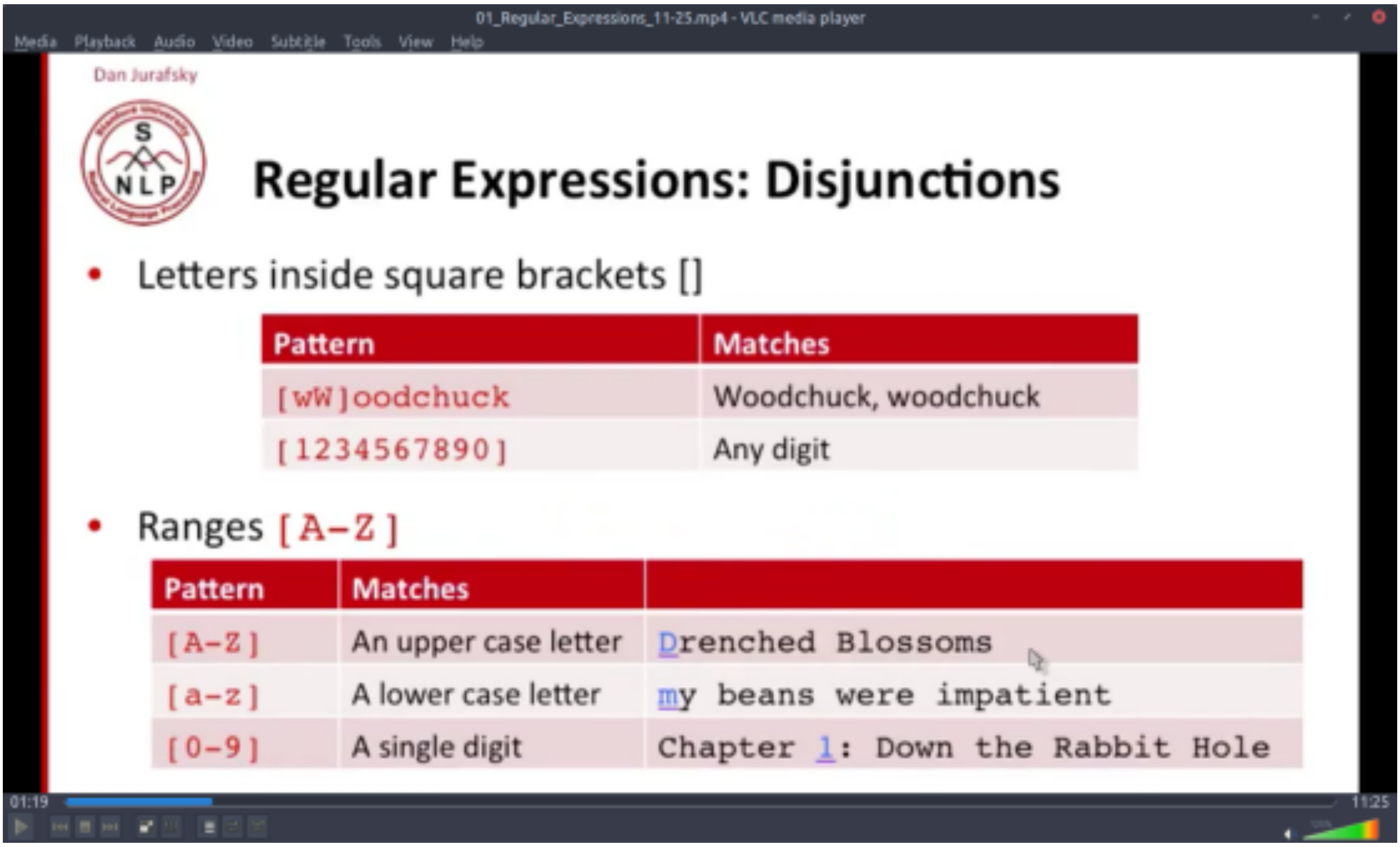
