Robust Knowledge Extraction from Large Language Models using Social Choice Theory

Nico Potyka*
Cardiff University, UK
potykan@cardiff.ac.uk

Yuqicheng Zhu*
Bosch Center for AI, Germany
Univ. of Stuttgart, Germany
Yuqicheng.Zhu@de.bosch.com

Yunjie He
Bosch Center for AI, Germany
Univ. of Stuttgart, Germany
Yunjie.He2@de.bosch.com

Evgeny Kharlamov
Bosch Center for AI, Germany
Univ. of Oslo, Norway
Evgeny.Kharlamov@de.bosch.com

Steffen Staab
Univ. of Stuttgart, Germany
Univ. of Southampton, UK
steffen.staab@ki.uni-stuttgart.de

ABSTRACT

Large-language models (LLMs) can support a wide range of applications like conversational agents, creative writing or general query answering. However, they are ill-suited for query answering in high-stake domains like medicine because they are typically not robust - even the same query can result in different answers when prompted multiple times. In order to improve the robustness of LLM queries, we propose using ranking queries repeatedly and to aggregate the queries using methods from social choice theory. We study ranking queries in diagnostic settings like medical and fault diagnosis and discuss how the Partial Borda Choice function from the literature can be applied to merge multiple query results. We discuss some additional interesting properties in our setting and evaluate the robustness of our approach empirically.

KEYWORDS

Large Language Models; Robustness; Social Choice Theory

1 INTRODUCTION

Large Language Models (LLMs) achieve state-of-the-art results in various natural language processing (NLP) tasks. Formally, LLMs represent a conditional probability distribution $P(T_{n+1}|T_1, \ldots, T_n)$ over tokens (character sequences) that predicts the next token given a fixed context of previous tokens. To answer a query Q, Q is decomposed into tokens $T_1, \ldots, T_Q$ and used to sample the first token A_1 of the answer from $P(A_1|T_1, \ldots, T_Q)$. A_1 can then be added to the context $T_1, \ldots, T_Q$ to sample the next answer token A_2. This process is repeated until a special end of text token is reached. Since

---

*Equal Contribution.

finding an optimal sequence of answer tokens is hard, answer sequences are often computed by a heuristic search like Beam [8], Top-K [4] or Nucleus [7] search that build up multiple promising token sequences in parallel. The sampling process is controlled by a temperature parameter. For temperature 0, the algorithms samples greedily. Increasing the temperature allows sampling tokens with lower local probability. While some authors associate higher temperatures with more creative answers, they can also result in higher probability answers because greedy selection can exclude high probability sequences that start with low probability tokens.

Given the success of LLMs in difficult NLP taks, they are increasingly being used for general question answering tasks. This is a natural application as it is reasonable to assume that LLMs picked up a lot of interesting information during training. However, one limitation of LLMs is that they will always produce an answer even if they did not learn anything about the question. This problem is referred to as *hallucination* in the literature [10]. The uncertainty of an answer is hard to quantify. While every answer sequence can be associated with a probability, this is merely the probability of the text sequence and should not be confused with the probability that the answer is correct (or that the LLM "believes" that the answer is correct). Theoretically, LLMs can be asked to output probabilities for their predictions, but it is hard to say how meaningful these probabilities are since there is nothing in a typical LLM architecture that would allow them to infer meaningful probabilities (unless they picked up a particular probability from the training corpora). Since query answering with LLMs is based on a heuristic search for high probability token sequences rather than on reasoning, we, in particular, have the following types of uncertainty:

(1) Query-Uncertainty: prompting the same query repeatedly can result in different answers.
(2) Syntax-Uncertainty: semantically equivalent queries that differ only syntactically can result in different answers.
(3) Distraction-Uncertainty: meaningless information added to the query can result in a different answer.

Let us note that, in principle, query uncertainty can be eliminated by setting the temperature parameter to 0. However, as outlined above, the deterministic answer will be somewhat random because it corresponds to some local optimum found by a heuristic search algorithm. We therefore aim at allowing some randomness in the answer, but increasing the robustness. The idea of robustness is that similar queries should result in similar answers. In particular,

the same query prompted multiple times can result in different answers in our setting. However, we would like that these answers are semantically similar. Similarly, we would like that syntactic changes of a query do not change the semantics of the answer.

In this work, we will mainly focus on making LLMs robust against *query uncertainty*, but we will also look at *syntax uncertainty* in our experiments. We explore to which extent an answer sampling strategy combined with social choice theory methods can improve the robustness of LLMs. The idea is as follows: instead of asking a query once, we ask it repeatedly (starting each time from the original question context). Our assumption is that if the LLM picked up the answer during training, then this answer should occur in the majority of cases. On the other hand, if it did not pick up information about the query and hallucinates an answer, we expect that the different answers will be very random. We will apply tools from social choice theory to aggregate the answers. We expect that, if the LLM picked up meaningful information, then our aggregation will result in a clear ranking of the different answers, while it will be mostly indifferent between the answers otherwise.

Let us emphasize that the outcome should be interpreted with care. If the LLM has been trained on a text corpora with false information, we may find that an LLM gives a false answer with high certainty. The probabilities that we derive should therefore be understood as subjective probabilities that reflect the uncertainty of the LLM and not as statistical probabilities. We view our method as most useful when being applied to LLMs that were trained on reliable literature (e.g., peer-reviewed articles and books) and not on random text from the internet. While reliable pretrained models like BioBert and MedBert exist [15, 21], they still require fine-tuning to be usable as question answering systems. Since our resources are limited, we will therefore use ChatGPT-turbo in our experiments, which was trained on mixed data with varying reliability. However, our experiments are only a proof of concept and the idea can directly be transferred to LLMs trained on high quality domain-specific data.

In our investigation, we will focus on diagnostic problems, where we try to identify the cause of a particular situation or condition. The identified cause is called the *diagnosis* for the condition. Typical examples are medical diagnosis (identify the medical condition that causes a set of symptoms) or fault diagnosis (identify the defective component in a technical system that causes malfunctions). The query consists of a description of the situation and we ask for a ranking of possible causes ordered by their plausibility. In order to take account of uncertainty, we repeat the query multiple times and collect the rankings. Tools from social choice theory can then be applied to merge the rankings and to quantify the uncertainty of the answer. To do so, we will build up on scoring-based voting methods for partial preference orderings [3].

