The ALCHEmist: Automated Labeling 500x CHEaper Than LLM Data Annotators

Tzu-Heng Huang, Catherine Cao, Vaishnavi Bhargava, Frederic Sala

WISCONSIN UNIVERSITY OF WISCONSIN–MADISON

SprocketLab

Challenges

LLM-as-a-annotator offers efficiency over costly human labelers but come with:

⚠ **API Cost:** Labeling a moderate dataset can cost thousands.

⚠ **Inflexibility:** Small label schema changes require a full pipeline rerun.

⚠ **Opaque:** API access doesn't allow any model inspection.

Our Solution

Instead of prompting LLMs for labels, we distill LLM into **labeling programs** that can run locally for free.

Introducing Alchemist

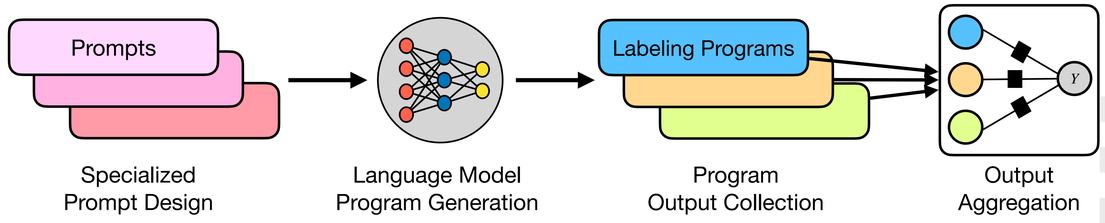
Synthesized programs either label directly or label a training dataset to train a distilled specialist model as your annotator.

Why is this good?

- The number of API calls no longer scales with dataset size.
- Synthesized programs allow unlimited predictions locally.
- Labeling logic can be inspected and corrected by domain experts, allowing easy adaptation to your changes.

Program outputs can be noisy! We use **weak supervision** to aggregate multiple noisy sources into final labels.

