ChatGPT

Detecting Questions and Imperative Sentences in English

Introduction

In conversational voice assistants, understanding **whether a user utterance is a question or an imperative command** is crucial. A *question* (interrogative) typically asks for information (e.g. *"What time is it?"*), whereas an *imperative* issues a directive or request (e.g. *"Set a timer for 5 minutes."*). Detecting these forms in English can be challenging, especially in spoken input where punctuation and clear structure may be absent. This report surveys tools and techniques for question/imperative detection in English text and speech, covering rule-based methods, machine learning (ML) approaches, deep learning techniques, prosodic (audio) cues, relevant datasets/benchmarks, available libraries/APIs, and known limitations.

1. Heuristic and Rule-Based Methods

Pattern-Based Identification: Simple heuristic rules can capture many questions and commands: - **Punctuation and Keywords:** In text, a question often ends with a `?`. Presence of a leading wh-word (*who, what, when, where, why, how*) or an inverted auxiliary verb (*Do*, *Can*, *Is* at the start) strongly indicates a question. Conversely, imperatives often end with `!` in writing, or have words like *please* for polite requests. - **Constituency Patterns:** Traditional parsers (e.g. Stanford PCFG) label question structures with grammar tags (like `SBARQ` or `SQ` in Penn Treebank). This can directly flag interrogatives in text. By contrast, imperatives usually parse as a verb phrase without an explicit subject. - **Dependency & POS Clues:** English imperatives often start with a base-form verb (VB) or modal (MD) and have an implied second-person subject ("you") that is omitted [1]. For example, *"Open the door"* has **"Open"** as the root verb with no `nsubj` (nominal subject) – a strong indicator of imperative mood [2]. Heuristic rules can exploit this: - If the first word's part-of-speech is a base verb (VB) or modal (MD) and the sentence doesn't end in `?`, classify as imperative [3] [4]. - If a sentence begins with a proper noun or interjection followed by a verb (*"Alice, listen"*, *"Please listen"*), it can still be imperative – rules may chunk the initial phrase and check if a verb phrase follows [5]. - **Example:** *"Just take a seat."* – Begins with an adverb *Just* then verb *take*; a chunking rule can identify the first chunk as a verb phrase, hence imperative [5]. - **Negative Imperatives & Question-Tags:** Some imperatives include question-like phrasing (e.g. *"Do this, will you?"*). Rules can catch these by looking for tag questions at the end (e.g. *", will you?"*) combined with an initial verb [6].

Rule-Based Question Detection: Similar straightforward rules help identify questions: - If a sentence ends with `?` or begins with an auxiliary verb (e.g. *"Do you…", "Are we…"*) or starts with a wh-word (*"What/When/How…"*), classify as a question. - Check syntax: a **constituency parse** yielding a question form (SBARQ/SQ) is a clear signal. For example, the parse tree of *"So do you go to college right now?"* is labeled as a Yes-No-Question [7]. - **Dependency parse** alone does not mark interrogatives (Universal Dependencies has no direct "question" feature aside from interrogative pronoun tags) [8]. So one must rely on tokens: e.g. presence of a `?` punctuation token, or interrogative pronoun (`WP/WRB` tags) which, however, can appear in indirect questions too [8]. - **Example:** *"Who cares?"* – parse will show **"Who"** as `nsubj` and **"cares"** as root verb, plus a `?` punct token; the combination of wh-pronoun + question mark is a strong indicator of an interrogative.

Using POS and Chunking: Custom chunk grammars can encode these rules. One approach is to define regex-like POS patterns for imperatives [9], for instance: - *<VB-Phrase>: {^[VB|MD].} to match sentences starting with a verb or modal.* - \<VB-Phrase>: {<UH|RB|NN.><*,>?<VB|VBP>}* to catch polite forms (*“Please, do…”*, *“Everyone, listen”*).
If the first chunk matches a `VB-Phrase` pattern, mark the sentence as imperative [5]. Similarly, one can define patterns for questions (e.g. initial `WP/WRB` or sentence-final question tag).

Precision vs. Recall: Rule-based methods are fast and transparent but can miss nuanced cases. They work well for prototypical forms, yet they can be **overly broad or narrow**: - A naive imperative rule (“first word is VB → imperative”) may misclassify common expressions or ellipses. For example, *“Thank you.”*, *“Guess not.”*, *“See you.”* all start with a base verb or imperative-looking form but are not commands. In one analysis, such cases contributed significantly to false positives when using a simple “verb-first” rule [10]. Fine-tuning the rules (e.g. excluding set phrases like “thank you”) can mitigate this [10]. - Polite or indirect commands phrased as questions (like *“Could you open the window?”*) will be labeled as questions by surface rules, even though pragmatically they function as requests. Pure syntax-based detection would miss that these are imperatives in intent. Handling these requires more complex rules or pragmatic interpretation [11]. - Some interrogative forms lack typical markers. **Declarative questions** (e.g. *“You’re coming along?”* – syntactically a statement but spoken with question intonation) can fool punctuation-based methods [12]. They need either prosodic cues or deeper parsing of structure and context.

In summary, heuristic detection (via punctuation, key phrases, POS, or parse patterns) provides a strong baseline. Many voice platforms start with such rules because they are straightforward to implement (e.g. check last character for `?` or first word for “please” or a verb). However, complex or colloquial utterances often require the robustness of ML-based methods described next.

2. AI/ML-Based Approaches (Supervised & Unsupervised)

Rather than manual rules, **machine learning models** can learn to classify sentences as question vs. imperative (or other classes) from data. This is typically framed as a supervised classification task: given an utterance (text or transcript), output a label like *Question*, *Imperative*, or *Statement*. Key techniques include:

- **Feature-Based Supervised Models:** Earlier work used features such as words, n-grams, and prosodic cues to train classifiers (e.g. logistic regression, SVMs, or decision trees). For example, Stolcke et al. (2000) built a statistical dialog act model for Switchboard dialogues, using lexical cues (e.g. the presence of wh-words, verb tense) and achieving good accuracy on identifying questions vs statements [13]. Similarly, **Shriberg et al. (1998)** explored decision trees with acoustic-prosodic features to classify dialogue acts (including question detection) and found that these features can improve accuracy [13]. In fact, they observed that certain prosodic signals (like extended pause or final rise) help distinguish questions from statements beyond lexical information [13]. Another study by Ang et al. (2005) combined prosody with language model features to automatically insert punctuation (commas, periods, question marks) in speech transcripts [14] – essentially learning to detect questions via punctuation prediction.