2 RELATED WORK

Prior research on uncertainty quantification of LLMs focused on investigating the probabilities of token sequences [6, 11]. However, as discussed before, the probability of the token sequence should not be confused with the probability that the token sequence expresses a valid claim. In particular, the same claim can be expressed by different (semantically equivalent) token sequences that obtain different probabilities. [14] address this issue by first clustering claims

with the same semantic meaning and summing their probabilities to calculate a "semantic entropy". Other work involves training or fine-tuning the LLMs to quantify uncertainty [12, 17, 18]. However, due to lack of transparent training specifics, these approaches might be difficult to reproduce in addition to being expensive.

Despite the demand for uncertainty quantification without relying on model fine-tuning or accessing the proprietary information of LLMs, there is little work in this area and much remains unexplored. To our best knowledge, only [24, 25] quantify uncertainty based on the verbalized confidence given by LLMs or self-consistency of the claims. The significance of verbalized confidence is unclear since there is nothing in a typical LLM architecture that would allow it to infer meaningful probabilities. Our approach aggregates answers and quantifies the uncertainty using methods from social choice theory. Moreover, We study queries that give a rank with multiple possible answers rather than one single answer as an output, no approach from existing work can be directly applied in our case.

The recent neuro-symbolic theorem prover *LINC* [19] uses LLMs as a semantic parser to translate natural language reasoning problems into first-order logic that can then be processed by a symbolic theorem prover. To decrease the risk of parsing errors, the authors parse and process the inputs repeatedly and apply majority voting to determine the outcome. This may be another interesting domain for applications of more sophisticated voting methods.

The notion of robustness that we consider here (similar inputs should result in similar outputs) follows the terminology in *Explainable AI* [1, 5, 16] and should not be confused with statistical [9] or adversarial [23] robustness. From an explanation point of view, our scoring method is interpretable in the sense that the scores can be explained from the LLM's responses to the repeated prompts. The responses can be further explained by the LLM's sampling procedure and the output probabilities of the transformer. However, understanding the output probabilities of transformers is difficult and a topic of current research [27].

3 SOCIAL CHOICE THEORY BACKGROUND

Social choice theory deals with aggregating individual preferences of different agents towards a collective choice [2]. The agents are often seen as voters who can express their preferences in different ways. For example, they may be able to vote for a single candidate, for multiple candidates or report a preference ordering over the candidates. We will focus on the latter setting here. Formally, we consider a finite set of *voters* $N = \{1, \ldots, n\}$ and a finite set of outcomes $O = \{o_1, \ldots, o_m\}$. A *partial order* $\succeq$ over O is a binary relation over O that is reflexive, anti-symmetric and transitive. We do not assume that it is complete, that is, there can be outcomes $o_i \neq o_j$ such that neither $o_i \succeq o_j$ nor $o_j \succeq o_i$. As usual, we write

- $o > o'$ iff $o \succeq o'$ and $o' \not\succeq o$,
- $o \sim o'$ iff $o \succeq o'$ and $o' \succeq o$.

If $o > o'$, we say that o is *strictly preferred* to o' and if $o \sim o'$, we say that we are *indifferent* between the two. A *profile* $p = [\succeq_1, \ldots, \succeq_n]$ contains one partial order for every voter and captures the preferences expressed by them.

The process of aggregating the voters' preferences can be formalized in different ways. A *social choice function* is a mapping f

from the set of all profiles to a non-empty subset of the outcomes. Intuitively, $f(p)$ should contain the outcomes that are maximally preferred by the voters. Ideally, $f(p)$ contains only a single element, but there are cases where a unique choice cannot be made without any ad-hoc assumptions (like chosing a random outcome or a lexicographically minimal one).

Social choice research often focuses on total orderings, where agents express preferences over all possible outcomes [2]. In our application, the outcomes are possible diagnoses, and the different answers do not necessarily contain the same diagnoses. We will therefore focus on preferences expressed by partial orderings. Since we are interested in quantifying the uncertainty of an answer (based on the variance in the rankings), scoring-based voting methods are a natural choice. We recall some ideas about aggregating partial preferences by scoring-based voting methods from [3].

To begin with, a *scoring procedure* $s_p : O \to \mathbb{R}$ is a mapping from outcomes to numerical values that is parametrized by a profile p [3]. Intuitively, $s_p(o)$ is the score of outcome o with respect to the preferences expressed by the profile p. Every scoring procedure induces a social choice function by letting

$$f(p) = \arg\max_{o \in O} s_p(o). \tag{1}$$

A *weighting procedure* $w_\geq : O \to \mathbb{R}$ maps outcomes to numerical values and is parametrized by a partial order $\geq$ [3]. Intuitively, $w_\geq(o)$ is the score of outcome o with respect to the preferences expressed by $\geq$. We can construct a scoring procedure from a weighting procedure by letting [3]

$$s_p(o) = \sum_{i=1}^{n} w_{\geq_i}(o), \tag{2}$$

where we assume $p = [\geq_1, \ldots, \geq_n]$.

A weighting procedure, in turn, can be based on how many other outcomes are less preferred and how many are incomparable. To do so, we can consider functions $\text{Down}_\geq : O \to 2^O$ and $\text{Inc}_\geq : O \to 2^O$ defined as follows [3]:

$$\text{Down}_\geq(o) = |\{o' \in O \mid o > o'\}|, \tag{3}$$

$$\text{Inc}_\geq(o) = |\{o' \in O \mid o \text{ and } o' \text{ are incomparable}\}|, \tag{4}$$

where, for a set S, $|S|$ denotes its cardinality. That is, $\text{Down}_\geq(o)$ is the number of outcomes ranked lower than o and $\text{Inc}_\geq(o)$ is number of outcomes incomparable to o.

