# Rule-First Transaction Extraction from Indian Bank SMS with a Small Language-Model Fallback: An Empirical Study

Rushi Daulatkar[1], Abhijeet Pathode[2], Atharva Dhote[3], Harsh Yadav[4], Rushikesh M. Shete[5]
[1,2,3,4]*Department of Computer Science and Business Systems, St. Vincent Pallotti College of Engineering and Technology, Nagpur, Maharashtra, India*
[5]*Assistant Professor, Department of Computer Science and Business Systems, St. Vincent Pallotti College of Engineering and Technology, Nagpur, Maharashtra, India*

***Abstract*—Banks in India must send short message service (SMS) alerts for electronic transactions, which makes these alerts a natural input for automatic expense tracking, yet pipelines that turn them into categorized transactions are rarely evaluated on real message streams. This paper evaluates the pipeline of a self-hosted personal-finance prototype that extracts transactions with ordered regular-expression rules and categorizes them through user corrections, a learned merchant table, keyword lists and, as a last resort, a 4-bit quantized 3-billion-parameter language model. On 1,153 bank-sender messages from one consenting author's handset, collected over 31 months and dominated by one bank's formats, transaction detection reached a precision of 96.4% and a recall of 99.9% (F1 = 0.981); amounts were exact for all 991 detected transactions and merchant names for 98.3% of scorable ones. All 37 false detections were money-related messages that did not record a completed transaction. In a cold-start replay, 19.7% of messages invoked the language model, yet 81.9% received a category originating from a model answer, directly or through caching. Under default sampling, 30.2% of model inputs received the same label in three runs. Against the account holder's merchant labels, end-to-end category accuracy was 20.3–28.3%, below a majority-class baseline of 31.5%, while keyword-derived categories were 66.3% correct. In this dataset, rule-based extraction performed well, whereas model-based categorization was unstable and inaccurate; deterministic decoding, output validation and user feedback are suggested remedies that remain to be evaluated.**

***Index Terms*—Edge inference, Personal finance management, Rule-based parsing, small language models, SMS information extraction, Transaction categorisation**

## I. INTRODUCTION

Banks in India are required to send SMS alerts to customers for electronic banking transactions [1], and India's unified payments interface (UPI) alone processed about 24 billion transactions in September 2026 [2]. An account holder's inbox therefore holds a running record of debits and credits, which makes it an attractive input for automatic expense tracking that needs neither manual entry nor access to bank application programming interfaces. Turning the alerts into useful records involves two distinct tasks. Extraction decides whether a message reports a completed transaction and recovers its direction, amount and counterparty. Categorization assigns the transaction to a spending category such as food or transport.

The two tasks have attracted different techniques. Rule-based information extraction remains widely used in practice because rules are transparent and easy to correct, even though research attention has moved to learned models [3]. Categorizing short transaction descriptions has been studied with support vector machines [4], deep neural networks [5], zero-shot prompting of large language models (LLMs) [6], [7] and fine-tuned financial language models [8]. Small, quantized LLMs can run on edge devices and commodity processors [9], [10], which makes it possible to host a model-based fallback on one's own server instead of sending financial text to a third-party service; users of finance applications report substantial privacy concerns [11]. Cascades that route only some

inputs to a model have been proposed to reduce inference cost [12].

We found little published evidence on how such a hybrid behaves on a real, long-running message stream. This study addresses four questions: how accurate hand-written extraction rules are when the stream includes money-related messages that are not transactions; how often a model fallback is invoked once a learned merchant table is in place; how stable the fallback's answers are, given that LLM outputs can vary under sampling [13], [14] and that reusing a model's own uncertain labels can entrench errors [15]; and how accurate the resulting categories are.

This paper reports a preliminary empirical evaluation of the pipeline of a self-hosted personal-finance prototype on 1,153 real bank-sender messages. Its contributions are: (i) an evaluation of rule-based extraction against a template-based reference annotation, with an error taxonomy; (ii) a chronological, cold-start replay that measures how categorization work is routed across layers; (iii) a repeated-run analysis of the quantized-model fallback and of how caching spreads its answers; and (iv) a measurement of end-to-end category accuracy against the account holder's own labels. The study is deliberately narrow, with one user, one dominant bank and one model, and is intended to inform design choices rather than to establish general performance.

