SMTE at ATE-IT: Ensemble Term Extraction with Italian BERT, spaCy, and Vocabulary-Based Filtering

Sofia Maule[1,*], Giorgio Maria Di Nunzio[1]

[1]*Department of Information Engineering, University of Padova, Via Gradenigo 6/b, 35131 Padova, Italy*

Abstract

This paper describes the SMTE system developed for the shared task ATE-IT (Subtask A: Automatic Term Extraction) at EVALITA 2026 in Italian institutional texts in the municipal waste-management domain. Given a sentence, the goal is to extract domain-specific single- and multi-word terms while complying with task constraints such as lowercasing, sentence-level de-duplication, and the removal of nested terms. SMTE follows an ensemble approach that combines (i) a fine-tuned Italian BERT token-classification model trained with BIO tags to detect term spans, (ii) a trained spaCy sequence labeller to provide complementary candidates, and (iii) a vocabulary built from gold training terms to support normalization and filtering. A final post-processing stage cleans and canonicalises predictions, removes truncated spans and nested terms, and formats the output JSON for evaluation and submission. We report our official evaluation results and discuss strengths, typical errors, and directions for future improvements.

Keywords

Automatic Term Extraction, Terminology, Italian NLP, Domain-specific term extraction, BERT token classification, spaCy, ATE-IT, EVALITA 2026

1. Introduction

Automatic Term Extraction (ATE) is a core task in Natural Language Processing (NLP) and terminology work. Given a domain-specific corpus, its goal is to identify the terms that denote key concepts in that domain, including single and multi-word expressions (e.g., technical noun phrases, derived terms, and common domain abbreviations) [1]. Unlike Named Entity Recognition (NER) [2], which focuses on the mentions of named entities such as people, organizations, or places, ATE targets domain terminology that may not correspond to proper names and is aimed primarily at building terminological resources for downstream applications such as information retrieval, ontology construction, knowledge graph enrichment, and domain adaptation of language models.

EVALITA 2026 is a periodic evaluation campaign for Italian NLP and speech technologies across different tasks [3]. The ATE-IT shared task provides a large-scale benchmark for ATE in Italian and offers a realistic setting based on institutional texts from the municipal waste-management domain. This domain introduces challenges such as lexical variation, abbreviations, synonyms, and complex multiword terms, making it a useful testbed for robust terminology extraction methods [4].

In this work, we specifically tackle the ATE-IT Subtask A (Term Extraction) [4]. In this task, systems receive a collection of sentences and must output, for each sentence, a list of domain-relevant terms. The task imposes strict output constraints: extracted terms must be lowercased, duplicates within the same sentence are not allowed, and nested terms must be avoided (e.g., if a longer term is extracted, its inner subspans should not be returned unless they occur independently). Performance is measured using two complementary metrics: Micro F1, which evaluates precision and recall across all sentences and occurrences [5], and Type F1, calculated based on unique extracted term types.In this context, we present our approach, named SMTE, which is an ensemble system that combines neural sequence labeling and resource-based filtering to extract accurate term spans under ATE-IT constraints.

EVALITA 2026: 9th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian, Feb 26 – 27, Bari, IT
*Corresponding author.
sofia.maule@studenti.unipd.it (S. Maule); giorgiomaria.dinunzio@unipd.it (G. M. Di Nunzio)
0000-0001-7116-9338 (G. M. Di Nunzio)

All code, notebooks, and scripts used to reproduce the experiments and generate the submission files are publicly available at https://github.com/sofiamaule/ATE-IT_SofiaMaule.

The remainder of this paper is organized as follows: Section 2 describes the proposed SMTE system and its main components; Section 3 reports experimental results; and Section 4 discusses strengths, limitations, and future directions.

2. Description of the system

SMTE is an ensemble pipeline for ATE-IT Subtask A that extracts domain-relevant terms from each sentence while enforcing the shared-task output constraints (lowercase, sentence-level de-duplication, and no nested terms) [4]. The final submission system combines two complementary sequence labellers: a BERT-based token classifier and a spaCy-based tagger, and merges their candidates through a vocabulary derived from the gold training terms. A final cleaning stage normalizes and canonicalises the extracted spans, removes noisy or truncated candidates, applies de-duplication and nested-term removal, and formats the output JSON for official evaluation and submission.

All components of SMTE were developed starting from the baseline code and reference implementations released during the course tutorship for the ATE-IT task, which provided working pipelines and multiple system variants. We then adapted and extended this material to build our final submission system [6].