- **Learning from Punctuation as Proxy:** A clever approach is to leverage large text corpora with punctuation as a form of supervision. For instance, **Margolis & Ostendorf (2011)** trained a question detector using sentences ending in `?` from Wikipedia talk pages as positive examples [15] [16]. This text-trained model (using only lexical features) achieved about 90% of the AUC performance of a model trained on in-domain spoken data with rich features [12]. In other words, even without prosody, the lexical patterns learned from punctuated text were largely effective at identifying questions in speech transcripts [12]. The shortfall was mainly on **declarative questions**, which lack obvious lexical markers [17]. The researchers further improved performance via **domain adaptation** – incorporating unlabelled spoken utterances and their prosodic features to adapt the text-trained model [18]. This semi-supervised method helped bridge stylistic differences between written questions and spoken questions.

- **Dialogue Act Classification:** More broadly, question/imperative detection can be seen as a subset of **dialogue act classification**, where an utterance is tagged with its communicative function (question, request/command, statement, backchannel, etc.) [19]. Standard benchmarks like the Switchboard Dialogue Act Corpus (SwDA) and Meeting Recorder Dialogue Act (MRDA) corpus provide labeled data for this. Classical ML approaches (pre-Deep Learning) on these corpora used algorithms like Naive Bayes or Maximum Entropy with features such as:

- Bag-of-words or n-grams (e.g. presence of `?` or words like *"please"*, *"what"*).
- POS tag patterns or rule-based features (e.g. binary feature "starts_with_verb").
- Prosodic features from audio (e.g. average pitch, final pitch rise, pause duration before speech).

- These models already demonstrated that **lexical cues alone can achieve high accuracy** in question detection when transcripts are error-free [13], and adding prosody yields incremental gains [13]. For example, Kim & Woodland (2003) reported that lexical features from an accurate transcript predicted question vs. statement almost as well as adding intonation features – indicating English word order and vocabulary carry strong signals for interrogatives [13].

- **Unsupervised and Weakly Supervised Methods:** In scenarios with limited labeled data (especially for imperatives, which might be rarer in some corpora), researchers have tried unsupervised techniques:

- **Clustering:** One might cluster utterances based on features to see if questions naturally group by having certain patterns (e.g. high pitch end or containing wh-words). However, dialogue acts are often too subtle for pure unsupervised clustering to cleanly separate them.
- **Heuristic Labeling for Training:** A practical strategy is to auto-label a dataset using simple rules, then train a classifier on this "silver" data. For example, label any sentence ending in `?` as a question (and others as non-questions) from a large corpus; train a model; then apply it to conversational input. This is essentially what the Wikipedia-talk approach did [16] – using punctuation in text as heuristic labels for training data.
- **Domain Adaptation:** As mentioned, Margolis et al. leveraged unlabeled in-domain data (spoken meetings) by adapting a text-trained classifier [18]. Techniques like self-training (where a model labels unlabeled data and retrains) or adversarial domain adaptation can help a model generalize from one genre (written text) to another (spoken dialogue) without full supervision.

Performance of Traditional ML: These approaches achieved reasonably good results. For instance, on the MRDA meeting corpus, a supervised ML classifier could detect questions with high accuracy (the text+prosody model in Margolis (2011) had an AUC close to the fully supervised case [12]). In email domain, researchers like Cohen et al. (2004) and Kwong & Yorke-Smith (2012) studied *speech acts* in emails, training classifiers to detect **questions vs requests in threads** to aid summarization [20]. They showed ML can identify imperative requests (e.g. "Please review the draft") and questions in text emails by learning from annotated examples.

Overall, classic ML methods laid the groundwork by showing that a combination of **syntactic cues and prosodic features** can be learned to distinguish interrogative and imperative moods. However, maintaining many features and rules for different domains was labor-intensive, which led to the adoption of data-driven deep learning methods described next.

3. Deep Learning Techniques (LSTM, Transformers, etc.)

Modern deep learning approaches have significantly advanced the accuracy of question and imperative detection. These models automatically learn rich representations of language (and even prosody, in some cases) without the need for manual feature engineering:

- **Recurrent Neural Networks (RNNs):** Sequence models like LSTMs and GRUs have been applied to dialogue act classification, treating each utterance as a sequence of words to encode. A unidirectional or bidirectional LSTM can output a representation of the sentence, which a softmax layer classifies as question, command, or other. For example, an LSTM-based model achieved around 79–82% accuracy on the 42-class Switchboard dialog act task [21], which includes multiple question types and action-directives. When focused on a binary or ternary classification (question vs imperative vs other), an LSTM can perform even better due to the simpler decision boundary. Researchers often improve RNN classifiers by using **hierarchical models** (contextual LSTMs) that consider not only the current utterance but also preceding utterances in the conversation. This helps, for instance, to distinguish *"Right."* as a backchannel vs. as a question (which might depend on context). Hierarchical RNN+CRF models have topped dialog act benchmarks like MRDA with over 90% accuracy [22]. In these models, the RNN encodes word sequences and a Conditional Random Field layer captures label dependencies (e.g. an answer is likely to follow a question) to refine predictions [23].

- **Transformer Models (BERT, T5, GPT):** Transformer-based language models pre-trained on vast text corpora have proven extremely effective at sentence classification tasks. **BERT** in particular, when fine-tuned, can leverage its deep understanding of syntax and context to detect subtle cues of interrogative vs imperative mood. Fine-tuning BERT on a dialogue-act corpus yields state-of-the-art results; for example, experiments report ~83% accuracy on Switchboard DA classification using BERT variants [24]. BERT can implicitly learn patterns like inversion or the presence of directive verbs without explicit feature coding. Similarly, **RoBERTa or ALBERT** models have been used to classify utterances (some research has used BERT-based models to classify each utterance in a conversation as one of the dialog act types, showing strong gains over older RNN methods [25]). HuggingFace's Transformers library makes it straightforward to fine-tune such models on custom data for question/imperative detection.

