option. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5852–5862, Abu Dhabi, UAE. Association for Computational Linguistics.

Till J. Winkler and Jochen Wulf. 2019. Effectiveness of it service management capability: Value co-creation and value facilitation mechanisms. *Journal of Management Information Systems*, 36(2):639–675.

Jochen Wulf and Jürg Meierhofer. 2024. Utilizing large language models for automating technical customer support. *Preprint*, arXiv:2406.01407.

Changlin Yang, Siye Liu, Sen Hu, Wangshu Zhang, Teng Xu, and Jing Zheng. 2023. Improving knowledge production efficiency with question answering on conversation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 225–234, Toronto, Canada. Association for Computational Linguistics.

Xin Zheng, Tianyu Liu, Haoran Meng, Xu Wang, Yufan Jiang, Mengliang Rao, Binghuai Lin, Yunbo Cao, and Zhifang Sui. 2023. DialogQAE: N-to-n question answer pair extraction from customer service chatlog. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6540–6558, Singapore. Association for Computational Linguistics.

A Appendix

A.1 Model List

Table 13 presents the list of LLMs used in our system. The models include both proprietary and open-source options, with parameter sizes ranging from 32B to 70B where available. For each model, we provide its name, type, parameter size, and a reference link for access or documentation.

A.2 Details of Product Name and Software Version Prediction

Model training and inference with filter classifiers were performed on a single GPU of an Nvidia Tesla cluster (Linux) machine with 32GB RAM. For experiments using open-source LLMs (from Meta, Qwen, and DeepSeek), we used vLLM for inference on 4×H100 80GB GPUs.

In the following tables, we listed the performance on the set of SRs for which the "gold" labels are NONE in a separate column (NoneSet) to highlight the difference in how different LLMs handle these cases based on an explicit instruction regarding "NONE" handling (See Extra row in Table 2). Overall, about 30% and 50% of the cases are NONE for the product name and software gold SRs, respectively.

Model	non-NoneSet	NoneSet
Llama-8B	0.5777	0.1000
Llama-70B	0.7111	0.1000
DeepSeek	0.4000	**0.7333**
QwQ-32B	0.8888	0.0000
GPT-4o	**0.9111**	0.2000
Claude-3-7-sonnet	0.8444	0.0000

Table 7: Product Name Performance across LLMs with MCQ Prompt

Model	non-NoneSet	NoneSet
Llama-8B	0.4667	0.3000
Llama-70B	0.6818	0.1818
DeepSeek	**0.7500**	0.2727
QwQ-32B	0.7209	0.5000
GPT-4o	0.7209	**0.7500**
Claude-3-7-sonnet	0.7209	0.4167

Table 8: Product Name Performance across LLMs with MCQ-NONE Prompt

Model	non-NoneSet	NoneSet
GPT-4o-mini	0.8888	0.1000
GPT-o1	0.8666	0.0000
GPT-o1-preview	0.8666	0.0000
GPT-4o	**0.9111**	**0.2000**

Table 9: Product Name Performance across GPT Models with MCQ Prompt

Model	non-NoneSet	NoneSet
Llama-70B	**0.9285**	0.8000
Llama-8B	0.7142	0.1333
DeepSeek	0.7857	0.8000
QwQ-32B	**0.9285**	1.0000
Claude-3-7-sonnet	**0.9285**	**1.0000**
GPT-4o	**0.9285**	**1.0000**

Table 10: SWV Performance (with MCQ+NONE) across LLMs

Model	non-NoneSet	NoneSet
Llama-70B	0.9285	0.0000
Llama-8B	0.7857	0.0000
DeepSeek	0.1429	0.7333
QwQ-32B	0.9285	0.3333
Claude-3-7-sonnet	0.9285	**1.0000**
GPT-4o	**1.0000**	0.0667

Table 11: SWV Performance (with MCQ) across LLMs

Prompt	non-NoneSet	None-Set
Product Name		
noMCQ	0.2619	0.4615
noMCQwEg+NONE	0.1621	**0.9444**
MCQ	**0.9111**	0.2000
MCQ+NONE	0.7209	0.7500
Software Version		
noMCQ	0.8571	0.9333
noMCQwEg+NONE	0.5714	**1.0000**
MCQ	**1.0000**	0.0667
MCQ+NONE	0.9285	**1.0000**

Table 12: Experiments with prompts using GPT-4o

Model Name	Type	Size	Link
Claude-Sonnet	Proprietary	Unknown	https://www.anthropic.com/claude/sonnet
GPT-4o	Proprietary	Unknown	https://openai.com/api/
QwQ-32B	Open Source	32B	https://huggingface.co/Qwen/QwQ-32B
Llama-70B	Open Source	70B	https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct
DeepSeek-R1	Open Source	70B	https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Llama-70B

Table 13: List of selected LLMs used in our system, including their type (proprietary or open source), parameter size (if available), and source links.