2.1. Preprocessing and data preparation

All sentences are normalized before training and inference to reduce noise and comply with the shared-task constraints. In particular, we:

- replace non-breaking spaces with regular spaces,
- collapse multiple whitespace characters,
- normalize apostrophes by mapping typographic quotes to a single form,
- lowercase the text (coherently with the uncased BERT checkpoint and with the required output format), and
- remove non-printable control characters.

The same normalization is applied to the gold terms used for training.

The input data are loaded from the official ATE-IT JSON files (sentence-level records with a `term_list`) and mapped to the internal representation used to train the sequence labelers..

2.2. BERT token-classification extractor

The main extractor is a token-classification model based on the Italian checkpoint `dbmdz/bert-base-italian-uncased` [7]. We fine-tuned it to detect term spans using a BIO tagging scheme with three labels: `O`, `B-TERM`, and `I-TERM` [8].

Training labels are generated automatically from the gold term lists by locating each term occurrence in the sentence string and mapping the resulting character-level spans to BERT subword tokens using offset mappings. To reduce spurious matches, a candidate match is accepted only if it satisfies a simple word-boundary check, i.e., the characters immediately before and after the match are either absent or non-alphanumeric.

For the final run submitted, the model is fine-tuned in the union of the official training and development sets using the HuggingFace `Trainer` [9]. The final training configuration is as follows:

- learning rate 2×10^{-5};
- batch size 16;
- 7 epochs;
- weight decay 0.01;

- linear learning-rate schedule with warmup ratio 0.1;
- random seed 42.

Mixed-precision training (FP16) is enabled when a CUDA device is available.

At inference time, each sentence is tokenized with truncation and padding, and the most likely label is selected for each token. Term candidates are decoded by grouping consecutive `B-TERM`/`I-TERM` tokens into spans, which are converted back to surface strings via tokenizer detokenization. Finally, the extracted strings are cleaned by trimming the surrounding punctuation, normalizing whitespace, and lowercase, to comply with the task constraints.

2.3. spaCy sequence labeller

As a complementary extractor, we train a spaCy pipeline [10] by defining term extraction as an NER task with a TERM of a single entity type . We start from the Italian pretrained model `it_core_news_md` when available, and train (update) the `ner` component only, while disabling the other pipeline components during optimization.

Training annotations are derived from the gold term lists by searching for each term as a substring of the sentence and converting matched character spans into spaCy entities through `Doc.char_span`, using `alignment_mode=expand` to ensure valid token boundaries. Since multiple matches may overlap, we apply an overlap-resolution heuristic that keeps longer spans when conflicts occur. To focus learning on informative examples, we discard sentences that contain no annotated `TERM` spans.

We train on the union of the official training and development sets for 40 iterations with dropout 0.1 and batch size 8. At inference time, we collect `TERM` spans from `doc.ents`, lowercase them, and remove duplicates at sentence level before the ensemble stage (Section 2.4).

2.4. Vocabulary-based filtering and merging

The final system combines the raw term candidates produced by the BERT extractor (Section 2.2) and the spaCy sequence labeler (Section 2.3) through a lightweight ensemble guided by a gold-derived domain vocabulary. The predictions of the two models are aligned at sentence level using the shared identifiers `(document_id, paragraph_id, sentence_id)`.

All terms are first canonicalized with a shared normalization function that converts the text to lowercase, applies Unicode normalization (NFKC), unifies apostrophes/quotes, collapses whitespace, and strips punctuation at the term boundaries. This normalization is applied to both predictions and gold terms so that matching and filtering are performed consistently.

Gold-derived vocabulary. We build a vocabulary of domain terms from the official training and development annotations (train + dev) by collecting all gold terms and storing their normalized forms. The resulting set acts as a domain whitelist used to validate candidates and to decide which spaCy spans are reliable, reducing the inclusion of well-formed but out-of-domain expressions.

Multiword upgrade (BERT → spaCy). BERT occasionally predicts short fragments that are part of a longer multiword expression (e.g., predicting “ferro” instead of “materiali ferrosi”). To address this, we attempt to “upgrade” each BERT term to a longer spaCy span when this is supported by the gold vocabulary. Concretely, for each BERT prediction, we search among spaCy candidates for the longest multiword span that (i) is present in the gold-derived vocabulary and (ii) contains the BERT term as a contiguous token subspan; if found, the BERT term is replaced by the longer span. We also maintain a small blacklist of clearly non-terminological items (e.g., time expressions and generic discourse words) that are discarded before and after upgrading.

