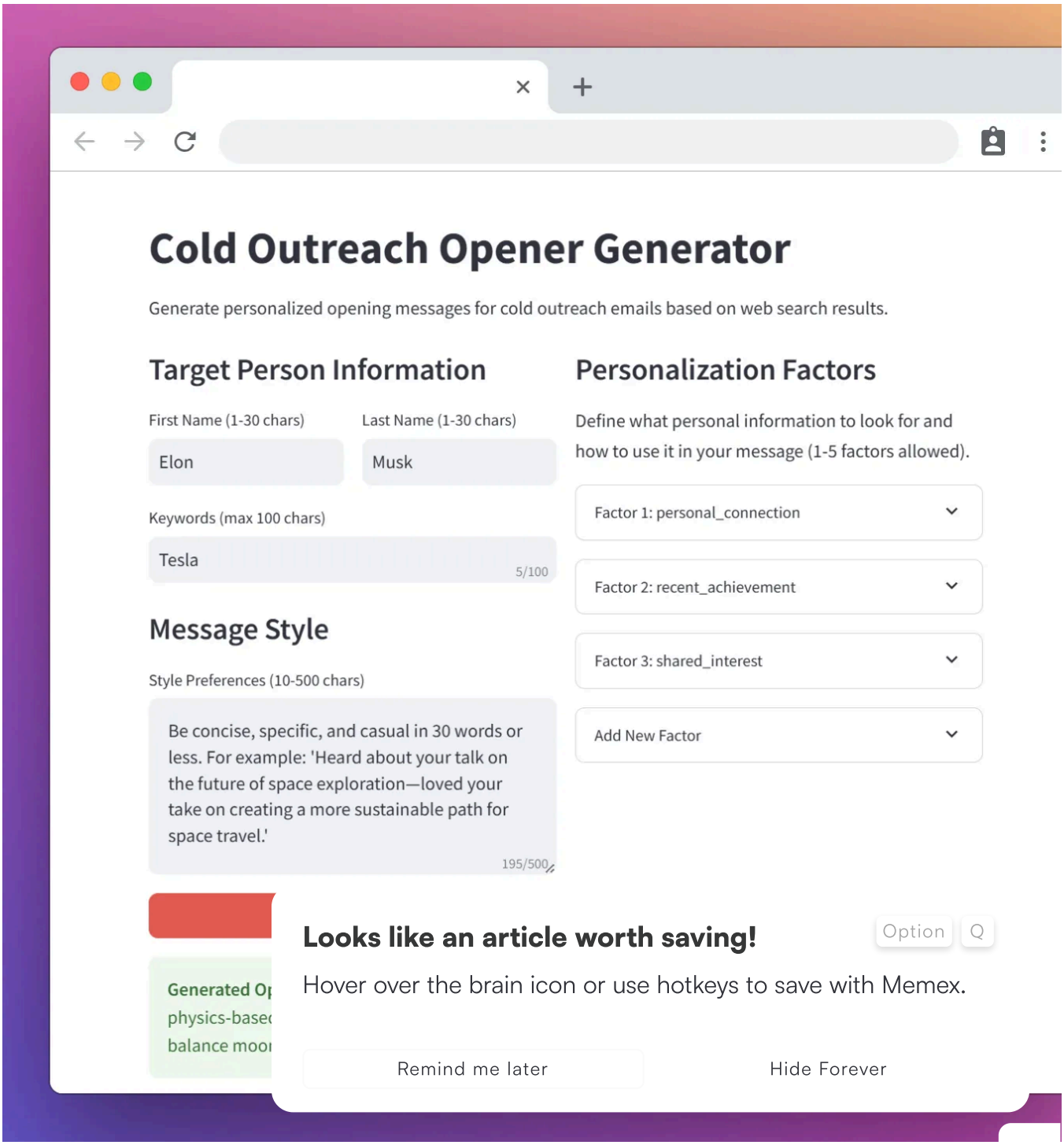
Use AI to Generate Cold Outreach Openers

Step-by-Step Tutorial using Pocket Flow, a 100-line LLM framework!

ZACHARY HUANG
MAR 14, 2025



Cold outreach is a numbers game—but that doesn't mean it has to feel like spam.

What if you could personally research **each prospect**, find their recent achievemer interests, and background, and craft a thoughtful opening message that shows you done your homework?