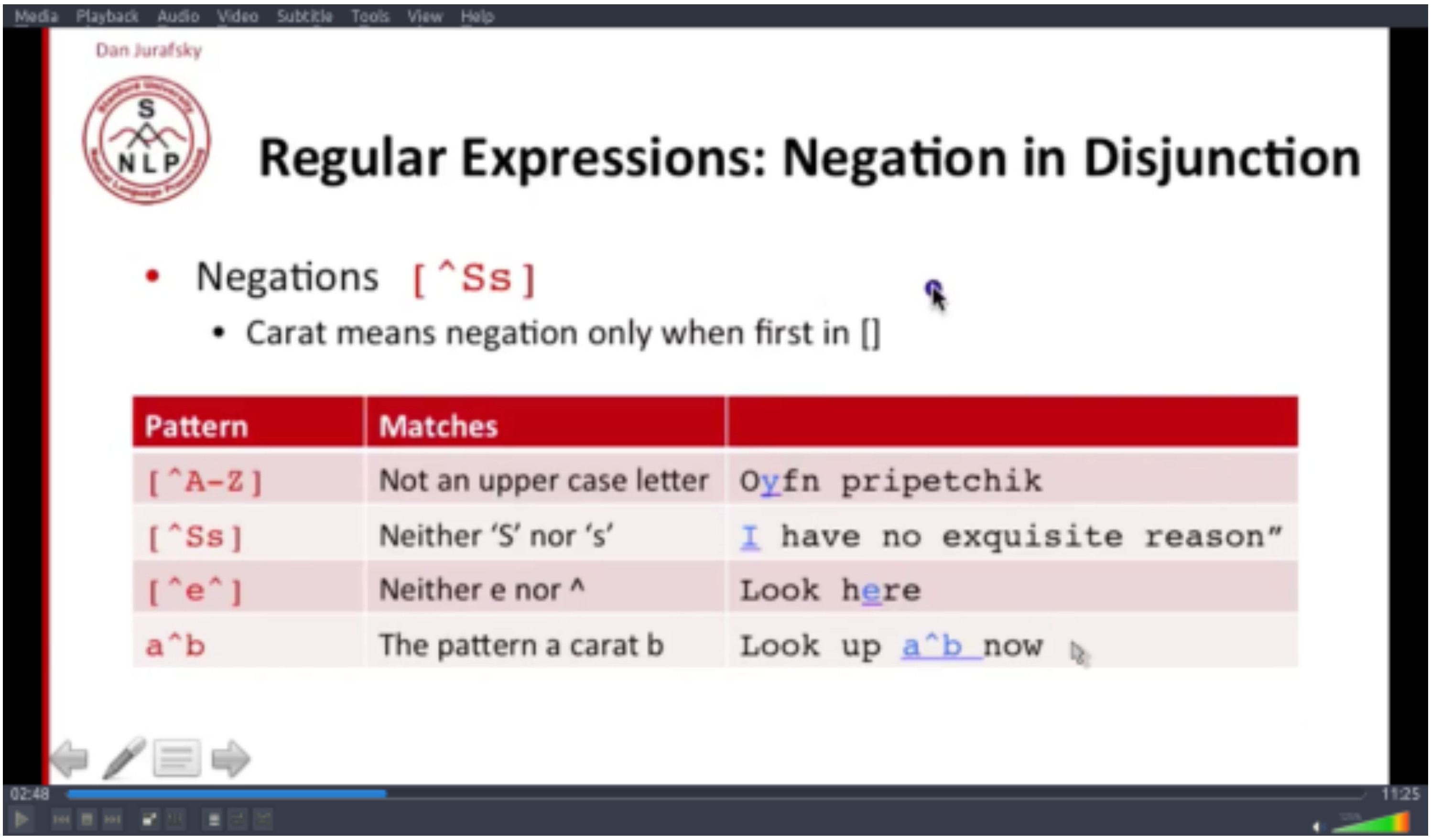
Natural Language Processing