Dictionary-filtered addition of spaCy spans. After upgrading, we enrich the output by adding additional spaCy multiword candidates that are present in the gold-derived vocabulary, are not already selected, and are not blacklisted. This step primarily increases recall for well-formed multiword terms that spaCy detects reliably but that may be missed by BERT, while keeping precision under control via the whitelist.

Generic unigram filtering and final formatting. Finally, we filter out overly generic single-word candidates using a small set of generic heads (e.g., servizio, materiali, impianto) when they do not appear in the gold vocabulary. Acronym-like strings (short alphabetic forms, optionally with dots) are preserved. The remaining terms are normalized and de-duplicated at sentence level using their canonical form, preserving the original output order (upgraded BERT terms first, then added spaCy spans). The final list is emitted in the required ATE-IT JSON format.

2.5. Post-processing

After merging, we apply a deterministic post-processing stage to enforce ATE-IT output constraints and remove systematic noise. Given a sentence and its raw predicted term list, we perform the following steps:

- **String normalization and sentence-level de-duplication** Each candidate term is normalized by trailing whitespace, collapsing multiple spaces, stripping punctuation only at the boundaries, and lowercasing. Empty strings are discarded. We then remove duplicates within the same sentence while preserving the original order.
- **Truncated-term filtering** Both extractors occasionally output incomplete multiword terms that end with a function word (e.g., "gestione dei", "batterie e"), typically caused by boundary errors in span decoding. We filter these cases with a simple heuristic: a candidate is marked as truncated if it has at least two tokens, and its last token belongs to a predefined set of Italian function-word endings (prepositions and conjunctions such as "di", "dei", "e", "a", etc.). All candidates flagged as truncated are removed.
- **Nested-term removal using character spans** The task explicitly excludes nested terms within the same sentence [4]. To enforce this constraint, we remove candidates that are strictly contained inside longer candidates in the sentence texts. For each candidate, we locate all (case-insensitive) occurrences in the sentence and represent them as spans $(start, end)$. We then sort all spans by decreasing length and apply a greedy selection: a candidate span is kept only if it is not fully contained in any already selected span. Finally, selected spans are ordered by their start offset to preserve sentence order, and we return the corresponding unique terms. This step avoids outputting both a long term and its inner subspans when they refer to the same textual occurrence.
- **Output format** After cleaning, the per-sentence term list is written in the official ATE-IT JSON format: `{document_id, paragraph_id, sentence_id, term_list}`. This guarantees lowercase-only terms, sentence-level de-duplication, and compliance with the no-nesting constraint.
- **Final submission sanity-check: token-based nesting removal** Before producing the final submission file, we run an additional lightweight nested-term check directly on each `term_list`, without using the original sentence text. A term is considered nested if its token sequence appears as a contiguous subsequence of a longer term in the same list (e.g., *trattamento rifiuti* inside *impianto di trattamento rifiuti*). All nested (shorter) terms are removed while maintaining order. The script also reports an approximate count of nested pairs before and after this pass, acting as a final safeguard against residual nesting errors.

3. Results

Data splits. During development, we trained models on the official training set and evaluated them on the official development set (`subtask_a_dev.json`) to select hyperparameters and pipeline variants. After selecting the final approach, we retrained the supervised components in a *train+dev* setting and ran inference on the test set to generate the official submission.

Evaluation protocol. We evaluate systems using the ATE-IT official metrics [4] by comparing predicted and gold term lists at sentence level using: (i) **Micro** Precision/Recall/F1 (aggregating true positives, false positives, and false negatives across all sentences) [5] and (ii) **Type** Precision/Recall/F1 (computed over the set of unique predicted vs. unique gold term types, ignoring frequency) .

Systems compared. We compared four families of approaches:

- **LLM baselines** (zero-shot / few-shot): `llm_zero_shot`, `llm_few_shot`.
- **Rule-based baselines**: `vanilla`, and a spaCy pattern-based extractor (`spacy_term_extraction_patterns`).
- **Supervised extractors**: fine-tuned BERT token classifier (Section 2.2) and a spaCy NER tagger trained on ATE-IT data (Section 2.3).
- **Ensembles**: BERT + spaCy merging with gold-vocabulary filtering (Section 2.4), also followed by LLM reranking.

3.1. LLM baselines and reranking: models and prompts

We report the exact models and prompt templates used for the prompt-based baselines and for LLM reranking.