That's exactly what we're building today: a tool that uses AI to automate what wou normally take hours of manual research and writing. In this tutorial, I'll show you to use AI to generate cold outreach openers that are:

- **Actually personalized** (not just "Hey {first_name}!")
- **Based on real research** (not made-up facts)
- **Attention-grabbing** (by referencing things your prospect actually cares about)

The best part? You can adapt this approach for your own needs—whether you're looking for a job, raising funds for your startup, or reaching out to potential client

The result is available at: https://github.com/The-Pocket/Tutorial-Cold-Email-Personalization

You can try it online: https://pocket-opener-851564657364.us-east1.run.app

Let's dive in.

How It Works: The System Behind Personalize AI Openers

Here's the high-level

1. **Input**: You provi keywords)
2. **Research**: The AI searches the web for information about your prospect

3. **Analysis**: The AI analyzes the search results for personalization opportunities
4. **Generation**: The AI crafts a personalized opening message based on its resea
5. **Output**: You get a ready-to-use opening message

The entire process takes about 30-60 seconds per prospect—compared to the 15+ minutes it might take to do this research manually.

This system is built using [Pocket Flow](), a 100-line minimalist framework for buildi LLM applications. What makes Pocket Flow special isn't just its compact size, but how it reveals the inner workings of AI application development in a clear, educat way.

Getting Started: Setting Up Your Environment

To follow along with this tutorial, you'll need:

1. API keys for AI and search services
2. Basic Python knowledge
3. Git to clone the repository

> **Note:** The implementation uses Google Search API and Claude for AI, but you easily replace them with your preferred services such as OpenAI GPT or SerpA depending on your needs.

If you just want to try it out first, you can use the [live demo]().

Step 1: Clone the Repository

Start by cloning the repository with all the code you need:

```
git clone https:/
Personalization.
cd Tutorial-Cold-
```

Step 2: Set Up Your API Keys

Create a `.env` file in the project root directory with your API keys:

```
OPENAI_API_KEY=your_openai_api_key_here
```

The tool is designed to work with different AI and search providers. Here's a simp implementation of `call_llm` using OpenAI:

```
# utils/call_llm.py example
import os
from openai import OpenAI

def call_llm(prompt):
    """Simple implementation using OpenAI."""
    client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))
    response = client.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=[{"role": "user", "content": prompt}]
    )
    return response.choices[0].message.content

# Test the function
if __name__ == "__main__":
    print(call_llm("Write a one-sentence greeting."))
```

```
# utils/call_llm.py example
import os
from openai import OpenAI

def call_llm(pror
    """Simple imp
    client = Oper
    response = c
        model="gp
        messages=
    )
    return response.choices[0].message.content
```

```
# Test the function
if __name__ == "__main__":
    print(call_llm("Write a one-sentence greeting."))
```

You can easily modify this to use other AI services or add features like caching.

The `search_web` utility function is implemented in a similar way—a simple func that takes a query and returns search results. Just like with the LLM implementati you can swap in your preferred search provider (Google Search, SerpAPI, etc.) bas on your needs.

Make sure your API keys work by testing the utility functions:

```
python utils/call_llm.py  # Test your AI implementation
python utils/search_web.py  # Test your search implementation
```

If both scripts run without errors, you're ready to go!

Step 3: Install Dependencies

Install the required Python packages:

```
pip install -r requirements.txt
```

Using the Tool: Your First Personalized Opene

Now that you have everything set up, let's generate your first personalized opener. tool offers multiple i

- **Command line i**
- **Web UI** for a use
- **Batch processing** for handling multiple prospects at scale

Choose the method that works best for your specific needs:

Method 1: Using the Command Line Interface

The simplest way to generate a single opener is through the command line:

```
python main.py
```

This will prompt you for:

- First name
- Last name
- Keywords related to the person (like company names or topics they're known

Method 2: Using the Web Interface

For a more user-friendly experience, run the web interface:

```
streamlit run app.py
```

This will open a browser window where you can:

1. Enter the target person's information
2. Define personalization factors to look for
3. Set your preferred message style
4. Generate and review the opening message

Method 3: Bat

For efficiently handli l b processing mode:

```
python main_batch.py --input my_targets.csv --output my_results.csv
```

Your input CSV should have three columns:

- `first_name`: Prospect's first name
- `last_name`: Prospect's last name
- `keywords`: Space-separated keywords (e.g., "Tesla SpaceX entrepreneur")

This is particularly useful when you need to reach out to dozens or hundreds of prospects. The system will:

1. Process each row in your CSV file
2. Perform web searches for each prospect
3. Generate personalized openers for each one
4. Write the results back to your output CSV file

The output CSV will contain all your original data plus an additional column with generated opening message for each prospect. You can then import this directly ir your email marketing tool or CRM system.