## II. MATERIALS AND METHODS

### A. System Under Evaluation

Fig. 1 summaries the pipeline. In the deployed configuration, the handset parses each bank message with a Kotlin port of the parser and sends only the extracted fields to the server, which does not store message text. A message from a bank sender identifier is first tested for direction: a debit pattern (debited, debit not followed by card, withdrawn, spent, paid) is checked before a credit pattern (credited, credit, received, deposited), with word boundaries. A message with no direction is discarded. The amount is taken from the first currency-prefixed number (Rs, INR, EUR or USD), and the counterparty from the first of ten ordered merchant patterns that matches, for example a UPI handle, the text between a semicolon and credited, or the name after towards.

Categorization runs on the server using the merchant string only. The merchant's name is normalized (lower case, punctuation and legal suffixes removed) and passed through four layers in order: L1, the user's own corrections; L2, a merchant table shared across accounts, matched exactly or by substring for names of five or more characters;

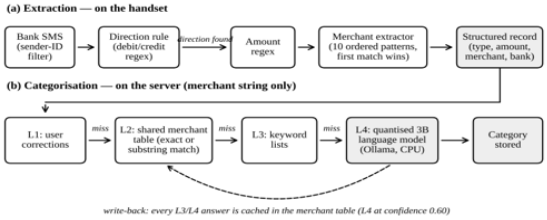


Fig. 1. Two-stage pipeline of the evaluated prototype: rule-based extraction on the handset and layered categorization on the server

L3, keyword lists for eight base categories (food, shopping, transport, bills, transfer, entertainment, health and other); and L4, a language model. Every L3 or L4 answer is written back to the merchant table with a confidence of 0.85 or 0.60 respectively, and table entries with a confidence of at least 0.60 are reused by L2. A model answer is therefore applied to later transactions with the same merchant.

L4 is Llama 3.2 3B Instruct [9], quantized to 4 bits (Q4_K_M, llama.cpp format [10]) and served with Ollama [16] on a CPU-only machine. The prompt asks for a one-word reply from the eight categories, lists typical keywords for seven of them and supplies the merchant string. The context window is limited to 256 tokens and no sampling temperature is set, so the runtime default applies. The first word of the reply is kept if it is a valid category; any other reply is mapped to other.

### B. Corpus

The corpus consists of every SMS in one author's inbox whose sender identifier contained the code of one of five banks, read with the Android debug bridge on 11 October 2026. Only bank-sender messages were exported; other messages were not written to disk. The messages span 13 March 2024 to 9 October 2026 and come from 56 sender identifiers belonging to 12 institution codes, although one bank's transaction-alert code accounts for 91.8% of them (Table I). Because the messages contain third-party names and account fragments, the corpus is not released. This paper quotes only short fixed phrases of the banks'

wording and reproduces no message, name or account detail.

Data and ethics. The study was conducted using one author's own personal financial SMS data, from that author's personal handset, with the author's consent. No other participants were recruited, and no third-party names, account details or message texts are reported. The authors' project supervisor confirmed that formal institutional ethics approval was not required for this study.

Table I. Composition of the SMS Corpus by Reference Class

| Class (reference annotation) | Templates | Messages |
|---|---|---|
| Completed transaction – debit | 6 | 728 |
| Completed transaction – credit | 4 | 264 |
| Promotion or service advertisement | 1 | 39 |
| Mutual-fund notice (SIP allotment, statement, fee change) | 1 | 31 |
| One-time password or authentication code | 1 | 22 |
| AutoPay mandate created or revoked | 1 | 18 |
| Advance notice of a future debit | 1 | 13 |
| Payment (collect) request awaiting approval | 1 | 10 |
| Declined payments; account, card, registration and corporate notices | 4 | 26 |
| Overseas transfer status | 1 | 2 |
| Total | 21 | 1,153 |

C. Reference Annotation

A completed transaction is a message reporting that money has already left or entered the account; notices of future, conditional, declined or pending payments are not transactions. Messages were grouped by shape, and every distinct shape was read and assigned to one of 21 templates: 10 for completed transactions and 11 for other messages. Each transaction template is a full-sentence regular expression anchored on the bank's wording and records the true direction, amount and counterparty position. The templates were written separately from the parser and are stricter. All 1,153 messages matched exactly one template group, 992 as transactions and 161 as non-transactions. Four transactions (two reversals, a mobile-number transfer and an overseas transfer) carry no counterparty name and are excluded from merchant scoring, leaving 988 scorable transactions.