`llm_zero_shot`. Model: `gemini-2.5-flash` via `google.generativeai`. We process sentences in batches of 20 and prompt the model to output one list of terms per sentence in a fixed textual format. The system prompt is:

> You are an automatic term extraction agent. You will receive a list of sentences as input. Your role is to extract waste management terms from the sentences. Output a list of terms for each sentence.
>
> Strictly adhere to the Example Output Format:
>
> Example Output Format: Sentence 1: [term1; term2; term3; term4] Sentence 2: [term5; term6] Sentence 3: [] Sentence 4: [] Sentence 5: [term7]
>
> Instructions: - Extract only terms, ignore named entities - Do not extract nested terms - Extract only terms related to waste management, ignoring other domains - If a sentence contains no terms, output an empty list for that sentence - You must output 20 lists of terms, one for each sentence

`llm_few_shot`. Model: `gemini-2.5-flash` via `google.generativeai`. We use the same batching and output format as in zero-shot, but we prepend annotated examples (few-shot) to steer extraction and constraint compliance. The system prompt is:

> You are an automatic term extraction agent for Italian municipal waste management texts. You will receive a list of sentences as input. Your role is to extract *all and only* waste management terms from each sentence. Output a list of terms for each sentence.
>
> A ”term” in this task is: - a single- or multi-word expression - refers to a concept in the waste management domain - often nouns or noun phrases (sometimes adjectives or verbs as part of a phrase)
>
> Non-terms are: - generic function words (e.g., ”e”, ”di”, ”per”, ”che”) - pure numbers or dates not part of a waste term - person names, city names, street names (unless part of an official name of a waste service)

Strictly adhere to the Example Output Format:

Sentence 1: [term1; term2; term3] Sentence 2: [term4] Sentence 3: []

———————————— ANNOTATED EXAMPLES (FEW-SHOT) ————————————

Example 1 Sentence: ”Il presente disciplinare per la gestione dei centri di raccolta comunali è stato redatto ai sensi del DM 13/05/2009.” Expected output: Sentence 1: [disciplinare per la gestione dei centri di raccolta comunali; centri di raccolta comunali]

Example 2 Sentence: ”Il servizio di raccolta differenziata porta a porta dei rifiuti urbani è attivo su tutto il territorio comunale.” Expected output: Sentence 1: [servizio di raccolta differenziata porta a porta; raccolta differenziata porta a porta; rifiuti urbani]

Example 3 Sentence: ”Il pagamento della Tassa Rifiuti (TARI) avviene tramite il portale pagoPA.” Expected output: Sentence 1: [tassa rifiuti; tari]

Example 4 Sentence: ”Il presente regolamento disciplina le modalità di conferimento dei rifiuti ingombranti presso l'isola ecologica comunale.” Expected output: Sentence 1: [regolamento; modalità di conferimento; rifiuti ingombranti; isola ecologica comunale]

Example 5 Sentence: ”In questa frase non sono presenti termini di gestione dei rifiuti.” Expected output: Sentence 1: []

YOUR TASK: Now you will receive 20 sentences in the following format:

Sentence k: <sentence_text>

For each sentence k, you MUST output exactly one line in this format:

Sentence k: [term1; term2; term3]

Instructions: - Extract only terms related to waste and waste management (e.g., tassa rifiuti, tari, isola ecologica, raccolta differenziata, impianto di trattamento rifiuti). - Prefer complete multi-word terms (full span) over shorter fragments. - Do NOT output nested terms: if you extract ”impianto di trattamento rifiuti urbani”, do NOT also output ”trattamento rifiuti urbani”, unless it appears as a separate term in the sentence. - Ignore named entities (people, cities, streets) unless they are part of an official waste management term. - If a sentence contains no relevant terms, output an empty list: Sentence k: []. - You must output one line for each of the 20 input sentences, in order (Sentence 1, Sentence 2, ...).

LLM reranking. Models (Groq-hosted): `openai/gpt-oss-120b`, `openai/gpt-oss-20b`, `qwen/qwen3-32b`. Given a sentence and a candidate term list (from supervised extractors), the reranker assigns a score in $[0, 1]$ and a keep/discard decision for each candidate, returning strict JSON. The system prompt is:

You are an automatic term extraction reranking agent for Italian municipal waste management texts.

You will receive: - one sentence in Italian, and - a list of candidate terms extracted by a baseline system.

Your task: - For each candidate term, decide if it is a good domain-relevant term in the context of the sentence. - Assign a relevance score between 0.0 and 1.0 (higher = better). - Decide whether to keep or discard each candidate.

