build-whisper-prompt.py

```
#!/usr/bin/env python3
"""
build-whisper-prompt.py
Extracts primary keys from a TranscriptOMatic YAML meta file and writes
them one per line to a .prompt file for use as a whisper vocabulary hint.

The output is a starting point — edit it manually to remove common words
that don't benefit from hinting and to stay within Whisper's ~224 token limit.

Usage:
    python3 build-whisper-prompt.py --game <slug>

    YAML:   resolved from <script-dir>/../meta/<slug>.yaml
    Output: <script-dir>/../meta/<slug>.prompt

    Run from anywhere; paths are relative to the script location.

Options:
    --game SLUG     Game slug (required)
    --force         Overwrite existing .prompt file (default: abort if exists)
"""

import sys
import yaml
import argparse
from pathlib import Path



# Sections to extract primary keys from, in priority order.
# Terms and locations first — most phonetically unusual for Whisper.
SECTIONS = ["terms", "surnames", "locations", "characters", "groups", "phrases", "players",
"gm"]



def load_yaml(path):
```

```
    with open(path, encoding="utf-8") as f:
        return yaml.safe_load(f)


def extract_keys(data):
    """Extract primary keys and titles from all relevant sections."""
    keys = []
    seen = set()

    def add(term):
        clean = str(term).split("(")[0].strip()
        if clean and clean not in seen:
            keys.append(clean)
            seen.add(clean)

    for section in SECTIONS:
        block = data.get(section, {}) or {}
        if not isinstance(block, dict):
            continue
        for key, entry in block.items():
            add(key)
            if isinstance(entry, dict):
                # Titles: phonetically unusual, benefit from hinting
                for title in (entry.get("titles") or []):
                    add(title)
                # English name: intentional alternate identity, include for Whisper awareness
                name_en = entry.get("name_en")
                if name_en:
                    add(name_en)
    return keys


def main():
    script_dir = Path(__file__).resolve().parent
    meta_dir = (script_dir / ".." / "meta").resolve()

    parser = argparse.ArgumentParser(
        description="Generate a whisper vocabulary prompt file from a YAML meta file."
    )
    parser.add_argument("--game", required=True, metavar="SLUG",
```

```
                        help="Game slug — resolves to meta/<slug>.yaml")
    parser.add_argument("--force", action="store_true",
                        help="Overwrite existing .prompt file")
    args = parser.parse_args()

    yaml_path = meta_dir / f"{args.game}.yaml"
    prompt_path = meta_dir / f"{args.game}.prompt"

    if not yaml_path.exists():
        print(f"□ YAML not found: {yaml_path}", file=sys.stderr)
        sys.exit(1)

    if prompt_path.exists() and not args.force:
        print(f"□ Prompt file already exists: {prompt_path}", file=sys.stderr)
        print( "   Use --force to overwrite.", file=sys.stderr)
        sys.exit(1)

    data = load_yaml(yaml_path)
    keys = extract_keys(data)

    prompt_path.write_text("\n".join(keys) + "\n", encoding="utf-8")

    print(f"□□YAML:   {yaml_path}")
    print(f"□ Written: {prompt_path}")
    print(f"   {len(keys)} terms — edit to remove unproblematic entries")
    print(f"   then check token count (target: <224 tokens)")


if __name__ == "__main__":
    main()
```

Created 2026-04-12 01:46:34 UTC by Mela
Updated 2026-04-12 11:00:16 UTC by Mela