Example batch processing workflow:

1. Prepare a CSV with your prospect list
2. Run the batch processing command
3. Let it run (processing time: ~1 minute per prospect)
4. Review and refine the generated openers in the output CSV
5. Import into your

Recommende

For the best results, v

1. **Start with single mode or the Streamlit UI** to fine-tune your personalization factors and message style. This gives you immediate feedback on what works
2. **Experiment with different settings** for a few test prospects until you find the perfect combination of personalization factors and style preferences.
3. **Once satisfied with the results**, scale up using the batch processing mode to handle your entire prospect list.

This workflow ensures you don't waste time and API calls processing a large batcl with suboptimal settings, and helps you refine your approach before scaling.

Understanding the Magic: How the AI Personalization Works

This system is built using [Pocket Flow](), a 100-line minimalist framework for buildi LLM applications. What makes Pocket Flow special isn't just its compact size, bu how it reveals the inner workings of AI application development in a clear, educat way.

Unlike complex frameworks that hide implementation details, Pocket Flow's minimalist design makes it perfect for learning how LLM applications actually wc under the hood. With just 100 lines of core code, it's impressively expressive, allov you to build sophisticated AI workflows while still understanding every componer Despite its small size, it provides many of the same capabilities you'd find in large libraries like LangChain, LangGraph, or CrewAI:

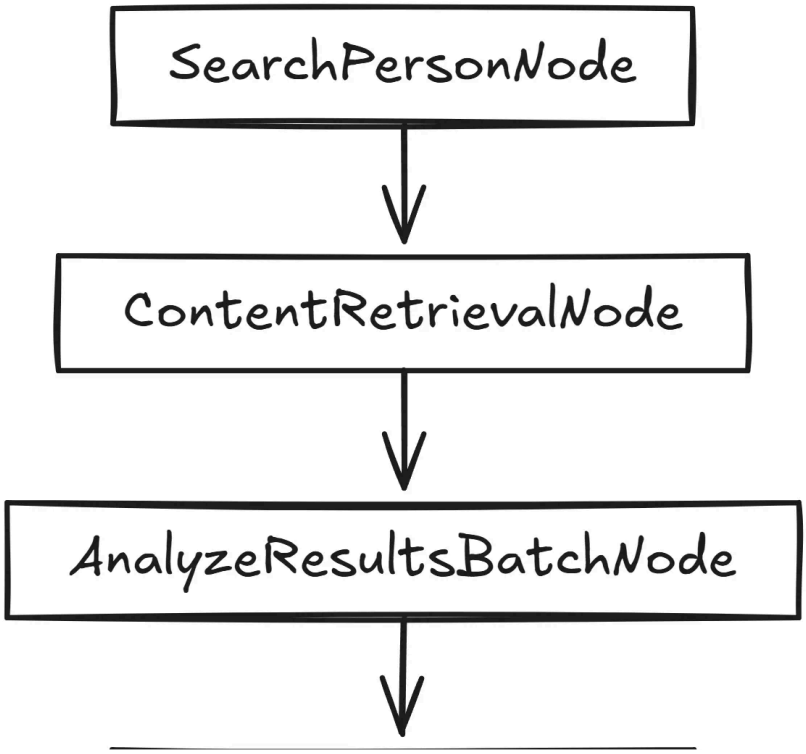
- **Agents & Tools**: Build autonomous AI agents that can use tools and make decisions
- **RAG (Retrieval Augmented Generation)**: Enhance LLM responses with exter knowledge
- **Task Decompos**
- **Parallel Process**
- **Multi-Agent Sys**

The difference? You can read and understand Pocket Flow's entire codebase in minutes, making it perfect for learning and customization.

Pocket Flow's approach to complex AI workflows is elegant and transparent:

- **Graph-based Processing**: Each task is a node in a graph, making the flow easy understand and modify
- **Shared State**: Nodes communicate through a shared store, eliminating compl data passing
- **Batch Processing**: Built-in support for parallel processing of multiple items
- **Flexibility**: Easy to swap components or add new features without breaking existing code

Let's look at how we've structured our cold outreach system using Pocket Flow:



The system follows a

1. **SearchPersonNode**: Searches the web for information about the prospec

2. **ContentRetrievalNode** (Batch): Retrieves and processes content from search results in parallel
3. **AnalyzeResultsBatchNode** (Batch): Analyzes content for personalization opportunities using LLM
4. **DraftOpeningNode**: Creates the final personalized opener

What makes this architecture powerful is its:

- **Modularity**: Each component can be improved independently
- **Parallel Processing**: Batch nodes handle multiple items simultaneously
- **Flexibility**: You can swap in different search providers or LLMs
- **Scalability**: Works for single prospects or batch processing

Now, let's break down the implementation details for each phase:

1. Web Search Phase

The system first searches the web for information about your prospect using their name and the keywords you provided:

```
# From flow.py
class SearchPersonNode(Node):
    def prep(self, shared):
        first_name = shared["input"]["first_name"]
        last_name = shared["input"]["last_name"]
        keywords = shared["input"]["keywords"]

        query = f"{first_name} {last_name} {keywords}"
        return query

    def exec(self
        search_re
        return se
```

By default, the implementation uses Google Search API, but you can easily swap t out for another search provider like SerpAPI in the `search_web` utility function. flexibility allows you to use whichever search provider works best for your needs c budget.

2. Content Retrieval Phase

Next, it retrieves and processes the content from the top search results:

```
class ContentRetrievalNode(BatchNode):
    def prep(self, shared):
        search_results = shared["search_results"]
        urls = [result["link"] for result in search_results if "link"
result]
        return urls

    def exec(self, url):
        content = get_html_content(url)
        return {"url": url, "content": content}
```

3. Analysis Phase

The system then analyzes the content looking for specific personalization factors defined:

```
class AnalyzeResultsBatchNode(BatchNode):
    def exec(self, url_content_pair):
        # Prepare prompt for LLM analysis
        prompt = f"""Analyze the following webpage content about
{self.first_name} {self.last_name}.
        Look for the following personalization factors:

{self._format_pe                                                    }"'

        # LLM ana                                                    s
        analysis_
        return ar
```

4. Generation Phase

Finally, the system crafts a personalized opener based on the discovered informati

```
class DraftOpeningNode(Node):
    def exec(self, prep_data):
        first_name, last_name, style, personalization = prep_data

        prompt = f"""Draft a personalized opening message for a cold
outreach email to {first_name} {last_name}.

        Style preferences: {style}

        Personalization details:
        {self._format_personalization_details(personalization)}

        Only write the opening message. Be specific, authentic, and
concise."""

        opening_message = call_llm(prompt)
        return opening_message
```

The system uses the `call_llm` utility function which can be configured to use different AI models like Claude or GPT models from OpenAI. This allows you to experiment with different LLMs to find the one that creates the most effective ope for your specific use case.

Customizing for Your Needs

The real power of this system is in the personalization factors you define. Here are some effective examples:

For Job Seeke

- **Recent company**)
  discuss how my

- **Shared alma mater**: "As a fellow [University] alum, I was excited to see your w on [Project]."
- **Mutual connection**: "I noticed we're both connected to [Name]. I've worked w them on [Project] and they spoke highly of your team."

For Sales Professionals:

- **Pain points**: "I noticed from your recent interview that [Company] is facing challenges with [Problem]. We've helped similar companies solve this by..."
- **Growth initiatives**: "Congratulations on your expansion into [Market]. Our solution has helped similar companies accelerate growth in this area by..."
- **Competitor mentions**: "I saw you mentioned working with [Competitor] in th past. Many of our clients who switched from them found our approach to [Fea more effective because..."

For Founders:

- **Investment thesis alignment**: "Your recent investment in [Company] caught r attention. Our startup is also focused on [Similar Space], but with a unique approach to..."
- **Industry challenges**: "I read your thoughts on [Industry Challenge] in [Publication]. We're building a solution that addresses this exact issue by..."
- **Shared vision**: "Your talk at [Conference] about [Topic] resonated with me. We building technology that aligns with your vision of [Vision]..."

Tips for Better Results

Here are some tips for getting the best results from the system:

1. **Be specific with** YCombinator"
2. **Test different pe** on the person

3. **Refine your style preferences**: The more specific your style guidance, the bett the results
4. **Review and edit**: AI-generated openers are a starting point, not the final prod
5. **A/B test**: Try different approaches and track which ones get better responses

Conclusion: Beyond Cold Outreach

While we've focused on cold outreach openers, the same approach can be used for

- Personalizing follow-ups after meetings
- Crafting tailored proposals based on prospect research
- Creating customized content that resonates with specific audience segments
- Building detailed prospect profiles for your sales team

The possibilities are endless when you combine AI with thoughtful personalizatio strategies.

The key is striking the right balance: using AI to scale your outreach without losir the human touch that makes connections meaningful.

Want to explore the full code? Check out the GitHub repository.

Have questions or want to share your results? Leave a comment below!

← Previous ext

Discussion about this post

Comments Restacks

Write a comment...