———————————— DOMAIN KNOWLEDGE (WASTE MANAGEMENT) ————————————

Valid waste management terms typically include:

1) Waste types and materials - rifiuti, rifiuti urbani, rifiuti ingombranti, rifiuti pericolosi, rifiuti organici - carta, cartone, plastica, vetro, metalli, alluminio, banda stagnata, legno, tessili - RAEE / R.A.E.E., toner, oli esausti

2) Services, infrastructures, processes - raccolta differenziata, raccolta porta a porta, modalità di raccolta, modalità di conferimento - servizio di raccolta, servizio di igiene urbana - centro di raccolta, centri di raccolta comunali, isola ecologica, ecocentro, piattaforma ecologica - impianto di trattamento rifiuti, impianto di smaltimento, discarica, compostaggio

System (dev)	P_μ	R_μ	$F1_\mu$	P_t	R_t	$F1_t$
Ensembles (BERT + spaCy + vocabulary)						
`ensemble_bert_2e-5_spacy_dictfilter`	0.770	0.734	**0.751**	0.728	0.674	0.700
`ensemble_bert_2e-5_spacy_baseline_dictfilter`	0.755	0.736	0.745	0.703	0.686	0.695
`ensemble_bert_spacy` (no vocab filter)	0.640	0.703	0.670	0.545	0.628	0.583
LLM reranking (on top of supervised candidates)						
`ensemble_bert_2e-5_reranked_llm`	0.782	0.701	0.739	0.760	0.653	0.702
`ensemble_bert_2e-5_cased_reranked_llm`	0.756	0.714	0.734	0.695	0.669	0.682
BERT-only tuning						
`bert_2e-5_cleaned`	0.758	0.710	0.733	0.714	0.649	0.680
`bert_2e-5`	0.732	0.710	0.721	0.677	0.649	0.662
`bert_3e-5_cleaned`	0.734	0.681	0.707	0.684	0.607	0.643
Baselines and spaCy-only						
`llm_few_shot`	0.463	0.665	0.546	0.404	0.719	0.517
`llm_zero_shot`	0.481	0.630	0.545	0.408	0.653	0.502
`vanilla`	0.428	0.738	0.542	0.657	0.562	0.606
`spacy_trained`	0.591	0.353	0.442	0.554	0.380	0.451
`spacy_baseline`	0.502	0.350	0.413	0.701	0.368	0.482
`spacy_term_extraction_patterns`	0.051	0.373	0.090	0.041	0.417	0.074

Table 1
Development-set results for the main exploratory runs. Micro metrics (subscript μ) aggregate matches across sentences; Type metrics (subscript t) are computed over unique term types [4].

3) Regulations, tariffs, administrative terms - regolamento, disciplinare, disciplinare per la gestione dei centri di raccolta comunali - tassa rifiuti, TARI, tariffa puntuale, tariffe - utenti, utenze domestiche, utenze non domestiche

4) Action verbs and phrases that are domain terms - conferire, conferiti, vanno conferiti, conferimento, modalità di conferimento - depositare, deposito, conferire i rifiuti, conferire i rifiuti vegetali

Non-terms are: - generic function words (es. ”e”, ”di”, ”per”, ”che”) - pure numbers or dates not part of a waste term - person names, city names, street names (unless part of an official waste management service) - clearly truncated fragments (e.g., ”dei”, ”oo”, ”r” alone)

———————————— OUTPUT FORMAT (STRICT JSON) ————————————

You MUST output ONLY a JSON object with this exact structure:

”reranked_terms”: [”term”: ”...”, ”score”: 0.0, ”keep”: true or false, ...]

Rules: - Do not add new terms that are not in the candidate list. - Do not modify the spelling of the candidates. - If a candidate looks truncated or not a full concept, set ”keep”: false and give it a low score. - If no candidate is good, you may return an empty list: ”reranked_terms”: []. - The JSON must be valid and parseable by Python’s json.loads().

Your scores will be later combined with a rule-based domain scorer, so: - assign higher scores to candidates that clearly match the domain knowledge above, - and lower scores to generic or borderline expressions.

The corresponding user prompt provides the sentence and the candidate list (one per line) and asks the model to output only the JSON object described above.

3.2. Exploratory results on development set

Table 1 reports the main exploratory runs on the development set. For readability, we group results by model family and highlight the best configurations.

Exploratory configurations. Before selecting the final SMTE pipeline, we varied both the supervised extractors and the ensemble strategy. For BERT, we compared *uncased vs. cased* Italian checkpoints and multiple fine-tuning settings (e.g., learning rate $2 \cdot 10^{-5}$ vs. $3 \cdot 10^{-5}$), and we measured the effect of deterministic cleaning on span decoding. For spaCy, we trained the `TERM` tagger starting from different Italian backbones (`it_core_news_sm` and `it_core_news_md`) and compared a trained tagger against a lighter baseline component. Finally, we tested multiple integration variants: (i) dictionary-guided merge (BERT+spaCy+vocabulary), (ii) a weaker merge using spaCy baseline spans, and (iii) LLM reranking on top of supervised candidates.