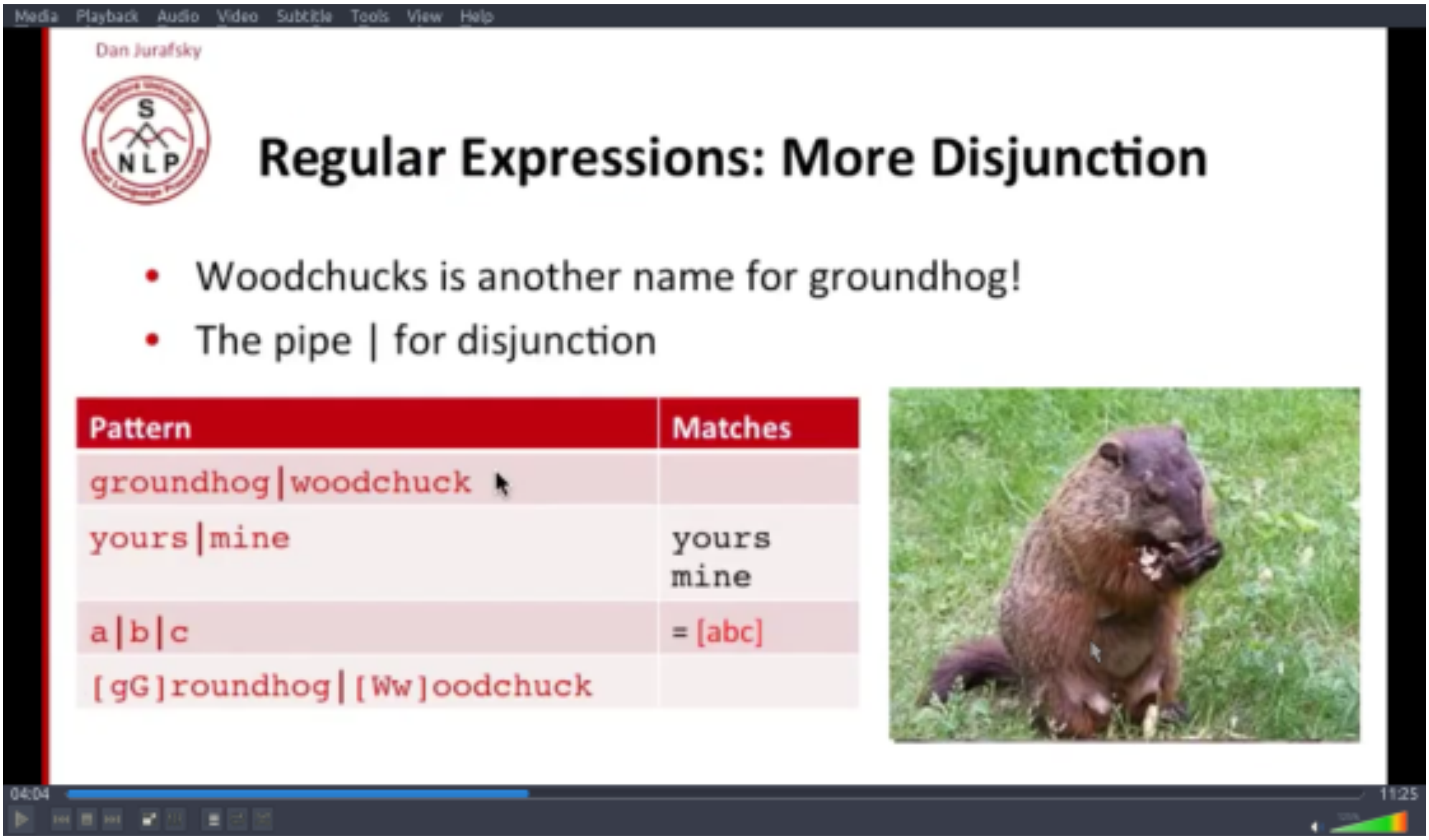
Regular Expressions



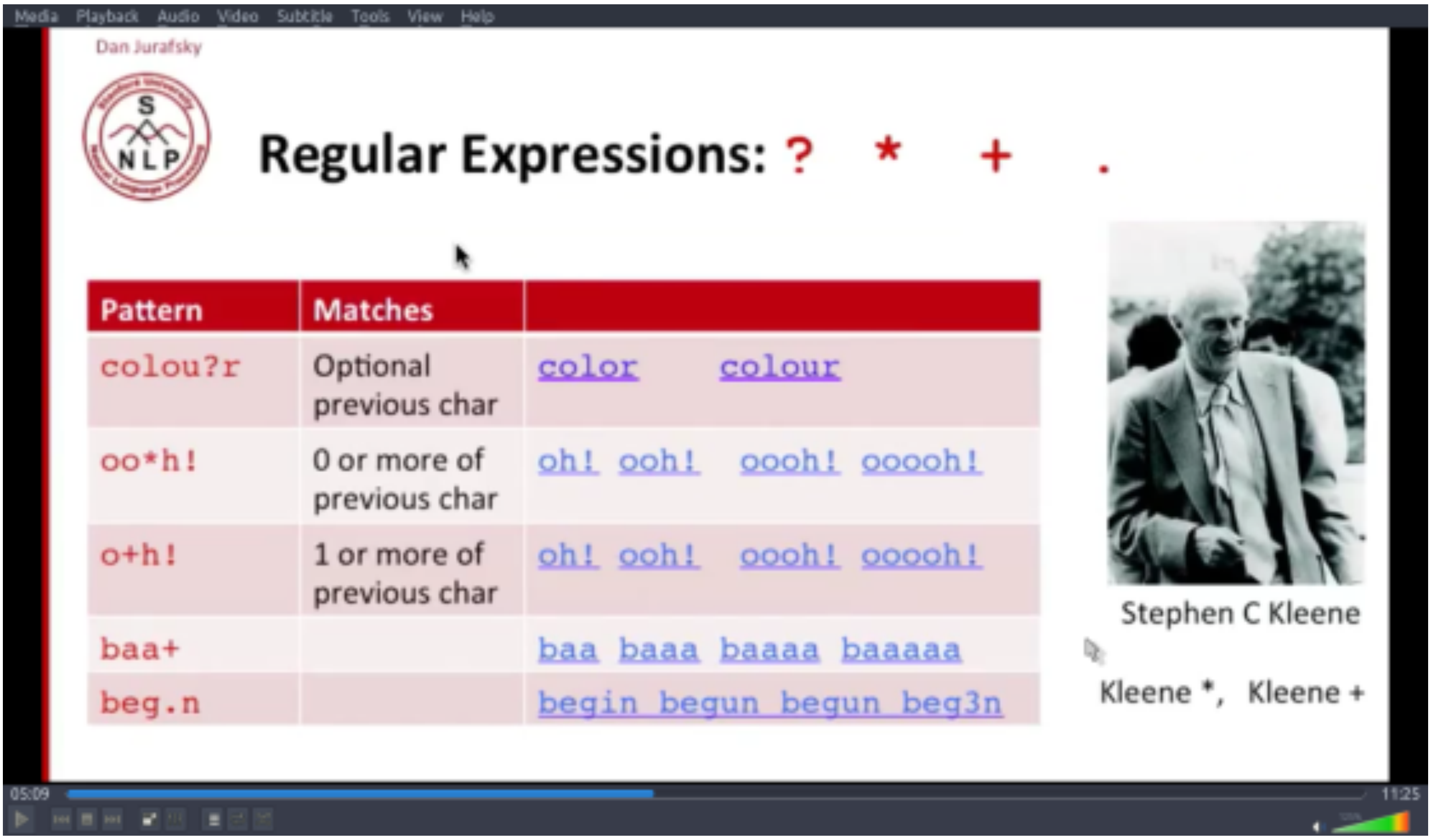


<u>Note</u>:

- “^” inside the bracket and outside the bracket acts differently
- Caret outside the bracket means start of the string

