summarize-meeting.sh

```
#!/usr/bin/env bash
set -euo pipefail

# ------------------------------------------------------------
# summarize-meeting.sh
# Post-session summarization on Mac using ollama
# Usage: summarize-meeting.sh [--de|--en] [--game <slug>] [<transcript.txt>]
#        Defaults to English, most recent transcript if none given
# ------------------------------------------------------------

OLLAMA_MODEL="${OLLAMA_MODEL:-gemma3:27b}"
BASE="$HOME/Syncthing/TranscriptOMatic/recordings"
META_DIR="$(cd "$(dirname "$0")/../meta" 2>/dev/null && pwd || echo
"$HOME/Syncthing/TranscriptOMatic/meta")"
LANG_MODE="en"
TRANSCRIPT=""
GAME_SLUG=""
USE_CONTEXT=0

# ------------------------------------------------------------
# Argument parsing
# ------------------------------------------------------------

while [[ $# -gt 0 ]]; do
  case "$1" in
    --de)     LANG_MODE="de"; shift ;;
    --en)     LANG_MODE="en"; shift ;;
    --game)       GAME_SLUG="$2"; shift 2 ;;
    --context)    USE_CONTEXT=1; shift ;;
    -*)
      echo "Usage: summarize-meeting [--de|--en] [--game <slug>] [--context]
[<transcript.txt>]" >&2
      exit 2
      ;;
    *)
```

```
      TRANSCRIPT="$1"; shift ;;
  esac
done

# ------------------------------------------------------------
# Find transcript
# ------------------------------------------------------------

if [[ -z "$TRANSCRIPT" ]]; then
  # Prefer normalized transcript; fall back to plain transcript
  TRANSCRIPT="$(ls -t "$BASE"/**/*_normalized.txt 2>/dev/null | head -n1 || true)"
  if [[ -z "$TRANSCRIPT" ]]; then
    TRANSCRIPT="$(ls -t "$BASE"/**/*_transcript.txt 2>/dev/null | head -n1 || true)"
  fi
  if [[ -z "$TRANSCRIPT" ]]; then
    echo "□ No transcript found in $BASE" >&2
    echo "   Usage: summarize-meeting [--de|--en] [--game <slug>] [<transcript.txt>]" >&2
    exit 1
  fi
fi

if [[ ! -f "$TRANSCRIPT" ]]; then
  echo "□ File not found: $TRANSCRIPT" >&2
  exit 1
fi

# If a plain transcript was given explicitly, check whether a normalized version exists
if [[ "$TRANSCRIPT" == *_transcript.txt ]]; then
  NORMALIZED="${TRANSCRIPT/_transcript.txt/_transcript_normalized.txt}"
  if [[ -f "$NORMALIZED" ]]; then
    echo "□□  Using normalized transcript: $NORMALIZED"
    TRANSCRIPT="$NORMALIZED"
  fi
fi

# ------------------------------------------------------------
# Auto-detect game slug from transcript filename if not given
# ------------------------------------------------------------

if [[ -z "$GAME_SLUG" ]]; then
```

```
  TRANSCRIPT_BASENAME="$(basename "$TRANSCRIPT")"
  # Try to find a matching meta file by checking if any slug appears in the filename
  for META_FILE in "$META_DIR"/*.yaml; do
    [[ -f "$META_FILE" ]] || continue
    CANDIDATE_SLUG="$(basename "$META_FILE" .yaml)"
    if [[ "$TRANSCRIPT_BASENAME" == *"$CANDIDATE_SLUG"* ]]; then
      GAME_SLUG="$CANDIDATE_SLUG"
      break
    fi
  done
fi

# ------------------------------------------------------------
# Load meta file if available
# ------------------------------------------------------------

META_CONTEXT=""
META_FILE=""

if [[ -n "$GAME_SLUG" && "$USE_CONTEXT" == "1" ]]; then
  META_FILE="$META_DIR/${GAME_SLUG}.yaml"
  if [[ -f "$META_FILE" ]]; then
    META_CONTEXT="$(uv run --with pyyaml python3 "$(dirname "$0")/build-summary-context.py" --
game "$GAME_SLUG")"
    echo "□□Meta:       $META_FILE (filtered)"
  else
    echo "⚠□  No meta file found for slug '$GAME_SLUG' in $META_DIR" >&2
  fi
else
  echo "□□  Running without context document (use --context to include)"
fi

# ------------------------------------------------------------
# Output paths
# ------------------------------------------------------------

SESSION="$(dirname "$TRANSCRIPT")"
TRANSCRIPT_BASE="$(basename "$TRANSCRIPT" .txt)"
SUMMARY="$SESSION/${TRANSCRIPT_BASE}_summary.md"
```

```
# ------------------------------------------------------------
# Skip if summary already exists (use FORCE=1 to override)
# ------------------------------------------------------------

if [[ -f "$SUMMARY" && "${FORCE:-}" != "1" ]]; then
  echo "□□  Summary already exists, skipping: $SUMMARY"
  echo "   Use FORCE=1 summarize-meeting to overwrite."