Discussion of the results. Results show a clear separation between *supervised* approaches trained on ATE-IT annotations and *prompt-based* / purely rule-based baselines. The best exploratory system is the supervised ensemble `ensemble_bert_2e-5_spacy_dictfilter` (Micro-F1 = 0.751; $P_\mu = 0.770$, $R_\mu = 0.734$), improving over the strongest single-component model, BERT-only with deterministic cleaning (`bert_2e-5_cleaned`, Micro-F1 = 0.733). This gain comes from combining three complementary signals: (i) BERT provides accurate contextual supervision for term boundaries, (ii) spaCy contributes additional syntactically well-formed multiword candidates, and (iii) the gold-derived vocabulary acts as a domain constraint that prevents spaCy expansions from introducing many out-of-domain false positives. A key mechanism is *multiword upgrading*: when BERT predicts a short fragment, it can be replaced with a longer spaCy span only if the resulting multiword expression is present in the gold-derived vocabulary, improving span completeness while keeping precision controlled.

Post-processing is consistently beneficial: for the same BERT configuration, Micro-F1 increases from 0.721 (`bert_2e-5`) to 0.733 (`bert_2e-5_cleaned`), mainly by removing truncated spans, duplicates, and nested outputs. Moreover, the vocabulary constraint is crucial: simply pooling the BERT+spaCy candidates without filtering (`ensemble_bert_spacy`, Micro-F1 = 0.670) is substantially worse than dictionary-guided merge. Finally, LLM baselines remain around Micro-F1 $\approx$ 0.545 and show the typical high-recall/low-precision pattern, suggesting that without supervised adaptation they struggle to match span guidelines and domain constraints, so they were later discarded.

LLM reranking does not outperform the vocabulary-filtered ensemble: for example, the `ensemble_bert_2e-5_reranked_llm` produces Micro-F1 = 0.739, increasing precision ($P_\mu = 0.782$) but decreasing recall ($R_\mu = 0.701$), the reranker acts mainly as a conservative filter that removes both false positives and some true terms, so the loss in recall outweighs the precision gain, and the overall F1 remains below the best dictionary-guided merge.

3.3. Final re-run and submission-ready outputs

After selecting the best approach (BERT + spaCy + vocabulary merge) and the best models and hyper-parameters, a final re-run was performed using simplified names and explicitly checked compliance with ATE-IT output constraints (in particular, *no nested terms*). Table 2 reports this final re-run:

- **`final_MD`** / **`final_SM`** are the ensemble outputs after the BERT+spaCy+vocabulary merge (Section 2.4), where spaCy is trained starting from `it_core_news_md` vs. `it_core_news_sm`.
- **`final_MD_submission`** / **`final_SM_submission`** are the corresponding outputs after applying `clean_final_submission.ipynb`, which performs an additional token-based nested-term removal on each sentence `term_list` as a final safeguard.

Note on scores (model selection vs. submission-ready output). The best exploratory configuration (`ensemble_bert_2e-5_spacy_dictfilter`) reaches Micro-F1 = 0.751 on the development set. However, after manual inspection, we observed residual violations of the official ATE-IT output constraints, in particular the presence of nested terms within the same sentence-level `term_list`. Our original post-processing already performs string normalization, sentence-level de-duplication, truncated-span filtering, and nested-term removal by locating term occurrences in the sentence text (character-span based filtering). Nevertheless, a small number of token-level nesting cases can survive because span matching is sensitive to edge cases, such as punctuation/whitespace variants, partial

System (dev)	P_μ	R_μ	$F1_\mu$	P_t	R_t	$F1_t$
`final_MD`	0.765	0.727	0.745	0.718	0.674	0.695
`final_SM`	0.743	0.725	0.734	0.688	0.674	0.681
`final_MD_submission`	0.773	0.723	**0.747**	0.725	0.665	0.694
`final_SM_submission`	0.774	0.721	0.746	0.725	0.665	0.694
`bert_2e-5`	0.732	0.710	0.721	0.677	0.649	0.662
`spacy_trained_MD`	0.598	0.339	0.433	0.585	0.384	0.464
`spacy_trained_SM`	0.476	0.326	0.387	0.405	0.368	0.385
`spacy_baseline_MD`	0.289	0.341	0.313	0.349	0.360	0.354
`spacy_baseline_SM`	0.289	0.341	0.313	0.349	0.360	0.354

Table 2
Final re-run on development set: ensemble outputs and submission-cleaned outputs. The `*_submission` variants include a final token-based nesting-removal safeguard.