D. Experiment 1: Extraction

The extraction function was loaded unchanged from the server source and applied to every message. A message counts as detected when a direction is found, the condition under which the handset forwards it. A true positive (TP) is a detected completed transaction, a false positive (FP) a detected non-transaction and a false negative (FN) an undetected completed transaction. Detection is scored with

$$P = TP / (TP + FP),\ R = TP / (TP + FN), \quad (1)$$

$$F_1 = 2PR / (P + R). \quad (2)$$

For detected transactions, direction and amount are compared with the reference (amount within 0.005), and the merchant is compared after lower-casing and removing non-alphanumeric characters. A match is exact when the strings are equal and partial when one contains the other. Per-message parsing time was measured with Python's high-resolution performance counter on an x86-64 cloud virtual machine (2 vCPU, Python 3.13), not on the handset.

E. Experiment 2: Cold-Start Routing Replay

All 1,028 detected messages, including the 37 false detections that the handset would forward, were replayed in chronological order through the deployed L1–L3 functions against an empty merchant table and an empty correction table. When a message reached L4, the call was recorded and the merchant was stored with confidence 0.60, as in production, so that later messages were routed exactly as they would be in production; routing does not depend on which label the model returns. For each message we record the resolving layer and whether its final category originates from a model answer, either directly or through L2 reuse of a cached model answer.

F. Experiment 3: Behaviour of the Model Fallback

The 202 inputs that reached L4 in the replay (193 distinct merchant strings and 9 messages with no merchant string) were sent to the same model with the production prompt, context limit and post-processing, three times each (606 calls). Because the team's machines could not run the model, the runs used a hosted cloud notebook runtime with GPU acceleration; the accelerator model was not logged during the runs. Timings from these runs are not reported because they do not represent the CPU deployment. We measure the share of replies that are valid categories, the label distribution, and inter-run

agreement: the share of inputs that received the same label in all three runs and the mean share agreeing across the three pairs of runs. Agreement measures consistency, not correctness.

G. Experiment 4: Category Accuracy

The account holder labelled each normalized merchant with the category the payments were actually for, using the eight base categories and written rules (for example, payments to friends with no specific purpose are transfer; investments, dividends and stipends are other), without seeing the system's predictions. Each replayed message inherits the label of its merchant; messages with no merchant string or an unlabeled merchant are excluded. Each message's predicted category is either keyword-derived (from L3 directly, or from L2 reuse of a keyword-derived table entry) or model-derived (from L4 directly, or from L2 reuse of a model answer). For model-derived messages, the label returned for the originating input in each run of Experiment 3 is used, giving three end-to-end accuracy figures. Accuracy is weighted by messages and compared with always predicting the most frequent reference class.

## III. RESULTS AND DISCUSSION

A. Extraction Accuracy

Table II summaries Experiment 1. The parser detected 991 of the 992 transactions and raised 37 false detections, giving a precision of 96.4%, a recall of 99.9% and $F_1$ = 0.981. Direction and amount were correct for all 991 detected transactions. Merchant names were exact for 971 of the 988 scorable transactions (98.3%) and at least partially correct for 983 (99.5%). Parsing took a median of 33 µs per message (95th percentile 65 µs).

Table II. Extraction Results on 1,153 Messages (992 Transactions)

| Measure | Result |
|---|---|
| Detection precision / recall / F1 | 96.4% / 99.9% / 0.981 |
| True positives / false positives / false negatives | 991 / 37 / 1 |
| Direction corrects (991 detected transactions) | 100% |
| Amount exactly correct (991 detected transactions) | 100% |
| Merchant exact / partial / wrong / missing (988 scorable) | 971 / 12 / 2 / 3 |
| Parsing time per message, median (95th pct.), cloud VM | 33 µs (65 µs) |

Two merchant patterns resolved 92.4% of the detected transactions: the semicolon pattern (68.8%) and the from UPI pattern (23.6%). This concentration reflects the dominance of one bank's UPI alert formats in the corpus and is the main reason the results should not be generalized to other banks without a wider corpus.