- *Example:* A BERT fine-tuned on a labeled dataset where class 0 = statement, 1 = question, 2 = command could learn that *"Could you pass the salt"* (despite the `?`) is likely a request (imperative) because similar structures in training were labeled as requests. It might also use subtle cues like presence of polite phrasing *"could you"* to differentiate it from an actual yes-no question. This is the kind of nuance that large language models can capture beyond surface form.

- **Sequence-to-Sequence and Generative Models:** Models like **T5 (Text-To-Text Transfer Transformer)** treat every NLP problem as text generation. We could feed an utterance into T5 and ask it to generate a label like "question" or "imperative." T5's language understanding often matches BERT's, and this approach has flexibility (e.g. the same model could be prompted to also paraphrase or convert a statement to a question). Similarly, **GPT**-family models (GPT-3, GPT-4, etc.) can, via prompting, identify whether a sentence is a question or a command. For instance, prompting GPT-4 with: *"User: 'Open the pod bay doors please.' Assistant: Is this a question or a* *command?"* would likely yield "a command." However, using large generative models for this classification is usually overkill in production due to cost and latency, unless a platform already employs them for broader NLU tasks. Instead, smaller fine-tuned transformers are preferred for real-time systems.

- **Convolutional Neural Networks (CNNs):** Some works have also used CNNs on text (treating the sentence as a sequence of word embeddings and applying convolution+pooling to capture key phrases). CNNs have been shown to perform well on sentence classification including question identification, with the advantage of being fast. For example, a CNN model was used in an early deep learning approach to punctuation prediction [26]. However, transformers have largely superseded CNNs for NLP tasks by capturing longer-range dependencies better.

- **Multi-Task and Joint Learning:** Detecting questions/imperatives might be improved by jointly learning related tasks:

- **Intent Classification:** In voice assistants, typically an *Intent classifier* determines the user's intention (e.g. *GetWeather*, *SetTimer*, *GeneralQuestion*). The model might inherently learn that some intents correspond to questions (e.g. a *KnowledgeQuery* intent might always be interrogative), while others correspond to commands (*MusicPlay* intent often comes from an imperative like "Play [song]"). Multi-task models have been proposed that perform dialogue act classification alongside intent detection and even slot filling [27]. A shared representation can benefit all these tasks – the model learns general language patterns and context, then specialized heads output both the act (question/command) and the specific intent domain. For example, an encoder might first classify an utterance as question vs request, and that information flows into the intent prediction for finer distinctions. This approach was shown to improve performance on each subtask compared to training them in isolation [27].
- **Contextual Dialog Models:** Deep models can also incorporate conversation context (previous turns, who is speaking, etc.) as additional input for classification. This helps because whether something is a question can depend on context. Example: *"Right?"* spoken alone could be a question (tag question confirming something) or just a discourse marker. If the previous turn was an explanation, *"Right?"* is likely a confirmation-seeking question. Context-aware Transformers (using self-attention across multiple utterances) have been developed for dialog act tagging and show improved accuracy, especially on short or ambiguous utterances [28].

- **Speech-Integrated Deep Models:** While many deep learning models focus on text input, there are end-to-end or multi-modal networks that incorporate audio features for speech utterances:

- One approach is a **dual encoder**: e.g. a Bi-LSTM or Transformer processes the word sequence (from ASR) while another network (possibly CNN or an RNN) processes a sequence of acoustic features (pitch, energy over time). The outputs are combined (concatenation or attention fusion) and passed to a classifier. This allows the model to learn, say, that an utterance ending with a rising pitch and the word "okay" should be a question, whereas falling pitch with the same word might be a statement. Some recent research in punctuation prediction followed this pattern, adding prosodic feature streams to transformer-based text models [29] [30]. Cho et al. (2022) demonstrated that adding intonation and energy features to a transformer model significantly improved question mark prediction in conversational speech transcripts [31] [32].
- There are also fully **end-to-end audio classification** models (e.g. using Wav2Vec 2.0 or other speech embeddings) that can directly classify an audio clip as question vs statement without intermediate text. These typically require substantial training data with labeled audio. One example might be training a simple CNN on spectrograms to detect a final rising pitch, but more

robustly, one would use a pre-trained ASR's encoder and fine-tune it for classification. However, this is less common; usually the text is available via ASR, so leveraging text plus a few targeted prosody features is more practical than reinventing a full audio classifier.

Performance: The deep learning techniques have pushed accuracy closer to human performance on benchmark datasets. On Switchboard, state-of-the-art dialog act models (often transformer-based) exceed 80–85% accuracy overall [24], and even higher on two-class or three-class distinctions. On the MRDA meetings corpus, deep models achieve over 90% accuracy [22]. These models are generally robust to variations in wording and can handle indirect forms better than rigid rules. For instance, a BERT model might correctly label *"Would you mind telling me the time"* as a question (or at least as an *information request* act) whereas a naive rule might be confused by the polite phrasing and question syntax. The richness of deep models also helps in detecting imperatives that don't strictly start with a verb (e.g. *"One more song."* said in a commanding tone could be recognized as a request to play another song based on context learned during training).

Of course, deep models require training data. In practice, one might fine-tune a model on a combination of general dialog act corpora and assistant-specific data (actual user queries labeled as question or command). Transfer learning from large language models means even relatively small fine-tuning datasets can yield good results, as the model already encodes a lot of English syntax and usage patterns.

4. Speech-Specific Considerations (Intonation and Prosody)

When dealing with *spoken input*, detecting questions and imperatives isn't just about word sequences – **how** something is said can be equally important: - **Intonation Patterns:** English yes/no questions often have a characteristic rising intonation towards the end, whereas statements typically have a falling intonation. Wh-questions usually end in a falling tone (similar to statements), but overall question intonation can differ subtly from declarative intonation. Imperative sentences can have a sharp, assertive tone (especially if it's a command) or even a rising tone if phrased as a polite request or a plea. Prosodic cues like **pitch contour** thus provide valuable signals. Research in prosody has identified that a rising final F0 (fundamental frequency) is a strong cue for yes-no questions. For example, an utterance *"It's five o'clock?"* may rely on a high rising terminal to indicate it's a question, in the absence of subject-auxiliary inversion. Systems can extract features such as the slope of pitch at the end of the utterance, or more explicitly, measure the difference between pitch in the final 200 ms vs the preceding segment [33] [34]. If the final pitch is significantly higher than the mid-utterance pitch, that suggests a question intonation (H-H% in intonational phonology terms).