  exit 0
fi

# ------------------------------------------------------------
# Language-specific prompt
# ------------------------------------------------------------

case "$LANG_MODE" in
  en) PROMPT_LANG="Write the summary in English." ;;
  de) PROMPT_LANG="Schreibe die Zusammenfassung auf Deutsch." ;;
esac

echo "□□Transcript: $TRANSCRIPT"
echo "□□Model:      $OLLAMA_MODEL"
echo "□□□ Language:   $LANG_MODE"
echo "□□Summary:    $SUMMARY"
echo "----"

# ------------------------------------------------------------
# Build prompt
# ------------------------------------------------------------

if [[ -n "$META_CONTEXT" ]]; then
  CONTEXT_BLOCK="REFERENCE DOCUMENT (metadata only — not part of the session transcript):
This YAML document provides background context for correctly interpreting the transcript
below.
It is NOT a session log or discussion. Do NOT summarize or reference its contents as in-game
or out-of-game events.
Use it exclusively to:
- Correctly spell and identify character names, player names, locations and in-game terms
- Understand roles, group memberships and character relationships
- The 'lang' field (e.g. ga, fr) indicates language origin of a name — it is metadata, not a
topic of discussion
```

```
- Use the 'short' name for characters in continuous prose; use full names only when
introducing a character
- The 'aliases' field lists transcription variants of a name — use only the primary key or
short name in the summary
- If a character appears in the transcript under an alias, identify them by their short name

Reference document:
---
${META_CONTEXT}
---
END OF REFERENCE DOCUMENT

The transcript follows below. Summarize only what is in the transcript.
"
else
  CONTEXT_BLOCK=""
fi

# ------------------------------------------------------------
# Run summarization
# ------------------------------------------------------------

ollama --nowordwrap run "$OLLAMA_MODEL" <<EOF > "$SUMMARY"
You are an expert note-taker for tabletop roleplaying game sessions.
The transcript is a recording of a TTRPG session and contains both in-character roleplay and
out-of-character table talk.

${CONTEXT_BLOCK}
Rules:
- Clearly distinguish between in-character events and out-of-character discussion
- Quote spoken statements in their original language
- ${PROMPT_LANG}
- Use character short names (as provided in context) rather than full names where natural
- ONLY summarize events, decisions and facts that are explicitly stated in the transcript
- If you are not certain something happened, omit it entirely — do NOT infer, imply or
extrapolate
- Pay close attention to who does what: do not invert agency (e.g. who challenges whom, who
saves whom)
- Do not conflate events from different sessions; only summarize what happens in this
transcript
```

```
- Subtext and player motivation matter: note what characters say and do, not what the model thinks they mean

Deliver:
1) Session overview (3-5 sentences summarising the main in-game events)
2) Key in-game decisions and developments
3) Important character moments (emotional beats, revelations, relationship shifts)
4) Highlights (memorable quotes, unexpected twists, standout scenes)
5) Notable out-of-character moments (rules discussions, retcons, player notes)
6) Cliffhangers and open threads going into the next session
7) Characters introduced or significantly developed this session

Transcript:
$(cat "$TRANSCRIPT")
EOF

echo "□ Summary written to $SUMMARY"
```

Created 2026-04-12 11:51:03 UTC by Mela
Updated 2026-04-14 15:26:01 UTC by Mela