System	P_μ	R_μ	$F1_\mu$	P_t	R_t	$F1_t$
Baseline	0.497	0.559	0.526	0.435	0.508	0.469
SMTE (ours)	**0.656**	0.577	**0.614**	0.645	0.529	**0.581**

Table 3
Official ATE-IT Subtask A results on the hidden test set for our submitted system (SMTE) and the shared-task baseline [11]. The table reports both Micro-level (P_μ, R_μ, $F1_\mu$) and Type-level (P_t, R_t, $F1_t$) evaluation metrics.

matches, or multiple occurrences of the same term in the sentence. For the final submission artifact, we therefore added a deterministic "safety" pass that removes nested terms directly at the `term_list` level via token-subspan containment checks (e.g., dropping *trattamento rifiuti* when *impianto di trattamento rifiuti* is also present in the list). This produces a submission-ready output that strictly enforces the constraints and slightly changes the precision/recall balance, yielding Micro-F1 = 0.747 on dev set. We report both values to clearly distinguish *model selection* from the final *constraint-compliant submission output.*

3.4. Official results on the test set

After selecting the final configuration, we trained the supervised components on the union of the official training and development data (*train+dev*) and generated predictions for the official hidden test set, which was evaluated by the shared-task organizers. Table 3 reports the official leaderboard scores for our submitted run (SMTE), together with the baseline provided for reference [11].

Development-set scores (Section 3.2) are used for model selection and ablations, while the official results are computed on a hidden test set. The drop from dev to test is expected and reflects generalization to unseen data (e.g., different term distribution and sentence formulations).

4. Discussion

This section summarizes what worked well in SMTE, what did not, and which improvements are most promising for future iterations.

4.1. What worked

Supervised extraction is essential. Development experiments show that supervised sequence labeling is the key factor for competitive performance in ATE-IT. Both BERT and the spaCy tagger benefit from direct training on the task annotations, which encode not only domain terminology, but also the shared-task span conventions. In contrast, prompt-based LLM baselines and purely rule-based systems tend to over-generate plausible candidates and struggle to consistently match the gold span boundaries and selection rules.

The ensemble improves span completeness and robustness. The best results are obtained by combining complementary signals: BERT contributes high-quality contextual boundaries, while spaCy contributes syntactically coherent multiword candidates. The ensemble logic is especially effective when BERT predicts an incomplete fragment of a multiword expression: the *multiword upgrade* step replaces short predictions with longer spaCy spans when supported by the gold-derived vocabulary. This mechanism improves term completeness without inflating false positives, yielding a better precision/recall balance than either component alone.

Deterministic cleaning and constraint enforcement reduce false positives. Post-processing steps (normalization, duplicate removal, truncated-span filtering, and nested-term removal) consistently improve quality. In this task, compliance checks are not only formatting details: violations, such as truncated terms or nested output, directly translate into false positives under the official evaluation. The final submission-cleaning step provides an additional safeguard by removing residual nested terms at the list level, ensuring that the output respects the ATE-IT constraints in a deterministic and reproducible way.

4.2. What did not work

LLM reranking is not reliably beneficial. LLM reranking slightly increases precision in some runs by discarding questionable candidates, but it does not improve Micro-F1 over the best vocabulary-filtered merge. The most common failure mode is a recall drop: reranking can remove correct terms when the model is uncertain about guideline-specific decisions (e.g., whether a generic-looking span is acceptable in this domain).

spaCy-only systems have limited recall and boundary noise. The spaCy trained tagger improves over baseline spaCy components, but alone it remains substantially weaker than BERT-based extraction. Typical issues are missed domain terms (especially less frequent expressions) and boundary mismatches (e.g., including or excluding function words at the edges of a term). This confirms the utility of spaCy primarily as a *complementary candidate generator* within the ensemble, rather than as a standalone extractor.

4.3. Error analysis and limitations

Boundary errors and partial spans. Despite cleaning, span boundary errors remain a core challenge: models may include trailing function words (“gestione dei”), misleading modifiers, or output truncated multiword expressions. These errors are amplified by sentence-level evaluation, where small boundary differences make a prediction count as incorrect.

Generic terms vs. domain-specific usage. The municipal waste-management domain contains many words that are meaningful in context but generic in isolation (e.g., servizio, gestore, materiali). Even with generic-header filtering and a gold-derived vocabulary, distinguishing true terminology from generic discourse remains difficult, especially for single-word candidates. This affects both precision (generic false positives) and recall (over-filtering legitimate generic-looking terms).