The following two properties of weighting procedures have been proposed in [3]:

Linearity: There exist constants $\alpha, \beta, \gamma \in \mathbb{R}$ such that

$$w_\geq(o) = \alpha \cdot \text{Down}_\geq(o) + \beta \cdot \text{Inc}_\geq(o) + \gamma. \tag{5}$$

Constant Total Weight: There exists a constant δ such that $\sum_{o \in O} w_\geq(o) = \delta$ for all partial orders.

Partial Borda Weighting (PBW) [3] $w_\geq^{PBW}$ is the linear weighting procedure defined by letting

$$\alpha = 2, \beta = 1, \gamma = 0 \tag{6}$$

in (5).

Definition 3.1 (PBW Weighting). The *PBW weighting* procedure is defined as

$$w_\geq^{PBW}(o) = 2 \cdot \text{Down}_\geq(o) + \text{Inc}_\geq(o). \tag{7}$$

One can show the following.

THEOREM 3.2 ([3]). *$w_\geq^{PBW}$ satisfies Linearity and Constant Total Weight and every other weighting procedure that satisfies these two properties is an affine transformation of PBW.*

We refer to [3, Theorem 1] for more details about this result.

The *partial Borda choice function* f^{PBW} is the social choice function induced by $w_\geq^{PBW}$ based on equations (1) and (2). It can be characterized as follows.

THEOREM 3.3 ([3]). *The partial Borda choice function is the unique social choice function that satisfies the following properties.*

Consistency: If p_1, p_2 are disjoint profiles and $f(p_1) \cap f(p_2) \neq \emptyset$ then $f(p_1) \cap f(p_2) = f(p_1 \cup p_2)$.

Faithfulness: If $p = [\geq_1]$ and $b \geq_1 a$, then $a \notin f(p)$.

Neutrality: f is invariant with respect to permutations of O (renaming the outcomes will not affect the result), that is, $f(\sigma(p)) = \sigma(f(p))$ for all bijective mappings $\sigma : O \to O$.

Cancellation: If for all outcomes $o_1 \neq o_2$, the number of voters who rank o_1 above o_2 equals the number of voters who rank o_2 above o_1, then $f(p) = O$.

We refer to [3, Theorem 2] for more details about this result.

4 IMPROVING THE ROBUSTNESS OF LLM QUERIES WITH PBW

As we saw in the previous section, aggregating partial preferences with PBW gives us several desirable analytical guarantees. We will now use PBW to improve the robustness of LLM ranking queries. The basic idea is to ask the LLM for the most plausible explanations of a situation repeatedly and to use PBW to aggregate the answers.

4.1 From Queries to Rankings

In order to obtain ranking answers from LLMs, we consider queries of a special form that we call *ranking queries*. We refrain from a formal definition and just explain the intuitive idea. Roughly speaking, a ranking query consists of

- a *condition description*,
- *answer instructions*.

Example 4.1. As a running example, we will use a medical scenario with the following ranking query:

> "A 20 year old professional runner suffers from a stinging pain in the forefoot. The foot is swollen and stiff. What are the most plausible explanations? Please keep the answer short and order by decreasing plausibility."

The first two sentences describe the condition, the last two sentences give the answer instructions. A typical answer provided by ChatGPT looks as follows:

> The most plausible explanations for a 20-year-old professional runner experiencing a stinging pain, swelling, and stiffness in the forefoot, ordered by decreasing plausibility, could be:

- Overuse Injury: Repetitive stress from running may have led to an overuse injury such as metatarsalgia or stress fracture.
- Tendonitis: Inflammation of tendons in the forefoot, like extensor tendinitis, could cause these symptoms.
- Ligament Sprain: A sprained ligament, like in the metatarsophalangeal joint, could result in pain, swelling, and stiffness.
- Footwear Issues: Poorly fitting or worn-out running shoes may contribute to foot problems and discomfort.
- Infection or Insect Bite: Infection or an insect bite should be considered but is less likely without specific signs like redness and fever.

In order to obtain our rankings, we run the prompt 5 times, each time starting from an empty context. The possible causes provided in the 5 answers, define our set of outcomes O. We summarize and normalize the answers such that synonyms and syntactic differences do not lead to different outcomes. Given an answer list $o_1, \ldots, o_q$ for one prompt, we associate it with the partial ordering

$$o_1 > \cdots > o_q > \overline{\{o_1, \ldots, o_q\}}, \tag{8}$$

where for every subset $S \subseteq O$, $\overline{S} = O \setminus S$ denotes the complement of S and $o > S$ is short for $o > o'$ for all $o' \in S$. That is, the outcomes occuring in the answer are preferred according to their order of appearance and they all are preferred to those outcomes that have not occured. The outcomes that did not occur are incomparable with respect to this ranking.

Example 4.2. For our running example, we obtained the following outcomes after manual normalization:

(1) bu: bursitis,
(2) fi: footwear issues,
(3) go: gout,
(4) in: infection,
(5) mn: Morton's neuroma,
(6) msr: metatarsal stress reaction,
(7) ni: neurological issue,
(8) oi: overuse injury,
(9) pf: plantar fasciitis,
(10) sf: stress fracture,
(11) te: tendonitis,
(12) tr: trauma.

The 5 answers for our running example correspond to the following partial oderings:

$$\text{oi} >_1 \text{fi} >_1 \text{tr} >_1 \text{in} >_1 \text{ni} >_1 \overline{\{\text{oi, fi, tr, in, ni}\}},$$

$$\text{oi} >_2 \text{pf} >_2 \text{fi} >_2 \text{in} >_2 \text{go} >_2 \text{tr} >_2 \overline{\{\text{oi, pf, fi, in, go, tr}\}},$$

$$\text{oi} >_3 \text{tr} >_3 \text{fi} >_3 \text{in} >_3 \overline{\{\text{oi, tr, fi, in}\}},$$

$$\text{sf} >_4 \text{pf} >_4 \text{mn} >_4 \text{msr} >_4 \text{bu}, >_4 \overline{\{\text{sf, pf, mn, msr, bu}\}}$$

$$\text{oi} >_5 \text{te} >_5 \text{fi} >_5 \text{in} >_5 \overline{\{\text{oi, te, fi, in}\}}.$$

We constructed the partial orders in our running example manually. In our experiments, we will use a more automated process that works as follows:

(1) *Determine Base-Outcomes:* Query the LLM for a list of potential causes that we call *base-outcomes*.
(2) *Determine Rankings:* Repeatedly ask the LLM for the most plausible causes and to rank them by their plausibility. We call these outcomes *ranking-outcomes*.
(3) *Normalize Rankings:* Normalize the rankings by matching ranking-outcomes with base-outcomes. We use word embeddings (Sentence-BERT[22]) to map the ranking-outcomes to the most similar base-outcomes. If the similarity of a ranking-outcome to all base-outcomes is smaller than 0.5, it will be discarded (and reported).