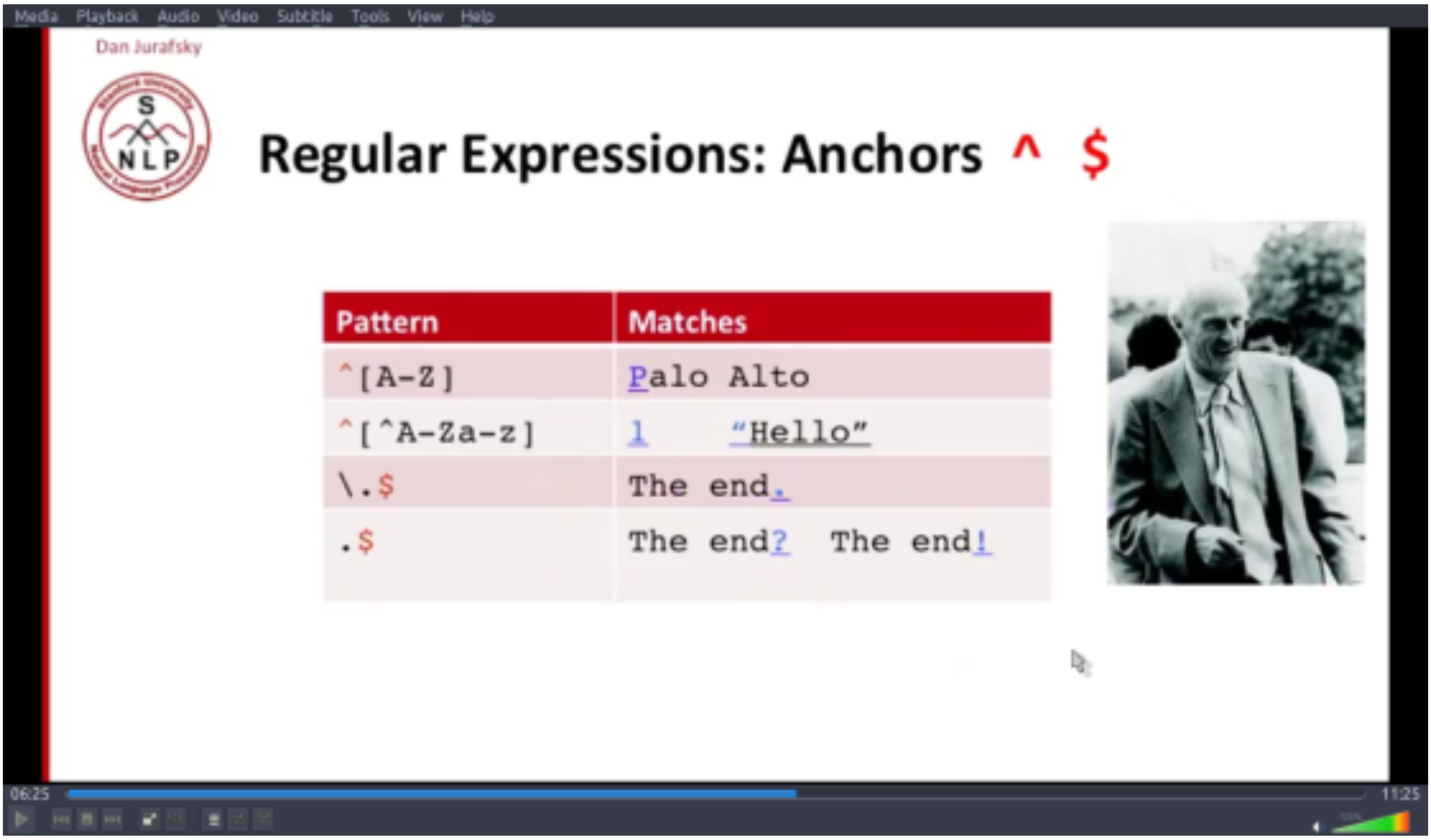


- Pipe “|” stands for or



Note:

- The character in the last example in place of “.” can't be empty



- Caret stands for start of the string
- Dollar stands for end of string

Cheat Sheet

<table>
<tr><th colspan="2">Character classes</th></tr>
<tr><td>.</td><td>any character except newline</td></tr>
<tr><td>\w \d \s</td><td>word, digit, whitespace</td></tr>
<tr><td>\W \D \S</td><td>not word, digit, whitespace</td></tr>
<tr><td>[abc]</td><td>any of a, b, or c</td></tr>
<tr><td>[^abc]</td><td>not a, b, or c</td></tr>
<tr><td>[a-g]</td><td>character between a & g</td></tr>
<tr><th colspan="2">Anchors</th></tr>
<tr><td>^abc$</td><td>start / end of the string</td></tr>
<tr><td>\b</td><td>word boundary</td></tr>
<tr><th colspan="2">Escaped characters</th></tr>
<tr><td>\. * \\</td><td>escaped special characters</td></tr>
<tr><td>\t \n \r</td><td>tab, linefeed, carriage return</td></tr>
<tr><td>\u00A9</td><td>unicode escaped ©</td></tr>
<tr><th colspan="2">Groups & Lookaround</th></tr>
<tr><td>(abc)</td><td>capture group</td></tr>
<tr><td>\1</td><td>backreference to group #1</td></tr>
<tr><td>(?:abc)</td><td>non-capturing group</td></tr>
<tr><td>(?=abc)</td><td>positive lookahead</td></tr>
<tr><td>(?!abc)</td><td>negative lookahead</td></tr>
<tr><th colspan="2">Quantifiers & Alternation</th></tr>
<tr><td>a* a+ a?</td><td>0 or more, 1 or more, 0 or 1</td></tr>
<tr><td>a{5} a{2,}</td><td>exactly five, two or more</td></tr>
<tr><td>a{1,3}</td><td>between one & three</td></tr>
<tr><td>a+? a{2,}?</td><td>match as few as possible</td></tr>
<tr><td>ab|cd</td><td>match ab or cd</td></tr>
</table>

Tokenization

- Fragments ⇒ main- mainly
- Filled pauses ⇒ uh

Dan Jurafsky

How many words?