System Prompt
You are a helpful *Cisco* assistant designed to output JSON.
Prompt
Which software version is principally discussed in these documents? Question: {question} The ONLY software version should be selected from the given software version list: [list of options] Response in this JSON format: {"software_version": "", "explanation": "", "summary": ""} *[DOCUMENT CONTENT]* "
Question
What product is principally discussed in these documents? Product names are a mix of alphanumeric characters and look like: ABC-PN1805-X, D3500-WAS6K, CM3-IU

Table 14: MCQ prompt is shown for software version assignment and QPrompt with exemplars is shown for product name assignment. Experiments with slight word variations of the prompt texts did not result in significant performance changes on the prediction task.

Type	Class	Precision	Recall	F1
PName	False (-)	0.70	0.87	0.77
	True (+)	0.80	0.59	0.68
SWV	False (-)	1.00	0.99	1.00
	True (+)	0.91	1.00	0.95

Table 15: Classification performance of filters on the validation set with FlanT5-large model

Model	Precision	Recall	F1
RoBERTa-base	0.89	0.82	0.85
RoBERTa-large	0.80	0.95	0.86
T5-base	**0.97**	0.92	0.94
T5-large	0.89	0.94	0.91
Flan-T5-base	0.88	0.99	0.93
Flan-T5-large	0.95	**1.00**	**0.97**

Table 16: Validation performance with various models for software version filter, Macro-averages on both classes are shown

A.3 FiNER-ORD experiments

To illustrate the benefit of the MCQ formulation over the general question formulation, we summarize experiments on FiNER-ORD, an open-research dataset from the financial domain (Shah et al., 2024) that includes English financial news articles where sentences were manually-annotated for person names, locations, and organizations. SOTA LLMs such as GPT-4o in zero-shot settings only achieve an accuracy of about 60% for organization entity mentions in this dataset.

To be comparable to our problem setting, we consider the sentences in the dataset that only include the "abbreviations" of organizations but whose full names are known at document level. For example, "Associated Press" is known early on based on document level information but the sentence only mentions "AP". The validation and test datasets for our experiments comprised of 84 and 256 sentences, respectively. We instruct GPT-4o LLM using the two question prompts "What organizations are being discussed in the following text?" versus "Which of the following organizations ..." with the fullnames of potential organizations provided as MCQ options in the prompt. With the simple question prompt, the accuracies range from 61-63% (similar to published numbers) whereas with the MCQ prompt, the performances were significantly higher and in the range 80-83% for the two sentence subsets. Our subsets of the original datasets and prompts are available upon request.

A.4 Experiments with smaller models

We show performance of smaller language models Flan-t5-large from Google[4] and Llama-3.1-8B-Instruct from Meta[5] in zero-shot settings on the metadata from a subset of 30 SRs in Table 17 for product name and software version prediction. As can be seen in this table (when compared

[4]https://huggingface.co/google/flan-t5-large

[5]https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

to LLMs/GPT-4o from OpenAI), performance of smaller models is significantly lower. In addition to context length limitations, we observe that smaller models are unable to generate output in consistent formats or follow instructions to provide explanations for a given answer choice. Consequently, best results were obtained with simpler prompts of the form "Question: Which of the product name labels in <list-of-labels> best match the following text? Context: []". Due to unavailability of large-scale labeled data, we did not investigate fine-tuning of smaller models in this study.

Model	PName	SWV
FlanT5-large	0.2666	0.3448
Llama-3.1-8B-Instruct	0.5000	0.3448
GPT-4o	**0.7333**	**0.5172**

Table 17: Aggregate accuracies for product name and software version prediction using smaller models are shown against GPT-4o performance

For the results in Table 17, we manually examined the output from Llama-3.1-8B-Instruct models since we were unable to adjust the prompt despite multiple reformulations to ensure a consistent output format. For the software version prediction experiments, similar to experiments with LLMs we added "NONE" handling with MCQ prompts (Table 4).