One can think of other methodologies to compute rankings from LLMs. To abstract from the details, let us assume that we have a transformation method $T(Q, N, t)$ of the following form.

Definition 4.3. A *transformation method* $T(Q, N, t)$ takes a ranking query as input, prompts it N times and produces a profile $[\succeq_1, \ldots, \succeq_N]$ from the answer rankings. The parameter t represents the time at which the query has been prompted.

The time parameter t is only a technical device to take account of the fact that the output of LLMs is non-deterministic. It can also be seen as the (unknown) random seed of the LLM. The time parameter allows us talking about potentially different outputs when aggregating repeatedly for the same input. For example, say we aggregate the answers for Q five times and then again five times, then we can denote the two results by $T(Q, 5, t_1)$ and $T(Q, 5, t_2)$. We will use this notation for the discussion of the *consistency* property later. The notation is also useful to make the idea of robustness more precise. Assume that we have an aggregation method A that aggregates the profiles obtained from a transformation method T in some way. Roughly speaking, we say that a pair (T, A) consisting of a transformation method T and an aggregation method A is

query-robust if the answers obtained for one query Q from $A(T(Q, N, t_1))$ and $A(T(Q, N, t_2))$ are "similar" when N is chosen sufficiently large,

syntax-robust if the answers for two syntactically different, but semantically similar queries Q_1, Q_2 from $A(T(Q_1, N, t_1))$ and $A(T(Q_2, N, t_2))$ are "similar" when N is chosen sufficiently large.

The choice of the similarity measure depends on the application. Correlation measures seem to be a natural choice for measuring similarity between rankings. Measuring similarity between queries is more difficult. For experiments, one simple way to generate similar queries is to make purely syntactical changes to a base query to obtain (almost) semantically equivalent queries.

In our application, our aggregation method A_{PBW} ranks the diagnoses from the given profile by their PBW score. We will use correlation measures to determine the similarity of these rankings for $N = 5$ in our experiments.

4.2 Answer Aggregation

In order to quantify the plausibility of different answers, we apply the PBW score. The larger the score, the more plausible the answer.

Table 1: PBW scores for running example: first column shows, the outcomes, columns 2-6 show the partial PBW scores per ranking, column 7 shows the PBW scores and column 8 the normalized PBW scores rounded to two digits.

O	$\succ_1$	$\succ_2$	$\succ_3$	$\succ_4$	$\succ_5$	$s^{PBW}(o)$	$\bar{s}^{PBW}(o)$
bu	6	5	7	14	7	39	0.06
fi	20	18	18	6	18	80	0.12
go	6	14	7	6	7	40	0.06
in	16	16	16	6	16	70	0.10
mn	6	5	7	18	7	43	0.07
msr	6	5	7	16	7	41	0.06
ni	14	5	7	6	7	39	0.06
oi	22	22	22	6	22	94	0.14
pf	6	20	7	20	7	60	0.09
sf	6	5	7	22	7	47	0.07
te	6	5	7	6	20	44	0.07
tr	18	12	20	6	7	63	0.10

To make the interpretation of the scores easier, we normalize them such that all values are between 0 and 1. We let

$$\bar{s}^{PBW}(o) = \frac{s^{PBW}(o)}{\sum_{o' \in O} s^{PBW}(o')} \qquad (9)$$

Table 1 shows the PBW scores for our running example.

4.3 Properties

We now discuss some analytical guarantees of our approach. Let us note that the normalized PBW score $\bar{s}^{PBW}$ is just a rescaling of the PBW score s^{PBW}. Therefore, the outcomes with maximal score and their relative order remains unchanged. To begin with, let us reinterpret the properties from Theorem 3.3 in our setting.

Consistency: Let Q be a ranking query and let $p_1 = T(Q, N_1, t_1)$, $p_2 = T(Q, N_2, t_2)$. If o has maximum score with respect to both $\bar{s}^{PBW}(p_1)$ and $\bar{s}^{PBW}(p_2)$, then o also has maximum score with respect to $\bar{s}^{PBW}(p_1 \cup p_2)$.

Faithfulness: If we prompt the query only once, then the highest ranked outcome obtains the maximum score.

Neutrality: The score of outcomes is independent of their identity.

Cancellation: If for all outcomes $o_1 \neq o_2$, the number of rankings that rank o_1 above o_2 equals the number of rankings that rank o_2 above o_1, then all outcomes get the same score.

As explained before, the above properties are sufficient to characterize PBW scoring [3]. That is, there is no other scoring function that satisfies all these properties (up to affine transformations). Since all properties seem desirable in our setting, $\bar{s}^{PBW}$ is a natural choice. In the following proposition, we note some additional desirable properties of $\bar{s}^{PBW}$ and s^{PBW} in our setting. The properties also hold for other instantiations of (5) as long as $\alpha > \beta$ remains satisfied.