Nested terms and overlapping candidates. The no-nesting constraint is strict and sometimes ambiguous in practice: a shorter span can be a valid term type but becomes invalid if it is nested inside a longer term in the same sentence occurrence. This requires sentence-aware reasoning and careful implementation.

Vocabulary dependence and domain drift. The vocabulary filter substantially improves robustness, but it introduces a limitation: it is built from train and development gold terms, so it may under-represent rare or unseen terminology in the test set. This can cap recall when new domain expressions appear (domain drift) or when gold annotations include valid variants not present in the training vocabulary.

4.4. Future work

Several extensions could further improve SMTE system:

- **Constraint-aware decoding.** Integrate "no nesting" directly into decoding (or candidate selection) instead of enforcing it after generation, e.g., by selecting a globally consistent set of non-overlapping spans per sentence.
- **Stronger span modelling.** Replace token-level BIO with span-based methods (e.g. pointer/span classification) to reduce boundary mismatches and truncated output.
- **Richer domain resources;** Expand the vocabulary with external terminological sources (e.g., waste-management glossaries) and add controlled normalization rules for abbreviations and orthographic variants.
- **Better calibration for generic terms;** Replace hard-coded generic unigram filtering with a learned classifier (or calibrated scoring) that uses contextual features to decide when generic-looking words are terminological.
- **Using more context;** The task is sentence-based, but lightweight document context (e.g., previous sentence, section title when available) could help disambiguate generic candidates and improve consistency.

In summary, the final system demonstrate that the most reliable strategy for ATE-IT is a supervised ensemble with deterministic constraint enforcement: BERT provides strong contextual supervision, spaCy adds multiword robustness, and vocabulary-guided filtering plus post-processing ensures both higher precision and compliance with the task output requirements.

Declaration on Generative AI

In this work Generative AI tools were used during development to assist with coding (drafting small code snippets, refactoring suggestions, and documentation wording) and to support the writing of parts of the report (e.g., wording refinements and clarity improvements). All suggested code was manually reviewed, debugged, and adapted to the project constraints, and the final submission consists of runnable notebooks verified by the author.

References

[1] G. M. Di Nunzio, S. Marchesin, G. Silvello, A systematic review of Automatic Term Extraction: What happened in 2022?, Digital Scholarship in the Humanities 38 (2023) i41–i47. doi:10.1093/llc/fqad030.

[2] B. Jehangir, S. Radhakrishnan, R. Agarwal, A survey on Named Entity Recognition — datasets, tools, and methodologies, Natural Language Processing Journal 3 (2023) 100017. doi:10.1016/j.nlp.2023.100017.

[3] F. Cutugno, A. Miaschi, A. P. Aprosio, G. Rambelli, L. Siciliani, M. A. Stranisci, Evalita 2026: Overview of the 9th evaluation campaign of natural language processing and speech tools for italian, in: Proceedings of the Ninth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2026), CEUR.org, Bari, Italy, 2026.

[4] N. Cirillo, G. M. Di Nunzio, F. Vezzani, Ate-it at evalita 2026: Overview of the automatic term extraction italian testbed task, in: Proceedings of the Ninth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2026), CEUR.org, Bari, Italy, 2026.

[5] R. Verborgh, M. Röder, R. Usbeck, A.-C. Ngonga Ngomo, Gerbil – benchmarking named entity recognition and linking consistently, Semantic Web 9 (2018) 605–625. doi:10.3233/SW-170286.

[6] M. Martinelli, Ata_tutorship: Tutorship baseline code and reference implementations for ate-it, GitHub repository, 2025. URL: https://github.com/MMartinelli-hub/ATA_Tutorship.

[7] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of NAACL-HLT, 2019, pp. 4171–4186. doi:10.18653/v1/N19-1423.

[8] L. Ramshaw, M. Marcus, Text Chunking using Transformation-Based Learning, in: Third Workshop on Very Large Corpora, 1995.

[9] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Brew, Transformers: State-of-the-art natural language processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2020, pp. 38–45. doi:10.18653/v1/2020.emnlp-demos.6.

[10] M. Honnibal, I. Montani, S. Van Landeghem, A. Boyd, spacy: Industrial-strength natural language processing in python, Website, 2025. URL: https://spacy.io/.

[11] ATE-IT Shared Task, Ate-it results (evalita 2026) — official leaderboard, Website, 2025. URL: https://nicolacirillo.github.io/ate-it/results.html.