- **Duration and Pausing:** It's not only pitch. Studies have shown **timing features** play a role too. Shriberg et al. found that **final lengthening and pausing** were even more predictive than pitch for question detection in conversational speech [35]. A slight elongation of the last syllable or a telling pause pattern can signal a question. For example, speakers often pause differently after questions (perhaps expecting an answer). A model might use features like the duration of the last vowel or the presence of a following silence of certain length as input. Such features are obtainable from audio by force-aligning the transcript or using voice activity detection to measure pauses.

- **Energy and Emphasis:** Commands might be delivered with higher intensity or a clipped, firm delivery. For instance, *"Stop!"* as a command could have a higher volume and shorter duration than *"Stop?"* (as in *"Stop?"* said in a questioning tone). Features like average intensity (loudness) and energy contour might thus contribute. That said, energy can be confounded by the speaker's emotion or distance from microphone, so it's a secondary cue.

- **Prosody in ASR and Punctuation Restoration:** Modern speech-to-text frameworks often attempt to infer punctuation from audio implicitly. **Google Cloud Speech API**, **Microsoft Azure Speech**, **AWS Transcribe**, and others output punctuated transcripts – if you ask a question, they will likely include a `?` in the result. They achieve this using internal models that combine acoustic and language model evidence. In essence, these systems have a built-in question detection module as part of their **punctuation restoration** stage. For example, Amazon's model might notice a rising intonation and a syntax match for a question, and decide to insert a question mark in the final text. Some research systems explicitly integrate prosody for punctuation: Cho et al. (2022) found adding F0 and energy features improved question mark prediction F1 on conversational transcripts, especially for spontaneous speech where purely text-based models struggle [31] [32]. In formal or read speech, language models alone can guess punctuation well, but in dialogues (which often lack clear syntactic structure and contain disfluencies) prosody adds significant value [30].

- **Turn-Taking Signals:** In conversation, a rising intonation can also be a turn-yielding cue (inviting the other to speak). A voice assistant might monitor intonation to decide if the user has finished speaking or if they are asking a question expecting an answer. Research at Stanford (e.g. on turn-taking for assistants) suggests using intonation to predict if the user will continue or not [36]. While not directly *question detection*, these paralinguistic cues often correlate: a rising tone at end could mean it's a question *and* that the user expects a response, so the system should answer rather than just acknowledge.

- **Imperative Tone:** Detecting an imperative via audio alone is less straightforward than question intonation. There isn't a single universal prosodic pattern for commands – it depends on context and emotion. Some commands might be barked with high pitch and energy; others, especially polite requests, might be soft. However, certain **phrasing** in audio can hint at an imperative:

- **Emphasis on action verb:** e.g. *"Turn left at the next corner"* might have stress on *"turn"* and *"left"*.
- **Falling, firm intonation:** Many directives in English end with a falling pitch (a definitive tone), more so than statements which might trail off. An abrupt drop at the end could indicate a completed command.

- **Prosodic context:** If a user's utterance is a single word or fragment spoken with a commanding tone (e.g. *"Off."* said curtly could mean "Turn it off"), a combination of short duration, sharp cutoff, and possibly higher volume can signal imperative intent.

- **Utilizing Prosodic Classifiers:** Tools like **openSMILE** (an open-source feature extractor for paralinguistics) can compute hundreds of features (pitch stats, energy, voice quality measures) from audio. These can feed into an ML classifier to augment text-based features. Some specialized studies have trained SVMs or deep models purely on prosodic features to classify utterances as question vs statement, achieving moderate success (prosody alone might not distinguish a wh-question from a statement well, since both can have falling tone). But combining prosody with words yields the best results [13]. In practice, a voice assistant could incorporate a lightweight prosody model: e.g., measure final pitch slope and if the text model is unsure (perhaps the text ended with a period but pitch was rising), override or adjust the classification.

- **Example (Integration):** Microsoft's Cognitive Services once demonstrated "intent recognition" that factored in voice tone. If you said *"Play some music."* vs *"Play some music?"* with the respective intonations, the system should treat the first as a command to start playback, and the second as maybe confusion or a question about music. By analyzing the audio pitch, the system can correctly differentiate these, even if ASR returns the same words.

In summary, speech introduces **additional layers of information**. Prosodic features (pitch, duration, pauses, energy) act as **biological punctuation** – they convey the structure and intent that punctuation and syntax do in text [37]. Effective question/imperative detection in voice assistants leverages these cues alongside the transcript. Many modern systems use a **multi-modal approach**, feeding both text and acoustic features into their models to improve accuracy in ambiguous cases.

5. Benchmarks, Datasets, and Evaluation

Research into question and imperative detection has benefited from several datasets and benchmarks: - **Switchboard Dialogue Act Corpus (SwDA):** This is a classic labeled dataset derived from the Switchboard conversational telephone speech corpus [38]. SwDA uses the DAMSL tagset (42 tags after clustering) [39], which includes multiple *question* categories (Yes-No-Question, Wh-Question, Or-Question, etc.) and a *Directive* category (commands/request actions). Each utterance in a telephone conversation is labeled. For example, an utterance like *“So do you go to college right now?”* is tagged as **Yes-No Question** [7]. This corpus has been the basis for many dialog act classification studies. Models are evaluated typically by accuracy or F1-score across all 42 classes, or sometimes focusing on a binary question vs. non-question task. Because the classes are imbalanced (questions are fewer than statements), metrics like F1 or AUC are often used for specific classes. State-of-the-art models (transformer-based) on SwDA achieve ~82–83% overall accuracy [24], and question detection within that is even higher (Yes-No questions can be identified with >90% F1 in some models, since lexical cues are strong).

- **ICSI Meeting Recorder Dialog Act (MRDA) Corpus:** This corpus contains multi-party meeting transcripts with dialogue act labels [40]. The tagset is a slight variant of DAMSL, tailored to meeting conversations (with tags like Statement, Question, Action Item etc.). It also has classes for *Backchannels* and others. Margolis & Ostendorf (2011) specifically used MRDA to evaluate question detection [41]. In MRDA, question detection is a bit more challenging because meetings include more informal and fragmentary speech (leading to things like declarative questions and trailing-off questions). Nevertheless, with adaptation, their text+prosody model performed well (AUC ~0.95, from their description). Modern models have achieved over 90% accuracy on multi-class DA tagging for MRDA [22].