PROPOSITION 4.4. *Let Q be a query that was prompted N times and resulted in the outcomes O and profile $p = [\succ_1, \ldots, \succ_N]$.*

Partial Agreement: *If there are $o_1, o_2 \in O$ such that $o_1 \succ_i o_2$ for all $1 \leq i \leq N$, then $\bar{s}^{PBW}(o_1) > \bar{s}^{PBW}(o_2)$.*

Full Agreement: *If prompting the query repeatedly resulted in the same rankings, that is, $\succ_i = \succ_j$ for all $1 \leq i < j \leq N$, then $\bar{s}^{PBW}(o_1) > \bar{s}^{PBW}(o_2)$ if and only if $o_1 \succeq_i o_2$.*

Domination: *If there is an $o^* \in O$ such that $o^* \succ_i o$ for all $1 \leq i \leq N$ and $o \in O \setminus \{o^*\}$, then $\arg\max_{o \in O} \bar{s}^{PBW}(o) = \{o^*\}$.*

PROOF. 1. The assumptions imply that $\text{Down}_{\succeq_i}(o_1) > \text{Down}_{\succeq_i}(o_2)$ and therefore $w^{PBW}_{\succ_i}(o_1) > w^{PBW}_{\succ_i}(o_2)$ for all $1 \leq i \leq N$. Hence, $\bar{s}^{PBW}(o_1) > \bar{s}^{PBW}(o_2)$. The same is true for $\bar{s}^{PBW}$ because it is just a rescaling of s^{PBW}.

2. Since all rankings are equal, the outcomes are totally ordered by $\succeq = \succeq_1$ in our setting. Hence, if $O = \{o_1, \ldots, o_m\}$ and $o_1 \succeq o_2 \succeq \cdots \succeq o_m$, then $w^{PBW}(o_i) = 2 \cdot \text{Down}_{\succeq}(o_i) = 2 \cdot (m - i)$. Hence, $s^{PBW}(o_i) > s^{PBW}(o_j)$ if and only if o_i is ranked higher than o_j. The same is true for $\bar{s}^{PBW}$ because it is just a rescaling of s^{PBW}.

3. Partial agreement implies that $\bar{s}^{PBW}(o^*) > \bar{s}^{PBW}(o)$ for all $o \in O \setminus \{o^*\}$, which implies the claim. □

5 EXPERIMENTS

To assess the effectiveness of our approach, we conduct experiments on three sets of ranking queries from manufacturing, finance, and medicine. We first describe our methodology for generating ranking queries and extracting responses in a semi-automatic manner (Algorithm.1 provides an overview of the generation process). Subsequently, we will introduce the selected baseline approaches and the metrics used to assess the robustness of the aggregated answers. Code is available at https://github.com/boschresearch/RobustLLM/.

5.1 Generation of Ranking Queries

5.1.1 Generate Symptom-Cause Matrices. To generate ranking queries for our experiments, we first generate symptom-cause matrices, which contain information about a list of underlying critical problems and the possible symptoms we could observe.

We generate those matrices with ChatGPT. In the first step, we ask ChatGPT for a list of critical problems (causes) C in a specific domain with the following prompts:

> *"In manufacturing, what are the critical problems that can severely impact the health and overall performance of the factory? Output a list of those problems and rank them based on degree of risk to factory."*

> *"What are the critical financial problems that can severely impact the health and overall performance of a company? Output a list of those problems and rank them based on degree of risk to company."*
>
> *"What are common diseases with similar symptoms?"*

The first step, gives us the possible diagnoses for the domain. In the second step, we generate symptoms for each diagnosis with the following prompt:

> *"What can we observe in factory/company/human body to identify the underlying problem <the specific problem>? Output a list of indicators and rank them based on your confidence."*

Similar to Section 4.1, we summarize and normalize the symptoms into a list denoted as $\mathcal{S}$, eliminating redundancy arising from synonyms and syntactic variations. Subsequently, we generate matrices as presented in [20, Appendix A.1].

5.1.2 Sample Symptom Sets. In real-world scenarios, we have to make a diagnosis based on a set of symptoms. Given a list of symptoms $\mathcal{S}$ and diagnoses $\mathcal{D}$ for a particular domain, we let $s_q \subseteq \mathcal{S}$ be a subset of the symptoms, which is used in the condition description of a ranking query. For example, {*Unplanned maintenance, Increased rework and scrap, Increased product recalls, Increase cost, Increased carrying costs*} is a subset of size 5 for the manufacturing domain. We let s_d denote the set of all possible symptoms that we could observe for one specific diagnosis $d \in \mathcal{D}$.

The number of all potentially possible symptom sets (all subsets of $\mathcal{S}$) is too large for our experiments. To find a set of reasonable size, we first quantify the uncertainty of symptom sets and then sample a subset of symptom sets based on their uncertainty. Intuitively, the uncertainty of a symptom set is lowest if it uniquely identifies a diagnosis. The uncertainty is highest if all diagnoses are compatible with the symptom set.

We use the Jaccard similarity to measure the similarity between a symptom set s_q and the symptoms s_d associated with diagnosis d:

$$Sim(s_q, s_d) = \frac{|s_q \cap s_d|}{|s_q \cup s_d|} \qquad (10)$$

We normalize it such that, for every symptom set s_q, the similarity values to different diagnoses sum up to 1:

$$\overline{Sim}(s_q, s_d) = \frac{Sim(s_q, s_d)}{\sum_{d \in \mathcal{D}} Sim(s_q, s_d)} \qquad (11)$$

Finally, we quantify the uncertainty of symptom set s_q by calculating the normalized entropy of the similarity distribution:

$$U(s_q) = -\frac{1}{\log_2(|\mathcal{D}|)} \sum_{d \in \mathcal{D}} \overline{Sim}(s_q, s_d) \log_2(\overline{Sim}(s_q, s_d)) \qquad (12)$$

Note that the entropy is always between 0 and $\log_2(|\mathcal{D}|)$, hence our normalized entropy is always between 0 and 1.

In order to investigate our method in lower/higher uncertainty settings, we sample two types of symptom sets for each query set based on $U(s_q)$. Since the majority of symptom sets is in the high uncertainty region, we pick the 1000 lowest uncertainty symptom sets for the low uncertainty query set. For the high uncertainty set, we focus on sets with uncertainty between 0.7 and 0.8. More precisely, the two symptom sets have been computed as follows:

- low uncertainty symptom sets - S_{low}: we sort the potential symptom sets by normalized entropy and select the 1000 symptom sets with minimum normalized entropy.
- high uncertainty symptom sets - S_{high}: we randomly select 1000 symptom sets with normalized entropy in the range of 0.7 to 0.8.