B. Error Analysis

Table III lists the false detections. None came from promotions or one-time passwords: all 39 promotions and 22 authentication messages were correctly rejected. Every false detection was a money-related message that does not record a completed transaction. Advance AutoPay notices (will be debited) and payment requests (on approval, Rs X will be debited) were taken as debits because the direction rule ignores tense and condition; 12 of the 13 advance notices were followed within ten days by a completed debit of the same amount to the same payee, so those payments were counted twice. Mutual-fund confirmations were taken as credits because they contain received. In money terms, the false debits added 5.0% to the value of true debits and the false credits 4.5% to the value of true credits. The single missed transaction was an overseas transfer reported as successfully processed, a wording the direction rule does not cover.

Table III. False Detections by Message Class

| Message class | Count | Parsed as | Cause |
|---|---|---|---|
| Advance notice of future debit | 13 | Debit | Future tense not excluded |
| Mutual-fund SIP confirmation | 12 | Credit | "received" matched |
| Payment request awaiting approval | 10 | Debit | Conditional debit not excluded |
| Overseas transfer status | 2 | Credit | "credited" to the payee |
| Total | 37 | | |

Merchant errors also had identifiable causes. All 14 merchants produced by the from/to pattern were wrong. The pattern lacks a leading word boundary, so in 12 cases it matched the letters "to" inside a payee

name and truncated the name. In 2 card purchases it captured the user's own account after debited from instead of the merchant. The 3 missing merchants were a payee whose name begins with a digit, a transfer to the user's own second account and a credit carrying only a reference code. Each of these errors points to a specific rule change, such as word boundaries, exclusions for future and conditional wording, and sender-specific handling of mutual-fund messages; these changes were not implemented or tested in this study. The observation is consistent with the view that rules remain practical for extraction [3], provided the negative classes are specified as carefully as the positive ones.

C. Routing of Categorization Work

In the cold-start replay, L2 resolved 76.5% of messages, L3 3.9% and L4 19.7% (202 model calls; 244 distinct merchant strings plus 9 messages without one); L1 was never used because the replay starts without corrections (Table IV). Keywords resolved few messages, which we attribute to most payees in this stream being individuals and small traders paid through personal UPI names. Fig. 2 shows that the share of messages sent to the model fell from 37% in the first 100 messages to 8–9% in mid-2026 and rose to 20–25% in the final months, when new payees appeared. In this rather than message volume.

Table IV. Layer Resolving Each Message in the Cold-Start Replay (n = 1,028)

| Layer | Messages | Share |
|---|---|---|
| L1 user corrections | 0 | 0.0% |
| L2 merchant table: reuse of keyword-derived entries | 146 | 14.2% |
| L2 merchant table: reuse of cached model answers | 640 | 62.3% |
| L3 keyword lists | 40 | 3.9% |
| L4 language model (direct calls) | 202 | 19.7% |
| Total | 1,028 | 100% |
| Category originating from a model answer (L4 + L2 reuse) | 842 | 81.9% |

The line in Fig. 2 shows the effect of caching. Of the 786 L2 resolutions, 640 reused a label the model had produced earlier. In total, 842 messages (81.9%) received a category that originated from a model answer, compared with 202 messages (19.7%) that triggered a model call.

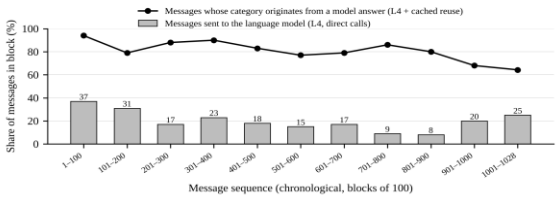


Fig. 2. Share of each 100-message block sent to the language model (bars) and share whose final category originates from a model answer (line)

The direct call rate therefore understates the share of the final data that depends on model output by a factor of about four.

D. Stability of the Model Fallback

Table V summarizes Experiment 3. Of the 606 replies, 467 (77.1%) began with a valid category; the rest began with words such as "Finance", "Transaction" or "Business" and were mapped to other, so that 47.7% of all answers ended as other. For the 9 inputs with no merchant string, the model answered "Food" in 22 of 27 calls. Only 61 of the 202 inputs (30.2%) received the same label in all three runs (56 of the 193 distinct merchant strings, 29.0%), and 36 inputs received three different labels. Messages whose category originates from an input whose label varied across runs make up 65.8% of the replayed stream, so a repeat of the same replay would be expected to change the category of many messages.