- **AMI Meeting Corpus:** Another meeting dataset (with audio/video) that has annotations for dialogue acts and topic segments. It’s similar in spirit to MRDA. Dialog act classification experiments have been done on AMI as well. It provides realistic data for evaluating how a model handles more spontaneous speech phenomena.

- **VerbMobil and MapTask:** Older corpora from the 90s, annotated with speech acts. VerbMobil (German/English appointment scheduling dialogs) and MapTask (a direction-giving dialogue corpus) have acts including questions and instructions. These are not as commonly used today but were important in early research. They can be useful for evaluating cross-domain performance (e.g. a model trained on Switchboard might be tested on MapTask to see if it still correctly spots commands like “Go north.”).

- **Emerging Corpora for Imperatives:** Recognizing a gap in resources specifically for imperative sentences, recent work has created datasets:

- **TVS (TV Series) and Wikipedia “Articles for Deletion” Imperative Corpus:** A 2020 study (Chen et al. 2020, LREC) built a corpus of imperatives by taking scripts from *The Big Bang Theory* (spoken dialogue) and discussions from Wikipedia’s AfD forums [42]. They annotated each utterance as imperative or not, following a pragmatic definition (i.e. does it function as a directive) [43]. They also devised some automatic rules to classify imperatives and evaluated them against the human annotations. This corpus revealed a range of imperative forms – from blunt commands to suggestions and requests in question form. The authors reported that a simple rule-based classifier (verb-first, etc.) had **low precision and recall on spoken data** [44] [10], underscoring the need for more flexible models. Specifically, the rule had many false positives in dialogue due to sentences like “See you” or “Thank you” (mis-tagged as imperatives) and couldn’t catch polite requests like *“Would you like to join me…?”* [11]. This corpus provides a new benchmark for imperative detection in both written and spoken English.

- **Email and Forum Datasets:** As mentioned, email threads have been labeled for question/command identification. The Enron email corpus was annotated for “actionable content” in some studies, marking requests (imperatives) vs. questions vs. other. Also, Stack Exchange or forums data have been used to identify if a post contains a question that needs answering (which is a kind of question detection). There was work on **identifying answer-seeking questions in online forums and Twitter** [45] – e.g. distinguishing a post that is genuinely asking a question from one just making a statement.

- **Speech Prosody Data:** Some datasets specifically target prosody:

- The **Boston University Radio Speech Corpus** and similar have intonational annotations (ToBI labels) for each sentence. One could derive a question detection task from those by seeing which sentences have a question intonation (marked by a final high boundary tone). However, these are read speech, not interactive dialogue.

- The **ComParE Challenges** (Computational Paralinguistics Challenges at Interspeech) occasionally include related tasks. While none have been exactly question/imperative detection, they’ve included things like detecting sentence modality or emotion, which involve similar prosodic analysis. These provide feature sets and benchmarks for audio classification that could be repurposed.

- **Evaluation Metrics:** For text-based detection, **accuracy** and **F1-score** (especially class-wise F1 for “question” or “imperative” class) are standard. In imbalanced scenarios (few imperatives among many statements, for instance), precision and recall for the minority class are monitored. A system embedded in an assistant might also be evaluated on *impact*: e.g., how often it correctly routes a question to the QA module or a command to the action executor. In speech, if punctuation restoration is the proxy task, metrics like per-token precision/recall for inserting `?` are used [46]. For example, a model might get an F1 of 0.95 for period insertion but only 0.85 for question mark insertion, reflecting that questions are fewer and sometimes harder to detect. Some evaluations use **AUC (Area Under ROC Curve)** for question detection [47], treating it as a binary detection problem. This is useful when one might adjust a threshold (e.g. a model outputs a probability of questionhood, and you choose a cutoff).

- **Benchmark Challenges:** There hasn’t been a specific shared task solely on question/imperative detection, but it often appears as a subtask in dialog system challenges. The Dialog State Tracking Challenges (DSTC) and others implicitly require identifying if the user asked something or requested an action as part of understanding the user’s act. The performance on those is wrapped into overall task success, but the underlying models can be analyzed for their question/command detection accuracy.

In summary, a variety of datasets from telephone conversations to meetings to emails are available to train and evaluate these detectors. The best models today leverage large corpora (like Switchboard) for general patterns and then fine-tune on domain-specific data (like an in-house dataset of actual voice assistant queries, which might have different distributions – e.g. far more commands like "play music" and factual questions). Cross-domain evaluation is important: a model trained on Switchboard might label everything in a voice assistant log as either question or statement, because Switchboard has almost no direct imperatives (people weren't giving each other commands on the phone). So one must ensure the training data covers the full spectrum or use multi-genre corpora. The availability of corpora like the Big Bang Theory/Wikipedia imperative corpus [42] is closing this gap by providing examples of directive utterances.

6. Available Libraries and APIs

Developers have access to various NLP libraries and APIs that facilitate question and imperative detection:

- **SpaCy:** A popular Python NLP library with fast tokenization, POS tagging, and dependency parsing. SpaCy does not explicitly label sentences as interrogative or imperative out-of-the-box, but its parsed outputs make it easy to implement rule-based detectors. For instance, using spaCy one can check:
- `doc[-1].text` for a `"?"` token at the end (question mark).
- `doc[0].pos_` to see if the first token is a verb and `doc[0].dep_ == "ROOT"` (meaning the sentence starts with a main verb) and look for absence of a subject dependency [2]. A spaCy answer on StackOverflow suggests exactly these clues for imperatives: *"The verb is the ROOT; No explicit subject (no nsubj dependency); The sentence starts with a verb; Exclamation marks increase the likelihood."* [2]. This logic can be coded in a few lines with spaCy.

- SpaCy's dependency parse can also reveal auxiliary-fronted questions: e.g. find if any auxiliary verb has dependency `aux` and comes before the subject. Alternatively, one can use spaCy's rule-based *Matcher* to find patterns (like [{"LOWER": {"IN": ["what","how","could","would","do","are","is"]}}, {"IS_PUNCT": false, "OP": "*"}] etc. followed by a `?`).