> *"Given we observe <symptom 1>, <symptom 2>, ... what critical problems might exist in factory? Please output top 5 possible issues ranked by confidence without additional text."*
>
> *"Given we observe <symptom 1>, <symptom 2>, ... what critical financial issue might we have in our company? Please output top 5 possible issues ranked by confidence without additional text."*
>
> *"Given following symptoms: <symptom 1>, <symptom 2>, ... what disease might the patient have? Please output top 5 possible issues ranked by confidence without additional text."*

Figure 1: Query templates for evaluating query uncertainty

> *Variant 1: "Given we detect <symptom 1>, <symptom 2>, ... what essential issues might exist in factory? Please output top 5 possible issues ranked by confidence without additional text."*
>
> *Variant 2: "What potentially serious problems in the manufacturing may there be if we notice <symptom 1>, <symptom 2>, ... ? Please output top 5 possible issues ranked by confidence without additional text."*

Figure 2: Syntactic variants of the manufacturing query.

We visualize the uncertainty distribution of S_{low} and S_{high} with histograms in [20, Appendix A.2].

5.2 From Symptom Sets to Ranking Queries

We study robustness with respect to query and syntax uncertainty in our experiments. To evaluate query uncertainty, we convert symptom sets to ranking queries using the template in Figure 1. To evaluate syntax uncertainty, we designed two query variants to investigate the effect of syntactic query changes that are semantically meaningless. In the first variant, we only replace part of the words with synonyms without changing the structure of the queries (e.g. we replace "observe" with "detect" and replace "critical problems" with "essential issues"). In the second variant, we also change the structure of the query. An an example, Figure 2 shows the variants of the manufacturing ranking template stated before.

5.3 Evaluation Protocol

We evaluate the robustness of our approach over three batches of ranking queries i.e. manufacturing, finance and medical queries, compared with two baseline approaches.

5.3.1 Baselines.

Algorithm 1 The pseudocode of ranking query generation

/* generate symptom-cause matrices */
$\mathcal{D} \leftarrow$ query ChatGPT
for $d \in \mathcal{D}$ **do**
    $s_q \leftarrow$ query ChatGPT
    add s_q to a list: $\mathcal{L}_s$.append(s_q)
end for
Symptom-Cause matrix $\leftarrow$ summarize and normalize $\mathcal{L}_s$

/* sample symptom sets */
for $s_q \in S_q$ **do**
    calculate Jaccard similarity: $Sim(s_q, s_d) \leftarrow \frac{|s_q \cap s_d|}{|s_q \cup s_d|}$
    normalization: $\overline{Sim}(s_q, s_d) \leftarrow \frac{Sim(s_q, s_d)}{\sum_{c \in C} Sim(s_q, s_d)}$
    calculate normalized entropy:
    $U(s_q) \leftarrow -\frac{1}{\log_2(|C|)} \sum_{c \in C} \overline{Sim}(s_q, s_d) \log_2(\overline{Sim}(s_q, s_d))$
    add $U(s_q)$ to the list: List of indicator entropy.append($U(s_q)$)
end for
Rank the S_q based on the normalized entropy (from largest to smallest).
$S_{low} \leftarrow$ the last 1000 indicator sets.
$S_{high} \leftarrow$ randomly select 1000 symptom sets with normalized entropy in the range of 0.7 to 0.8.

/* generate ranking queries from symptom sets */
Convert symptom sets into ranking queries using query template.

- **Without Aggregation**: we do not aggregate rank answers and directly evaluate the robustness of single answers.
- **Average Rank**: we treat each rank preference equally and aggregate the ranks by simply averaging the ranks. Given N ranks $r_1 \ldots r_N$ to be aggregated, the aggregation function is defined as $A(r_1 \ldots r_N) = \frac{1}{N} \sum_{i=1}^{N} r_i$ in this case.

5.3.2 Evaluation Metrics. We use **Kendall's** rank correlation coefficient (R_τ) [13] and **Spearman's** rank correlation coefficient (R_s) [26] to evaluate the robustness of the aggregated ranks.

Let n be the number of items to be ranked. Kendall's rank correlation coefficient is defined as follows:

$$R_\tau = \frac{C - D}{\binom{n}{2}} = \frac{2(C - D)}{n(n - 1)}, \tag{13}$$

where C is the number of concordant pairs (pairs that have the same order in predicted and ground truth ranks) and D is the number of discordant pairs (pairs that have different order in both ranks). A higher R_τ value indicates a better match between the predicted and true ranks.

Spearman's rank correlation coefficient is defined as follows:

$$R_s = \frac{cov(rank1, rank2)}{\sigma_{rank1} \cdot \sigma_{rank2}}, \tag{14}$$

where $cov(.)$ is the covariance between two variables and σ is the standard deviation. Similar to Kendall's tau, a higher R_s value indicates a better match between the predicted and true ranks. Algorithm.2 illustrates our approach to evaluating ranking queries.

Algorithm 2 The pseudocode of ranking query evaluation

/* Evaluate query robustness*/
Require: Q
    for $i \leftarrow 1 \ldots K$ **do**
        $p_i \leftarrow A(T(Q, N, t_i))$
        add p_i to a list: P.append(p_i)
    end for
    $R_Q = \frac{1}{2\binom{K}{2}} \sum_{p_1, p_2 \in P, p_1 \neq p_2} score(p_1, p_2)$

/* Evaluate syntax robustness*/
Require: $Q_1 \ldots Q_K$
    for $i \leftarrow 1 \ldots K$ **do**
        $p_i \leftarrow A(T(Q_i, N, t_i))$
        add p_i to a list: P.append(p_i)
    end for
    $R_Q = \frac{1}{2\binom{K}{2}} \sum_{p_1, p_2 \in P, p_1 \neq p_2} score(p_1, p_2)$

5.4 Experiment Settings

In our experiments, we evaluate the robustness of the answer with respect to repeated queries (query uncertainty) and syntactic changes (syntax uncertainty). Algorithm 2 explains briefly how we evaluate the robustness of the aggregated ranks.

