STUDENT HANDOUT · MODULE RESOURCE

Reading vs. Meaning

Why you can read a broken sentence — and a language model often cannot

Start with an experiment on yourself

Read the line below. Do not think about it. Just read it:

```
Th qck brwn fx jmps vr th lzy dg
```

You almost certainly read *"The quick brown fox jumps over the lazy dog"* — even though more than a third of the letters are missing. Now notice something strange: you did not *decode* that sentence letter by letter. You *recognised* it. Your brain guessed the meaning first and filled in the missing letters afterwards.

That is the difference this handout is about. **Humans read for meaning. Language models read tokens.** Understanding that one difference explains a whole family of strange AI behaviours — and it teaches you something about how *you* learn, too.

How your brain reads: top-down prediction

Your brain is a prediction machine. When you read, it does not passively receive letters; it constantly guesses what should come next, using everything you know about language, the world, and the situation. Psychologists call this **top-down processing**: meaning flows down from your expectations and repairs the raw input. That is why you can read degraded text, understand a mumbled sentence at a noisy party, or finish a friend's sentence before they do.

Try one more. Fill in the blanks without thinking:

```
Jeg skal hente børnene i børneh____ klokken f____ten
```

If you speak Danish, you wrote *børnehaven* and *femten* instantly — not because you analysed the letters, but because you know how everyday life works. Children get picked up from kindergarten, and pick-up times are on the hour. Your **world knowledge** repaired the text. No letter-counting was involved.

How a language model reads: tokens

A large language model (LLM) never sees letters the way you do. Before your message reaches the model, it is chopped into **tokens** — chunks of characters that frequently appear together in the model's training data. Common words become a single token; rare words are split into several fragments:

Text	What the model sees (roughly)
understanding	`[understanding]` — 1 token
forståelse	`[for][stå][else]` — 3 tokens

udnersådtnlese *(scrambled)*	**`[ud][ner][så][dt][nl][ese]`** — fragments

The idea to hold on to: the model's smallest unit of “sight” is the token, not the letter. When you ask an LLM how many r's are in a word, it is a bit like asking you how many brush strokes are in a painting you are looking at from across the room. The information is technically *in there*, but it is not the level at which the system naturally perceives.

The interesting part: both are prediction machines

Here is the twist that makes this more than a party trick. The LLM is *also* a prediction machine — that is genuinely how it works. It predicts the most likely next token, just as your brain predicts the most likely next word. So why does it fail where you succeed?

Because it predicts at the **wrong level** for this task. When text is degraded — vowels removed, letters scrambled, words reversed — the input shatters into rare, near-random token fragments the model has hardly ever seen. Its predictions have nothing solid to stand on. Your brain, meanwhile, drops down to the letter level when it needs to, then jumps back up to meaning. You can move between levels. The model, by itself, largely cannot.

Examples to try (and why they work)

1 · Missing vowels — meaning survives, tokens do not

```
Th qck brwn fx jmps vr th lzy dg
Dn lll hvfr lggr Kbnhvn (Den lille havfrue ligger i København)
```

You recognise the whole; the model receives fragments. Note that the very newest models often solve the English version — the famous examples appear in their training data. The Danish version is more reliable as a demo, for reasons explained below.

2 · Scrambled middles — the “Cambridge” effect

```
Ifgøle en usnørdeglese kan mneneskser lsæe orrd sevlom
bgostavrene er bladnet
```

Humans read words by their outer shape and context, so keeping the first and last letters is usually enough. The English version of this meme is so famous that models have memorised it — which is itself a lesson: a model can look like it reads, when it actually remembers.

3 · Context repair — world knowledge, not letters

```
Jeg skal hente børnene i børneh____ klokken f____ten
```

The blanks cannot be filled by spelling rules alone. You need to know how life works. This is the purest demonstration that human reading is meaning-first.

4 · Stacked degradations — the reliable failure

```
grd yzl ht rv spmj xf nwrb kcq hT
```

Reversed *and* vowel-stripped. Models have learned each trick separately from internet examples, but combinations produce token sequences too rare to have been memorised. Effortful for you — but solvable. For the model, usually

not.

Why Danish costs more than English

Tokenizers are built by scanning enormous amounts of text and merging the most frequent character sequences into single tokens. That training text is dominated by English — so English gets the efficient, whole-word tokens. Danish pays a price three times over:

1. Less training data. Danish sequences appear rarely, so fewer Danish words earned their own token. *Understanding* is one token; *forståelse* is three.

2. Special characters. æ, ø and å are rare in the global training mix, so words often split exactly at those letters. *Kærlighed* fragments where *love* does not.

3. Compound words. Danish glues concepts together — *arbejdsmarkedsuddannelse* — which guarantees multi-token splits.

The practical consequence: the same prompt in Danish typically uses noticeably more tokens than in English. That means higher cost, faster use of the model's context window, and sometimes slightly weaker performance — because the model's "units of thought" line up less cleanly with Danish word boundaries. This is also why scrambled *Danish* text is a more reliable demo than scrambled English: the fragments are rarer, so the model has less memory to lean on.

What this teaches you about learning

Notice what happened when you read the degraded sentences: it took **effort**. Your brain had to work — generate a guess, test it against the letters, revise, try again. And precisely *because* it took effort, you will remember it. Learning scientists call this a **desirable difficulty**: you learn by struggling to make sense of something you do not yet understand. Meaning that arrives without effort tends to slide straight through.

This is the deeper reason the exercise belongs in this course. It is not just a fact about tokenizers — it is a demonstration, running live in your own head, of what understanding actually is: an active construction, not a passive download. An LLM that fluently completes your sentence has predicted well. Whether anything was *understood* is a different question — and now you have felt the difference from the inside.

The durable insight (write this down)

Humans read meaning-first and repair the text. LLMs read token-first and predict the text. Specific failure examples are perishable — models patch them constantly, and the newest model may solve the exact sentence printed here. The token explanation is what lasts. If a demo stops working, that is not the lesson breaking; that is the lesson continuing.

Try it yourself: the companion widget

This handout has an interactive companion in the course. In the widget you can type any sentence — English or Danish — degrade it step by step, and watch a simplified simulation of what the tokenizer sees on the other side. Predict before you click: will the model cope? Then check.

Reflection questions

1. When you read “Th qck brwn fx…”, at what moment did you *know* what it said? Before or after you had checked all the letters? What does that tell you about how you read?

2. A model answers the scrambled Cambridge sentence perfectly, but fails on a scrambled sentence you invented yourself. What is the difference — and why does it matter when judging what an AI “can do”?

3. Danish text costs more tokens than English. Who is affected by that, and in what ways? Think beyond price.

4. Give one example from your own studies where struggling with something confusing taught you more than a clear explanation would have. What does that have in common with the exercise you just did?

A note on honesty: everything on these pages describes a moving target. Frontier models improve at character-level tasks every year, some now route tricky inputs through separate tools, and any specific example may stop failing. The claims here are about *how the technology works*, not about a permanent list of things AI cannot do. Test, observe, update — that is the AI-confident habit.