Alchemist Pipeline

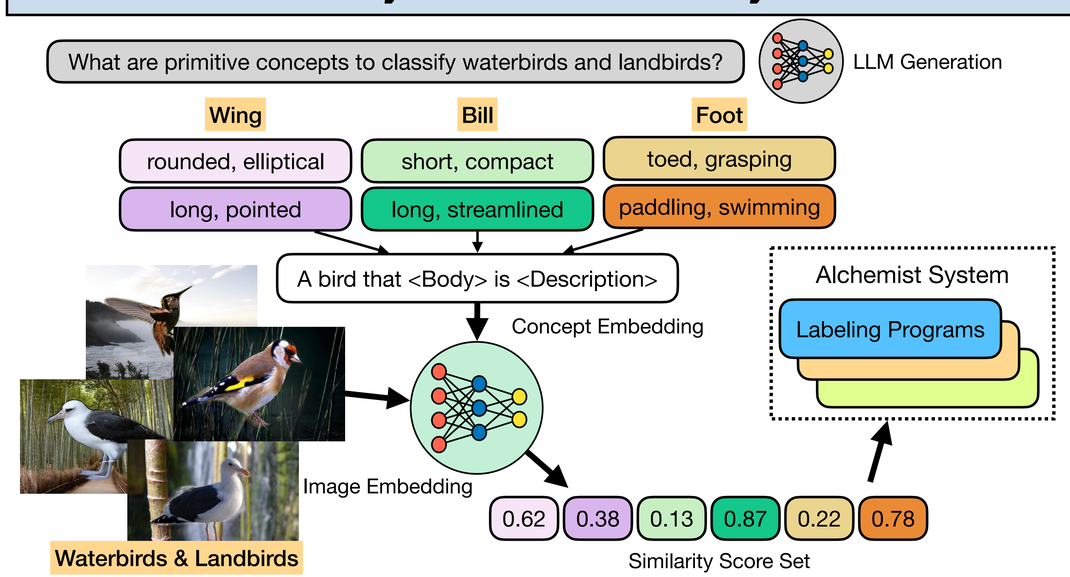


Synthesized Programs

"Write a bug-free and executable function in python to label comment on Youtube as spam or ham. If the comment is spam, function returns 1. If the comment is ham, function returns 0. def label_spam (text_comment):"

```python
def label_spam(text_comment):
    # Define spammy characteristics
    url_pattern = r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+'
    email_pattern = r'\S+@\S+'
    phone_pattern = r'\(?([0-9]{3})\)?([ .-]?)([0-9]{3})\2([0-9]{4})'
    multiple_exclamations = r'!!+'
    multiple_questions = r'\?\?+'
    keyword_list = ['subscribe', 'channel', 'check out', 'follow', 'my page', 'discount', 'save', 'free', 'buy', 'purchase']

    # Check if comment has URL, email, or phone number
    if re.search(url_pattern, comment) or re.search(email_pattern, comment) or re.search(phone_pattern, comment):
        return 1

    # Check if comment has multiple exclamation marks or question marks
    if re.search(multiple_exclamations, comment) or re.search(multiple_questions, comment):
        return 1

    # Check if comment has spammy keywords
    for keyword in keyword_list:
        if keyword in comment.lower():
            return 1

    # If none of these conditions are met, return 0 (ham)
    return 0
```

Beyond Text-Modality



Main Result

	Zero-shot Prompting		Alchemist with GPT-3.5	
	Cost ($)	Accuracy (%)	Cost ($)	Accuracy (%)
YouTube	0.096	87.1	0.004	89.1
SMS	0.240	90.7	0.004	90.0
Yelp	3.873	84.5	0.005	57.5
IMDb	3.400	73.7	0.004	66.2
MedAbs	1.944	31.1	0.006	34.6
Cancer	15.925	71.6	0.003	96.8
Finance	0.201	64.1	0.007	66.0
French	0.641	61.1	0.006	69.0
	Total Cost ($)	Avg. Acc (%)	Total Cost ($)	Avg. Acc (%)
	26.32	70.49	**0.039 (674x Cheaper)**	**71.15**

Alchemist achieves **higher** performance than zero-shot prompting, **yet costs 674 times less using just 10 API calls.**

Image Dataset Result

```python
def label_bird_image(toed_grasping_score, paddling_swimming_score):
    """
    Labels bird images into classes based on foot type similarity scores.
    Parameters:
    - toed_grasping_score (float): Similarity score for 'toed, grasping'.
    - paddling_swimming_score (float): Similarity score for 'paddling, swimming'.
    Returns:
    - str: "landbird", "waterbird", or -1 if it cannot be determined.
    """
    threshold = 0.5
    if toed_grasping_score > threshold and paddling_swimming_score < threshold:
        return "landbird"
    elif paddling_swimming_score > threshold and toed_grasping_score < threshold:
        return "waterbird"
    elif abs(toed_grasping_score - paddling_swimming_score) < 0.1:  # Similar scores
        return -1
    else:
        if toed_grasping_score > paddling_swimming_score:
            return "landbird"
        else:
            return "waterbird"
```

Feature Extractor	Method	Avg. Acc. (↑)	Worst Group Acc. (↑)
—	Vanilla Alchemist with GPT4o	39.5	36.7
	Vanilla Alchemist with Claude 3	78.1	2.2
CLIP ViT-B/32	Zero-shot Prompting	82.0	31.8
	Alchemist with GPT4o	80.5	28.3
	Alchemist with Claude 3	77.4	**46.3**
CLIP ViT-L/14	Zero-shot Prompting	**90.4**	33.5
	Alchemist with GPT4o	80.2	**46.7**
	Alchemist with Claude 3	73.7	**34.6**

Alchemist works **beyond text datasets.** On the **Waterbird** image dataset, we improve **robustness** to spurious correlations, while maintaining average accuracy.