5.4.1 Evaluation of Query Uncertainty. To evaluate the robustness with respect to repeated queries, we query ChatGPT N times with ranking query Q at time $t_1 \ldots t_K$ and aggregate the answers with aggregation function A to get K aggregated answers. Note in our experiment, we specifically set $K = 3$ and $N = 5$ ($N = 1$ when A is "without aggregation", since we do not aggregate answers in this baseline). The overall robustness of the query R_q is evaluated by calculating pairwise Kendall's and Spearman's rank correlation coefficient (we use $score(x, y)$ to denote the calculation of both coefficients) and averaging the coefficients. The mean values and standard deviation of all R_q is reported in our results.

5.4.2 Evaluation of Syntax Uncertainty. We also evaluate the robustness with respect to syntactic changes, the process is very similar to evaluation of query uncertainty. The only difference is that in this case, instead of repeatedly aggregating outputs for the same Q, we use K different ranking queries with the same semantic meaning but different syntax. In our experiment, $K = 3$.

5.5 Evaluation of Query Uncertainty

Table 2 presents results for query uncertainty. Our approach consistently outperforms both baselines, "without aggregation" and "average rank," across all three ranking query sets, demonstrating its superiority in both high and low uncertainty scenarios.

5.6 Evaluation of Syntax Uncertainty

Table 2 provides an overview of the outcomes pertaining to syntax uncertainty. Our approach outperform both baseline methods in the majority of scenarios examined. We observe a substantial reduction of both Kendall's and Spearman's coefficients compared to Table 3. This suggests that syntactic variants introduce more variability.

Table 2: Evaluation of query uncertainty: we submit the same ranking query to ChatGPT-turbo five times and then aggregate the results. We repeat this process three times and evaluate the robustness of the three aggregated results. The temperature is set to 1, which is the default setting in the web version of ChatGPT.

	High uncertainty ranking queries						Low uncertainty ranking queries					
	without aggregation		average rank		PBW (our)		without aggregation		average rank		PBW (our)	
Dataset	Kendall	Spearman	Kendall	Spearman	Kendall	Spearman	Kendall	Spearman	Kendall	Spearman	Kendall	Spearman
Manufacturing	0.29 (0.29)	0.34 (0.33)	0.63 (0.14)	0.76 (0.13)	**0.78 (0.09)**	**0.84 (0.08)**	0.33 (0.3)	0.38 (0.34)	0.62 (0.13)	0.74 (0.13)	**0.75 (0.09)**	**0.81 (0.09)**
Finance	0.49 (0.45)	0.53 (0.46)	0.74 (0.2)	0.82 (0.18)	**0.81 (0.11)**	**0.86 (0.09)**	0.57 (0.43)	0.6 (0.44)	0.75 (0.18)	0.82 (0.16)	**0.79 (0.11)**	**0.84 (0.1)**
Medical	0.54 (0.44)	0.59 (0.46)	0.75 (0.28)	0.81 (0.27)	**0.83 (0.14)**	**0.88 (0.13)**	0.56 (0.6)	0.58 (0.61)	0.67 (0.53)	0.7 (0.54)	**0.84 (0.23)**	**0.85 (0.23)**

Table 3: Evaluation of syntax uncertainty: we submit the same ranking query to ChatGPT-turbo five times and then aggregate the results. We repeat this process for three syntactic variants and evaluate the robustness of the three aggregated results. The temperature is set to 1, which is the default setting in the web version of ChatGPT.

|	High uncertainty ranking queries						Low uncertainty ranking queries					
	without aggregation		average rank		PBW (our)		without aggregation		average rank		PBW (our)	
Dataset	Kendall	Spearman	Kendall	Spearman	Kendall	Spearman	Kendall	Spearman	Kendall	Spearman	Kendall	Spearman
Manufacturing	0.25 (0.2)	0.29 (0.23)	0.27 (0.19)	0.32 (0.22)	**0.43 (0.18)**	**0.46 (0.19)**	0.31 (0.23)	0.35 (0.25)	0.31 (0.2)	0.37 (0.23)	**0.46 (0.2)**	**0.49 (0.21)**
Finance	0.56 (0.32)	0.62 (0.31)	0.57 (0.25)	0.63 (0.25)	**0.66 (0.17)**	**0.71 (0.18)**	0.61 (0.34)	0.67 (0.32)	0.55 (0.25)	0.62 (0.25)	**0.66 (0.16)**	**0.72 (0.16)**
Medical	0.64 (0.28)	0.71 (0.26)	0.71 (0.22)	0.78 (0.2)	**0.8 (0.16)**	**0.84 (0.15)**	0.83 (0.26)	0.85 (0.24)	0.84 (0.22)	**0.87 (0.19)**	**0.85 (0.18)**	0.86 (0.18)
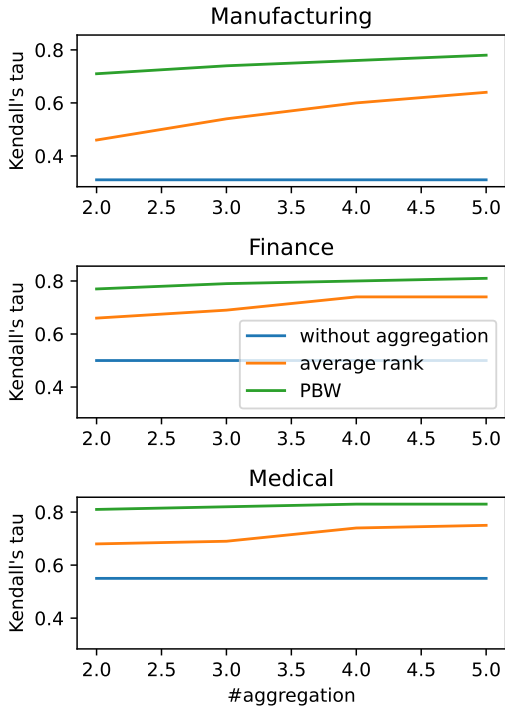
Figure 3: Robustness with respect to the number of answers used for aggregation.