Table V. Language-Model Fallback over Three Runs (202 Inputs, 606 Calls)

| Measure | Result |
|---|---|
| Replies beginning with a valid category | 467 (77.1%) |
| Answers ending as "other": explicit / / mapped from invalid reply | 150 (24.8%) / 139 (22.9%) |
| Inputs with identical label in all 3 runs | 61 (30.2%) |
| Mean pairwise inter-run agreement | 47.5% |
| Inputs with 3 different labels | 36 (17.8%) |
| Replayed messages whose category originates from a varying input | 676 (65.8%) |

Two design choices are consistent with this instability. First, decoding is not deterministic: with the runtime's default sampling, repeated calls can return different labels, an effect known to affect single-run evaluations of LLMs [13], [14]. Second, the first answer is cached at a confidence that clears the reuse threshold, so a

single answer is copied to later transactions from that merchant, which resembles the confirmation bias reported when models are trained on their own pseudo-labels [15]. The cascade's efficiency [12] is thus also the mechanism that propagates an unreliable answer. Possible remedies, not implemented or tested here, include greedy decoding, output constrained to the category list, not caching invalid or low-agreement answers, and recording each category's source so that a later user correction can replace it.

E. Category Accuracy

The account holder labelled 212 of the 244 merchant strings, covering 898 of the 1,028 replayed messages (87.4%); 178 of these had keyword-derived and 720 model-derived categories. Table VI reports message-weighted accuracy. End-to-end accuracy was 20.3%, 21.5% and 28.3% for the three runs, below the 31.5% obtained by always predicting the most frequent reference class (transfer, 283 of 898 messages). Keyword-derived categories were 66.3% correct in every run, whereas model-derived categories were 8.9–18.9% correct, and a majority vote over the three runs did not improve end-to-end accuracy (21.3%).

Table VI. Category Accuracy against the Account Holder's Labels (898 Messages)

| Messages (n) | Run 1 | Run 2 | Run 3 |
|---|---|---|---|
| Keyword-derived category (178) | 66.3% | 66.3% | 66.3% |
| Model-derived category (720) | 8.9% | 10.4% | 18.9% |
| All evaluated messages (898) | 20.3% | 21.5% | 28.3% |
| Model answer per merchant, first L4 call (166 merchants) | 21.1% | 21.7% | 27.1% |
| Majority vote of three runs (898) | 21.3% | | |
| Majority-class baseline, always "transfer" (898) | 31.5% | | |

In run 1, most errors involved payments to individuals: only 10 of 283 transfer messages (3.5%) were categorized as transfers, and 169 were categorized as other. Payments to named people that the account holder regards as food or bills were categorized as food, shopping or entertainment. Shopping, where brand names are informative, was recognized in 48 of 56 messages (85.7%). A merchant string that is a person's name carries little information about the purpose of a payment, so categorising such payees from the string alone is unlikely to be reliable; user corrections (L1) or other signals, such as amount and periodicity, may be needed.

F. Limitations

The corpus comes from one user and is dominated by one bank's formats, so the results describe this stream rather than Indian bank SMS in general. The reference annotation was produced by the authors using templates; every template was reviewed manually, but the annotation was not independently double-checked. The replay used the server's Python implementation of the parser rather than the handset's port; an earlier internal check by the authors found no differences between Python and Java regular-expression behaviour on 933 messages. The model runs used different hardware from the deployment and only three repetitions per input, so labels obtained on the CPU deployment could differ; no comparison with a fine-tuned classifier was made. Category labels come from a single annotator, the account holder; 32 merchant strings (121 messages, 11.8% of the replay) were left unlabelled, and 9 messages had no merchant string.

