

LLM Engineering

MASTER AI & LARGE LANGUAGE MODELS



Sleeves rolled up

What you can now do

- Confidently code with Frontier Models
- Build a multi-modal AI Assistant with Tools
- Use HuggingFace pipelines for a wide variety of inference tasks

Today we get into the lower level HuggingFace API

- Create tokenizers for models
- Translate between text and tokens
- Understand special tokens and chat templates

Introducing the Tokenizer

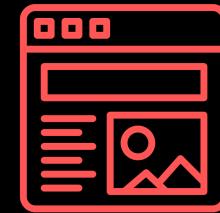
Maps between Text and Tokens for a particular model



Translates between Text and Tokens
with `encode()` and `decode()`
methods



Contains a Vocab that can include
special tokens to signal information
to the LLM, like start of prompt



Can include a Chat Template that
knows how to format a chat
message for this model

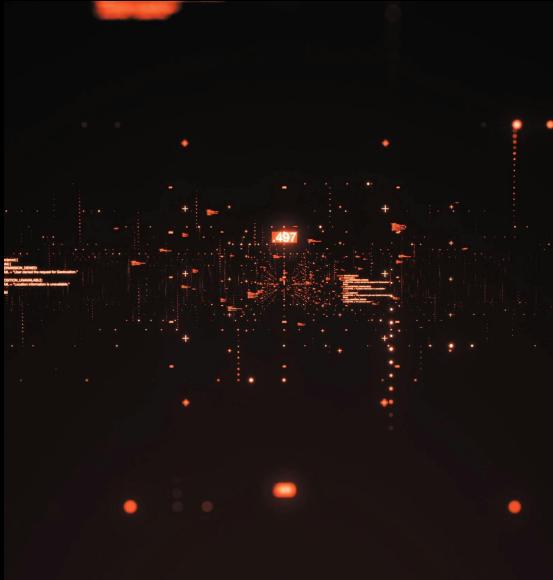
The Tokenizers for key models

We will experiment with tokenizers for a variety of open-source models



Llama 3.1

Meta led the way



Phi 3

Microsoft's entrant



Qwen2

Leader from Alibaba Cloud



Starcoder2

Coding model

```
        message: 'Could not register user. Error: ' + err.message);
endgrid.SendgridAPIKey);
/emailtemplate.html", "utf8", (err, data) => {
  if (err) {
    res.status(500).send(`Error sending email: ${err.message}`);
  } else {
    res.status(200).send(data);
  }
}

// Handle account confirmation
app.post('/api/accounts/confirm', (req, res) => {
  const { token } = req.query;
  if (!token) {
    return res.status(400).send('Token is required');
  }

  const decodedToken = jwt.decode(token);
  if (!decodedToken || !decodedToken.id) {
    return res.status(401).send('Invalid token');
  }

  const user = await User.findById(decodedToken.id);
  if (!user) {
    return res.status(404).send('User not found');
  }

  user.isConfirmed = true;
  user.save();
  res.status(200).send('Account confirmed successfully');
});

// Handle password reset
app.post('/api/accounts/reset', (req, res) => {
  const { token } = req.query;
  if (!token) {
    return res.status(400).send('Token is required');
  }

  const decodedToken = jwt.decode(token);
  if (!decodedToken || !decodedToken.id) {
    return res.status(401).send('Invalid token');
  }

  const user = await User.findById(decodedToken.id);
  if (!user) {
    return res.status(404).send('User not found');
  }

  user.resetToken = generateResetToken();
  user.resetExpires = Date.now() + 30 * 60 * 1000;
  user.save();
  res.status(200).send('Reset token sent to your email');
});

// Handle password change
app.put('/api/accounts/change', (req, res) => {
  const { token, password } = req.query;
  if (!token || !password) {
    return res.status(400).send('Token and password are required');
  }

  const decodedToken = jwt.decode(token);
  if (!decodedToken || !decodedToken.id) {
    return res.status(401).send('Invalid token');
  }

  const user = await User.findById(decodedToken.id);
  if (!user) {
    return res.status(404).send('User not found');
  }

  const isPasswordValid = await bcrypt.compare(password, user.password);
  if (!isPasswordValid) {
    return res.status(401).send('Incorrect password');
  }

  user.password = generatePassword();
  user.save();
  res.status(200).send('Password changed successfully');
});

// Handle password recovery
app.post('/api/accounts/recover', (req, res) => {
  const { email } = req.body;
  if (!email) {
    return res.status(400).send('Email is required');
  }

  const user = await User.findOne({ email });
  if (!user) {
    return res.status(404).send('User not found');
  }

  const token = generateResetToken();
  user.resetToken = token;
  user.resetExpires = Date.now() + 30 * 60 * 1000;
  user.save();
  res.status(200).send(`Reset token sent to ${email}`);
});

// Handle password recovery
app.get('/api/accounts/recover/:token', (req, res) => {
  const token = req.params.token;
  const user = await User.findOne({ resetToken: token });
  if (!user) {
    return res.status(404).send('User not found');
  }

  const isTokenValid = jwt.verify(token, process.env.RESET_TOKEN_SECRET);
  if (!isTokenValid) {
    return res.status(401).send('Invalid token');
  }

  const password = generatePassword();
  user.password = password;
  user.resetToken = null;
  user.resetExpires = null;
  user.save();
  res.status(200).send(`Password reset successful. Your new password is ${password}`);
});

// Handle password recovery
app.get('/api/accounts/recover/:token', (req, res) => {
  const token = req.params.token;
  const user = await User.findOne({ resetToken: token });
  if (!user) {
    return res.status(404).send('User not found');
  }

  const isTokenValid = jwt.verify(token, process.env.RESET_TOKEN_SECRET);
  if (!isTokenValid) {
    return res.status(401).send('Invalid token');
  }

  const password = generatePassword();
  user.password = password;
  user.resetToken = null;
  user.resetExpires = null;
  user.save();
  res.status(200).send(`Password reset successful. Your new password is ${password}`);
});
```

Ready for transformers..

What you can now do

- Confidently code with Frontier Models
- Build a multi-modal AI Assistant with Tools
- Use HuggingFace pipelines and tokenizers

After next time you will be able to use the powerful APIs

- Work with HuggingFace lower level APIs
- Use HuggingFace models to generate text
- Compare the results across 5 open source models