5.7 Evaluation of Sample Efficiency

Another important question is how many answers do we need to aggregate? That is, how should we choose the parameter N for our transformation method $T(Q, N, t)$. Figure 3 shows the robustness with respect to the number of answers used for aggregation. We can see that even aggregating only two answers with our approach

can already significantly increase the robustness. Note that figure 3 shows the robustness in case of query uncertainty (high uncertainty version) and only Kendall's tau is reported. However, we observed similar trends for other settings.

6 CONCLUSIONS

To improve the robustness of the answers from LLMs, we suggest to sample answers repeatedly and to aggregate the answers using social choice theory. Our approach is based on the Partial Borda Choice function as it gives several interesting analytical guarantees. Our investigation primarily focuses on the application of ranking queries within diagnostic contexts, such as medical and fault diagnosis. Our experiments show that our approach significantly improves the robustness against both query and syntax uncertainty.

Queries that ask for a single most plausible answer can be understood as a degenerated special case of our ranking queries. This is because an answer o can be understood as the partial preference $o > \overline{O \setminus o}$ (the provided answer is ranked above all other answers and the ranking is indifferent about all other answers). In this special case, our average rank baseline corresponds to majority voting. One interesting venue for future work is to compare partial Borda voting in the single-answer setting to other nonranking voting methods. One may also interpret the ranking as an expression of approval (the answer approves of a diagnosis if it is mentioned) and to aggregate the answers by using approval voting methods.

Uncertainty in LLM outputs can also be caused by meaningless information or adversarial attacks injected into the queries. In future work, we aim to investigate whether social choice theory methods can also be applied to effectively improve the robustness of LLM outputs in the presence of such perturbations.

7 ACKNOWLEDGEMENTS

REFERENCES

[1] David Alvarez-Melis and Tommi S. Jaakkola. 2018. On the Robustness of Interpretability Methods. 2018 Workshop on Human interpretability in Machine Learning.

[2] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia (Eds.). 2016. *Handbook of Computational Social Choice.* Cambridge University Press. https://doi.org/10.1017/CBO9781107446984

[3] John Cullinan, Samuel K Hsiao, and David Polett. 2014. A Borda count for partially ordered ballots. *Social Choice and Welfare* 42 (2014), 913–926.

[4] Angela Fan, Mike Lewis, and Yann N. Dauphin. 2018. Hierarchical Neural Story Generation. In *Annual Meeting of the Association for Computational Linguistics (ACL).* Association for Computational Linguistics, 889–898.

[5] Hidde Fokkema, Rianne de Heide, and Tim van Erven. 2023. Attribution-based Explanations that Provide Recourse Cannot be Robust. *Journal of Machine Learning Research* 24, 360 (2023), 1–37. http://jmlr.org/papers/v24/23-0042.html

[6] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On Calibration of Modern Neural Networks. In *ICML (Proceedings of Machine Learning Research, Vol. 70).* PMLR, 1321–1330.

[7] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The Curious Case of Neural Text Degeneration. In *ICLR.* OpenReview.net.

[8] Xuedong Huang, Alex Acero, Hsiao-Wuen Hon, and Raj Reddy. 2001. *Spoken language processing: A guide to theory, algorithm, and system development.* Prentice hall PTR.

[9] Peter J. Huber. 1981. *Robust Statistics.* Wiley. https://doi.org/10.1002/0471725250

[10] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *Comput. Surveys* 55, 12 (2023), 1–38.

[11] Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics* 9 (2021), 962–977.

[12] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. 2023. Language models (mostly) know what they know. *Findings of the Association for Computational Linguistics* (2023), 8653−-8665.

[13] Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika* 30, 1/2 (1938), 81–93.

[14] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic Uncertainty: Linguistic Invariances for Uncertainty Estimation in Natural Language Generation. In *International Conference on Learning Representations ICLR.*

[15] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* 36, 4 (2020), 1234–1240.

[16] Francesco Leofante and Nico Potyka. 2024. Promoting Counterfactual Robustness through Diversity. In *AAAI Conference on Artificial Intelligence (AAAI'24).*

[17] Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Teaching Models to Express Their Uncertainty in Words. *Transactions on Machine Learning Research* (2022).

[18] Sabrina J. Mielke, Arthur Szlam, Emily Dinan, and Y-Lan Boureau. 2022. Reducing Conversational Agents' Overconfidence Through Linguistic Calibration. *Transactions of the Association for Computational Linguistics* 10 (2022).

[19] Theo Olausson, Alex Gu, Benjamin Lipkin, Cedegao E. Zhang, Armando Solar-Lezama, Joshua B. Tenenbaum, and Roger Levy. 2023. LINC: A Neurosymbolic Approach for Logical Reasoning by Combining Language Models with First-Order Logic Provers. In *2023 Conference on Empirical Methods in Natural Language Processing (EMNLP).* Association for Computational Linguistics, 5153–5176. https://aclanthology.org/2023.emnlp-main.313

[20] Nico Potyka, Yuqicheng Zhu, Yunjie He, Evgeny Kharlamov, and Steffen Staab. 2023. Robust Knowledge Extraction from Large Language Models using Social Choice Theory. *arXiv preprint arXiv:2312.14877* (2023).

[21] Laila Rasmy, Yang Xiang, Ziqian Xie, Cui Tao, and Degui Zhi. 2021. Med-BERT: pretrained contextualized embeddings on large-scale structured electronic health records for disease prediction. *NPJ digital medicine* 4, 1 (2021), 86.

[22] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing EMNLP-IJCNLP.* 3982–3992.

[23] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR).*

[24] Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D Manning. 2023. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. *Conference on Empirical Methods in Natural Language Processing EMNLP* (2023).

[25] Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. 2023. Can LLMs Express Their Uncertainty? An Empirical Evaluation of Confidence Elicitation in LLMs. *arXiv preprint arXiv:2306.13063* (2023).

[26] Jerrold H Zar. 2005. Spearman rank correlation. *Encyclopedia of Biostatistics* 7 (2005).

[27] Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. 2023. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology* (2023).