- **NLTK (Natural Language Toolkit):** NLTK provides tools for regex-based chunk parsing and grammar parsing. As we saw, one can use NLTK's POS tagger and then write a RegexpParser grammar to catch imperatives [9]. NLTK also has a simple function to detect a question mark at end of sentence (one could combine this with other checks). It's more of a toolkit than a ready-made solution, but for academic or prototype purposes, NLTK is very handy to implement the heuristic approaches mentioned.

- **AllenNLP:** An open-source library by the Allen Institute for AI, focused on deep learning for NLP. While AllenNLP doesn't have a pre-built "question detector" predictor, it offers easy interfaces to train custom classification models. For example, one could use AllenNLP to quickly define a dataset reader for utterances labeled as question/imperative and train a transformer classifier. AllenNLP's model zoo and guides include dialogue act classification examples. If someone wanted to reproduce a state-of-the-art approach from research (like a hierarchical LSTM with attention), AllenNLP's modular components make that easier. It also has pre-trained models for related tasks (e.g. semantic role labeling) which indirectly might help (for instance, SRL could tell you if a sentence has an implied “(You) [V]...” structure indicating imperative).

- **Hugging Face Transformers and Model Hub:** This has arguably become the go-to resource for NLP models:

- The **Transformers library** allows one to fine-tune BERT, RoBERTa, etc., on a classification task with just a few lines of code (using the Trainer API or pipelines). You can formulate the problem as a text classification with labels {question, imperative, other}.
- The **HuggingFace Model Hub** already contains models that are relevant:
    - *Punctuation restoration models:* e.g. models like `felflare/bert-restore-punctuation` or `vaiokk/chinese-bert-punctuation` etc., which are trained to insert question marks [48]. NVIDIA’s NeMo toolkit also provides a pretrained punctuation model for English that predicts commas, periods, and question marks [49]. These can be repurposed to just detect if a question mark should be present (which equates to question detection). Using such a model is as simple as feeding it the text and seeing if it outputs a `?` at the end.
    - *Dialog act models:* there may be community models fine-tuned on Switchboard or other corpora, capable of labeling utterances. For example, one could find a model card for a BERT fine-tuned on SwDA. The categories would include question types and command (directive). By mapping those, you essentially get a question/imperative classifier. A quick search might find a model like `stanfordnlp/Switchboard-DAMSL-BERT` or similar (for illustration).
    - *Zero-shot classification:* If an out-of-the-box model is needed without fine-tuning, one could use HuggingFace’s zero-shot pipeline with a prompt like: *“This sentence is a {question, command, statement}.”* and see which is most likely. Models like `facebook/bart-large-mnli` can do zero-shot intent classification. While not as accurate as a fine-tuned model, they can capture obvious cases (likely identifying the entailment of “question” if a `?` is present, etc.).

HuggingFace also hosts datasets – e.g., one could retrieve the Switchboard or MRDA dataset via the `datasets` library to train models.

- **Speech-to-Text APIs with Punctuation:** The big cloud providers (Google, Amazon, Microsoft, IBM) all offer speech recognition that returns punctuated transcripts. Developers using these APIs indirectly get question detection “for free” because when the user asks something, the transcribed text will contain a `?` if the model inferred it. For instance, Google’s API is known to insert question marks when the audio’s intonation and wording suggest a question. These services train their models on enormous amounts of conversational data, so they have learned patterns to some extent. However, they are not 100% reliable – sometimes a question may be transcribed with a period or no punctuation (especially if intonation is flat or the question is indirect). As a backup, or if using an ASR that doesn’t do punctuation, one can apply a post-processing model. **Open source ASR** frameworks like **Kaldi, Wav2Vec 2.0, Mozilla DeepSpeech, Coqui STT, Whisper** often output raw text (or minimal punctuation). For these, one can integrate a punctuation model such as:
- **NVIDIA NeMo Punctuation Model:** which can be run on the transcript to insert commas, periods, question marks [49].
- **Silero/Stanford Punctuator:** There are pre-trained models like Silero’s punctuation or older ones (Stanford’s Punctuator from 2016) that you can use. These typically use bi-directional LSTMs or transformers.

- **Whisper by OpenAI** – though primarily an ASR model, it actually does output punctuation and even line breaks. If using Whisper for transcription, it might place a `?` in the output. Whisper's internal transformer decoder has effectively learned question vs statement punctuation from its training on web data. So in an offline setting, Whisper could be both the ASR and the question detector.

- **Linguistic Tools:** Tools like **Praat** (phonetics software) or the **openSMILE** toolkit can be used if you want to do a custom acoustic-prosodic analysis. For example, Praat scripting can extract pitch contours and detect rising vs falling tone (Praat can even apply a t-test to see if final pitch > initial pitch significantly, etc.). openSMILE can extract features and then you could feed them into an SVM to classify if an utterance's prosody is question-like. These require more expertise and are more often used in research than production. In production, it's more common to rely on the ASR's learned model.

- **Dialog System Platforms:** High-level frameworks like **Rasa (Open Source)** or **Google Dialogflow / Microsoft LUIS / Amazon Lex** focus on intent detection, but they allow writing rules or small models to catch question forms. For instance, Rasa's training data could include an *intent* like `ask_question` vs `give_command` and you could provide example sentences. Under the hood, Rasa uses either spaCy features or transformers (Diet classifier) to classify intents, so it would similarly learn question patterns if you structure your intents that way. Dialogflow has a concept of *system entities*, but not specifically "question or not". However, one can use context: e.g. if an input ends with a question mark, one might route it differently. Many voice assistants (Alexa, Google Assistant) abstract this by just letting you define what intents you expect – they do not explicitly give you a question/command flag, but the way you design intents can capture it (like an FAQ intent for questions vs action intents for commands).

- **GitHub Repositories:** There are also specialized GitHub projects. For example:

- The repository **saksham-jain177/task_extraction** on GitHub outlines an approach for imperative detection and "to-do" extraction, using rules such as *"sentence begins with base-form verb"* and looking for task indicators [50]. It's not a library per se, but an example implementation in code that others can adapt.
- Research code from papers: many dialog act papers release their training code (e.g. a BERT model fine-tuned on Switchboard). One could adapt that to the specific question vs imperative distinction.