## IV. CONCLUSIONS

On 1,153 real bank SMS messages from one user, ordered regular-expression rules detected transactions with a precision of 96.4% and a recall of 99.9%, with exact amounts for all detected transactions; every observed error had an identifiable cause in how non-transaction messages or payee names were handled. The layered categoriser sent one message in five to the quantised language model, yet more than four in five messages received a category originating from a model answer, and those answers were inconsistent across repeated runs. End-to-end category accuracy (20.3–28.3%) fell below a majority-class baseline, while keyword-derived categories were 66.3% correct. For pipelines of this kind, the direct call rate of a model fallback understates its influence on the stored data. The results suggest that model answers should be produced deterministically, restricted to valid labels and validated before they are cached; whether these changes improve accuracy remains to be tested. Future work will extend the corpus to more users and banks, evaluate these remedies, and compare the cascade with learned classifiers.

REFERENCES


[1]. Reserve Bank of India, "Customer protection—Limiting liability of customers in unauthorised electronic banking transactions," Circular RBI/2017-18/15, DBR.No. Leg.BC.78/09.07.005/2017-18, Jul. 6, 2017.

[2]. DD News, "UPI transactions surge 23% to 24.07 billion in September: NPCI," Oct. 1, 2026.

[3]. Chiticariu, L., Li, Y., and Reiss, F. R., "Rule-based information extraction is dead! Long live rule-based information extraction systems!" in Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Seattle, WA, USA, 2013, pp. 827–832.

[4]. García-Méndez, S., Fernández-Gavilanes, M., Juncal-Martínez, J., González-Castaño, F. J., and Seara, O. B., "Identifying banking transaction descriptions via support vector machine short-text classification based on a specialized labelled corpus," IEEE Access, vol. 8, pp. 61642–61655, 2020, doi: 10.1109/ACCESS.2020.2983584.

[5]. Kotios, D., Makridis, G., Fatouros, G., and Kyriazis, D., "Deep learning enhancing banking services: A hybrid transaction classification and cash flow prediction approach," Journal of Big Data, vol. 9, Art. no. 100, 2022, doi: 10.1186/s40537-022-00651-x.

[6]. Brown, T., et al., "Language models are few-shot learners," in Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 1877–1901.

[7]. Moraes, D. de S., et al., "Using zero-shot prompting in the automatic creation and expansion of topic taxonomies for tagging retail banking transactions," arXiv:2401.06790, 2024.

[8]. Aluffi, P. A., et al., "Categorising SME bank transactions with machine learning and synthetic data generation," arXiv:2508.05425, 2025.

[9]. Meta AI, "Llama 3.2: Revolutionizing edge AI and vision with open, customizable models," Sep. 25, 2024.

[10]. ggml-org, "llama.cpp: LLM inference in C/C++," GitHub repository, accessed Oct. 11, 2026.

[11]. Haggag, O., Pedace, A., Pan, S., and Grundy, J., "An analysis of privacy regulations and user concerns of finance mobile applications," Information and Software Technology, vol. 184, Art. no. 107756, 2025, doi: 10.1016/j.infsof.2025.107756.

[12]. Chen, L., Zaharia, M., and Zou, J., "FrugalGPT: How to use large language models while reducing cost and improving performance," Transactions on Machine Learning Research, 2024.

[13]. Song, Y., Wang, G., Li, S., and Lin, B. Y., "The good, the bad, and the greedy: Evaluation of LLMs should not ignore non-determinism," in Proceedings of the 2025 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Albuquerque, NM, USA, 2025, pp. 4195–4206, doi: 10.18653/v1/2025.naacl-long.211.

[14]. Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y., "The curious case of neural text degeneration," in Proceedings of the International Conference on Learning Representations (ICLR), 2020.

[15]. Arazo, E., Ortego, D., Albert, P., O'Connor, N. E., and McGuinness, K., "Pseudo-labeling and confirmation bias in deep semi-supervised learning," in Proceedings of the International Joint Conference on Neural Networks (IJCNN), Glasgow, U.K., 2020, doi: 10.1109/IJCNN48605.2020.9207304.

[16]. Ollama, "Ollama," GitHub repository, accessed Oct. 11, 2026.

[17]. Transmission Systems for Communications, 3rd ed. Western Electric Co., Winston-Salem, NC, USA, 1985, pp. 44–60.

[18]. Motorola Semiconductor Data Manual. Motorola Semiconductor Products Inc., Phoenix, AZ, USA, 1989.

[19]. Vidmar, J., "On the use of atmospheric plasmas as electromagnetic reflectors," IEEE Transactions on Plasma Science, vol. 21, no. 3, pp. 876–880, Aug. 1992.