In summary, **developers have two routes**: use *existing tools with minor configuration* (like spaCy or an ASR API that already inserts punctuation), or *train a custom model* (using libraries like HuggingFace or AllenNLP) if higher accuracy is needed or if the domain has particular quirks. For many voice assistant applications, relying on the ASR's punctuation + a bit of heuristic post-check (like ensure the presence of a `?` aligns with certain keywords) is a practical starting point. For advanced systems, a dedicated classification model can be deployed as part of the NLU pipeline (taking the transcript text and outputting a label).

7. Limitations and Mitigation Strategies

Despite sophisticated methods, detecting questions and imperatives is not foolproof. It’s important to be aware of limitations:

- **Ambiguity and Indirect Speech Acts:** Not all questions look like questions, and not all commands look like commands. English speakers often use **indirect speech acts**:
- *“Could you close the window?”* – Formally interrogative (yes/no question form), but pragmatically a request/command. A system might misclassify it as a question and attempt to answer “Yes, I could” rather than performing the action. Mitigation: incorporate semantic or pragmatic context. For a voice assistant, usually any *“could you...?”* directed at it should be interpreted as a request. This can be handled by intent classification (mapping such phrasings to an intent like `CloseWindow`). But in pure grammatical detection, it’s a challenge. One mitigation is to use additional cues like the presence of an action verb and second person pronoun “you” – one paper noted that questions starting with “Do you” or “Have you” were a common source of false positives for imperative rules [51] (since they look like questions but sometimes they were requests in context). Special handling of these forms can improve interpretation.
- Rhetorical questions: *“Why bother?”* – It has a question form, but the speaker isn’t actually asking for an answer; it’s more of a sarcastic statement. A classifier might label it as a question (it is grammatically one). In a voice assistant scenario, the system might futilely attempt to answer it. Distinguishing rhetorical questions from genuine ones is extremely hard without deeper understanding or prosodic cues (rhetorical ones might have a different intonation sometimes). Currently, most systems will just treat them as questions; mitigation might rely on user prompting or ignoring certain patterns known to be rhetorical.

- Politeness variations: *“I was wondering if you could tell me the time.”* – Entirely a declarative sentence, yet it’s a polite question. Pure rule-based or surface-trained models can miss this, labeling it as a statement. Context (the assistant’s role) might help: if user says anything about “tell me the time,” it’s likely a request for time. In an NLU pipeline, the presence of phrases like “tell me” or “I wonder” could trigger a question interpretation even if not structured as one. More advanced: large language models have enough knowledge to decipher that as a question, so using their embeddings or few-shot prompting could catch such cases.

- **ASR Errors:** In spoken input, the speech recognizer can make transcription errors that obscure question/imperative cues. For example:

- Misrecognized words: *“Can you play music”* might be transcribed as “You play music.” Missing the initial “Can” changes the structure from a question to what looks like an imperative or statement. A detector might then misclassify. Mitigation: using confidence scores or alternate hypotheses. If the ASR is unsure, it might provide “Can you play music” as an alternate. The system could verify if adding a “can” makes a difference in interpretation. Prosody might also reveal that it was a question (rising tone on “music?”).
- Missing punctuation: If the ASR doesn’t insert a question mark where it should, a downstream module might assume it’s a statement. Mitigation: always run a punctuation restoration model on raw ASR output if the ASR isn’t reliable on that front. There are open models that can do this with reasonable accuracy.

- Segmenting issues: Sometimes whether something is a question depends on where you consider the sentence boundary. Speech can run on, and an ASR might put two sentences together or split incorrectly. E.g. *“I have two questions. Do you like music and do you sing?”* If ASR missed the period, it could come out as “I have two questions do you like music and do you sing." A detector might struggle to see the two questions inside. Mitigation: better utterance segmentation (using silence or language cues) before applying detection.

- **Prosody Variability:** While prosody is helpful, it's not consistent:

- Not all yes-no questions have a strong rising tone (some can be flat or even falling if the speaker is confident or using a certain dialect).
- Some statements in certain dialects (e.g. Uptalk) *do* have rising ends even when they aren't questions, which could confuse an intonation-based classifier.

- Imperatives can be shouted (high pitch) or spoken softly. So, relying on an "imperative tone" classifier could yield false signals. Mitigation: use prosody as a supportive feature, not sole criterion. Many systems give more weight to lexical cues and use prosody as a secondary check.

- **Edge Cases in Grammar:** English allows some tricky forms:

- **Imperative questions:** e.g. the suggestion form *"Let's go, shall we?"* – The main clause "Let's go" is imperative (first-person inclusive), followed by a tag "shall we?" which is a question. How to classify this whole utterance? It's both a proposal and has a question nuance. Likely one would classify it as a proposal/command. A model might get confused if it focuses on the "?" at the end. Mitigation: perhaps treat tag questions as separate or give precedence to the main clause mood.
- **Yes as a question:** Simply saying *"Yes?"* with rising intonation – often means "What do you want?" or "Can I help you?" when someone calls you. The word "yes" alone normally is an answer, but with questioning intonation it's effectively a question (*"Yes?" = "What do you need?"*). Classifiers based purely on text would call it an answer/statement, missing the rising tone. Only prosody or context reveals it's a question. These one-word or very short utterances are extremely ambiguous without audio or conversational context. Mitigation: incorporate dialogue context (e.g. if the system just called the user's name and the user says "Yes?" back, it's obviously a response, not an affirmative answer).

- **Commands phrased as statements:** e.g. *"You will now drop your weapons."* Grammatically declarative (subject + future tense verb), but contextually it's an imperative (often used as a firm command). Without context or perhaps a very authoritative tone, a model might label it as a statement of fact about the future. Mitigation: This ventures into understanding intent and modality. Possibly certain modal verb usages ("you will X" as a directive) could be flagged via rules or recognized if the training data included such patterns (like in fiction or dialogues).

- **Domain Shift:** A model trained on one type of data may not generalize:

- If a classifier was trained mostly on questions about factual info (who/what/when queries), it might misidentify a very polite or convoluted request. Or a model trained on casual speech might fail on a very formal question.
- The vocabulary and style differences require careful use of training data. Mitigation: include diverse training examples. Also, employing **unsupervised adaptation** techniques as discussed (like Margolis did with unlabeled data) can help adjust to a new domain (e.g. adapting from written web text to actual transcribed user queries).

- Continuous learning: In a live assistant, one can collect cases where the system misinterpreted and then add those to training. For example, if users often say "Tell me about X" and initially the system didn't treat that as a question, that can be corrected by training the model that such patterns are information requests.

- **Evaluation Difficulties:** Deciding ground truth isn't always straightforward for imperatives vs questions. Human annotators sometimes disagree on whether something is a question or a statement (especially for indirect forms). For example, *"I'd like to know if the store is open."* Some might label that as a statement (since it doesn't invert or use `?`), others as a question (since it clearly is a request for info). This ambiguity in labels can limit model performance (the model may actually capture the ambiguity and be "uncertain"). Mitigation: use consistent annotation guidelines and possibly allow a category for "ambiguous" or handle such cases upstream (maybe by rephrasing or confirmation).

Mitigation Strategies Summary: - **Hybrid Approaches:** The most robust systems often combine rules and ML. A rule-based filter might handle extremely clear cases (high precision rules) and a statistical model handles the rest. For instance, if a sentence ends in `?`, you almost certainly label it a question (to avoid the model overthinking it). Conversely, a sentence starting with "Please" and a verb might be auto-labeled imperative. The ML model deals with nuance where these simple cues conflict. - **Confidence and Fallbacks:** If a classifier isn't confident (e.g. probability between classes is low margin), the system can either: - Use prosody or context as a tiebreaker. - Ask a clarification if appropriate (though in voice assistants this is last resort: "Did you mean to ask a question or give a command?" – not a great user experience, so usually avoided). - Default to a safe interpretation. Some assistants might choose to treat uncertain cases as questions and provide some info, or as commands and do nothing unless certain. Choosing the default depends on the application (for instance, a smart speaker might default to answering, whereas a robot might default to not executing an unclear command). - **Continuous Improvement:** Monitor errors where the assistant responded inappropriately (answered a command or executed when asked something trivial), and update the model or add rules to handle those patterns. - **Leverage Context:** If the system has dialogue context or user profile, use it. E.g., if the previous system action was asking the user *"Do you have any other questions?"*, then even if the user's next utterance is not clearly phrased as a question, it might well be one (they are expected to ask something). Similarly, if the user just gave a command and the next utterance is "and the weather?", it's fragmentary but likely means *"and [tell me] the weather?"* – essentially a question. Context-aware models can decipher these follow-ups better.

- **Multilingual Considerations:** (Out of scope here, but just to note) If expanding beyond English, each language has its own cues for questions and commands (e.g. syntax vs particles vs intonation). A technique working in English might not directly apply elsewhere, so detection components often have to be language-specific or require new training data in the target language.

Finally, we note that "question vs imperative" can sometimes be a simplification. In designing a conversational AI, one might actually classify user utterances into a richer set of **intents or dialog acts**: e.g. *Information Request*, *Action Request*, *Greeting, Feedback, etc.*. This holistic approach inherently covers questions and commands. For technical development, however, focusing on the binary or pairwise detection (question vs command) with the techniques above is a solid foundation, and can be expanded as needed.

References: The insights and techniques discussed are drawn from a range of studies and resources, including dialogue act tagging research [52] [53], prosody and punctuation studies [37] [35], and practical solutions shared in the NLP community [1] [3]. These sources illustrate the progression from rule-based methods to modern deep learning and the importance of combining lexical and acoustic information for robust detection. Each method contributes to handling the rich variability of how English speakers ask questions and issue commands in natural conversation.

[1] [2] [4] python 3.x - Extracting English imperative mood from verb tags with spaCy - Stack Overflow
https://stackoverflow.com/questions/59008046/extracting-english-imperative-mood-from-verb-tags-with-spacy

[3] [5] [6] [9] nlp - Approach for identifying whether a sentence includes an imperative within it - Stack Overflow
https://stackoverflow.com/questions/29473169/approach-for-identifying-whether-a-sentence-includes-an-imperative-within-it

[7] [21] [22] [23] [24] [28] [38] [39] [40] Dialogue | NLP-progress
http://nlpprogress.com/english/dialogue.html

[8] stanford nlp - Detecting questions using Parsey's Dependency parser - Stack Overflow
https://stackoverflow.com/questions/44420055/detecting-questions-using-parseys-dependency-parser

[10] [11] [20] [42] [43] [44] [51] TV-AfD: An Imperative-Annotated Corpus from The Big Bang Theory and Wikipedia’s Articles for Deletion Discussions
https://aclanthology.org/2020.lrec-1.805.pdf

[12] [13] [14] [15] [16] [17] [18] [19] [41] [47] [52] [53] Question Detection in Spoken Conversations Using Textual Conversations
https://aclanthology.org/P11-2021.pdf

[25] [PDF] Dialog Acts Classification for Question-Answer Corpora
https://ceur-ws.org/Vol-2385/paper6.pdf

[26] [29] [30] [31] [32] [37] isca-archive.org
https://www.isca-archive.org/interspeech_2022/cho22b_interspeech.pdf

[27] [PDF] A Deep Multi-task Model for Dialogue Act Classification, Intent ...
https://www.cse.iitb.ac.in/~pb/papers/cognitive-computation20-dac.pdf

[33] [34] cstr.ed.ac.uk
https://www.cstr.ed.ac.uk/downloads/publications/1998/Shriberg_1998_a.pdf

[35] (PDF) Detecting question-bearing turns in spoken tutorial dialogues
https://www.researchgate.net/publication/221483044_Detecting_question-bearing_turns_in_spoken_tutorial_dialogues

[36] Is It My Turn Yet? Teaching a Voice Assistant When to Speak
https://hai.stanford.edu/news/it-my-turn-yet-teaching-voice-assistant-when-speak

[45] Automatic classification of product reviews into interrogative and non-interrogative: Generating real time answer
http://www.science-gate.com/IJAAS/2019/V6I8/1021833ijaas201908004.html

[46] clarin-pl/2021-punctuation-restoration · Datasets at Hugging Face
https://huggingface.co/datasets/clarin-pl/2021-punctuation-restoration

[48] felflare/bert-restore-punctuation - Hugging Face
https://huggingface.co/felflare/bert-restore-punctuation

[49] Punctuation and Capitalization Model - NVIDIA Documentation
https://docs.nvidia.com/nemo-framework/user-guide/24.12/nemotoolkit/nlp/punctuation_and_capitalization.html

[50] saksham-jain177/task_extraction - GitHub
https://github.com/saksham-jain177/task_extraction