- I do uh main- mainly business data processing
  - Fragments, filled pauses
- Seuss's cat in the hat is different from other cats!
  - **Lemma**: same stem, part of speech, rough word sense
    - cat and cats = same lemma
  - **Wordform**: the full inflected surface form
    - cat and cats = different wordforms

Dan Jurafsky

How many words?

N = number of tokens

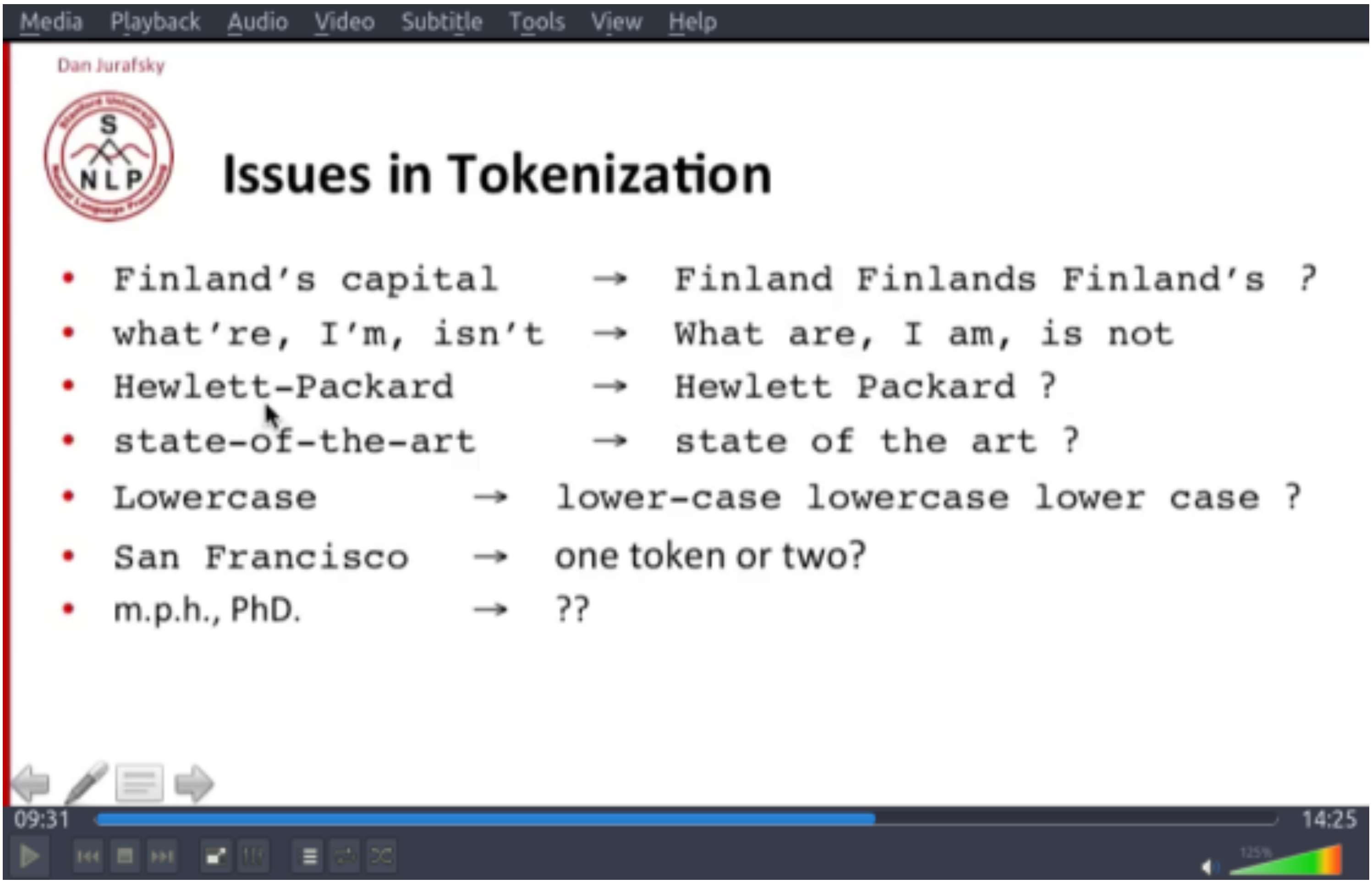
V = vocabulary = set of types

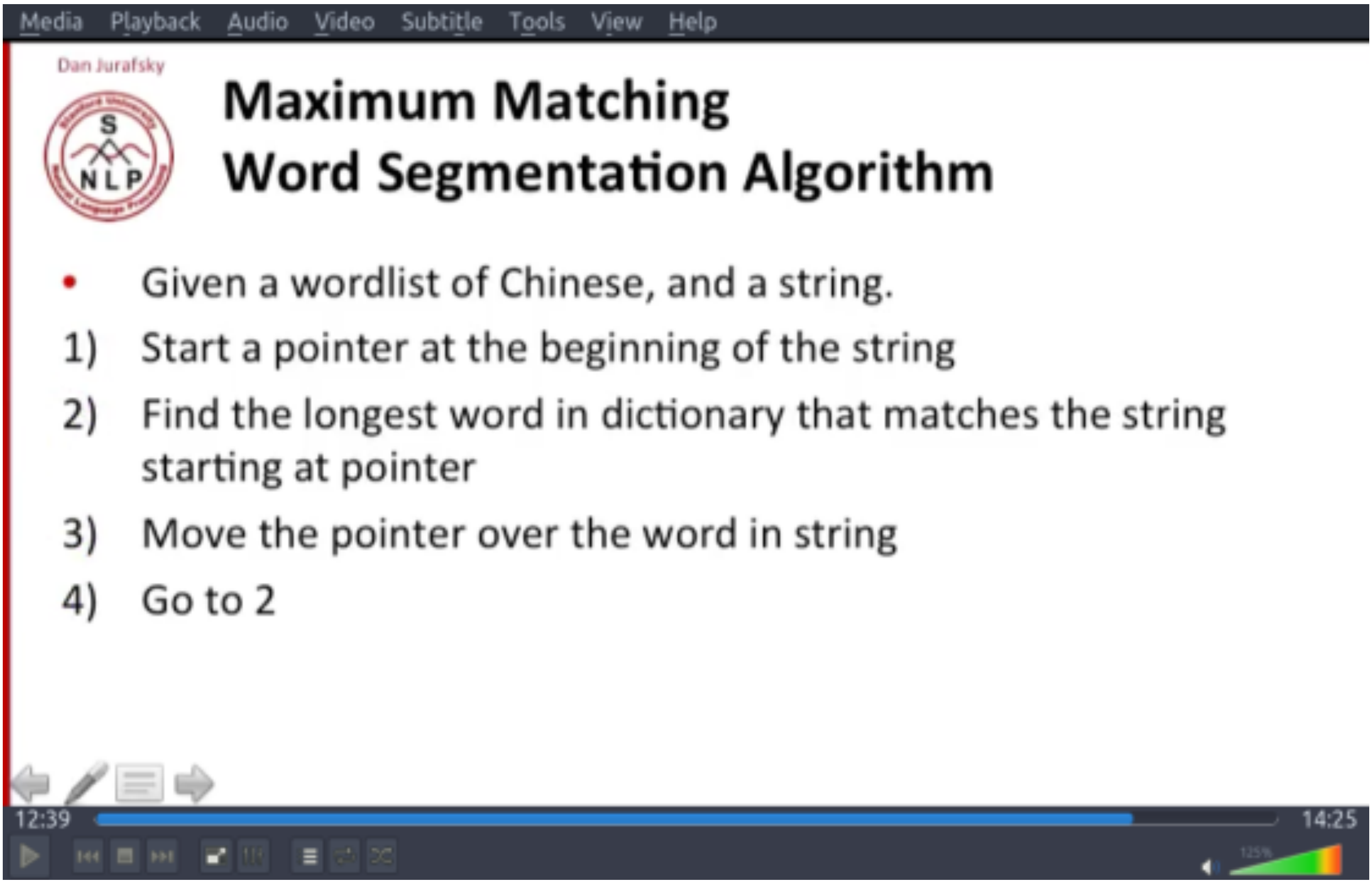
$|V|$ is the size of the vocabulary

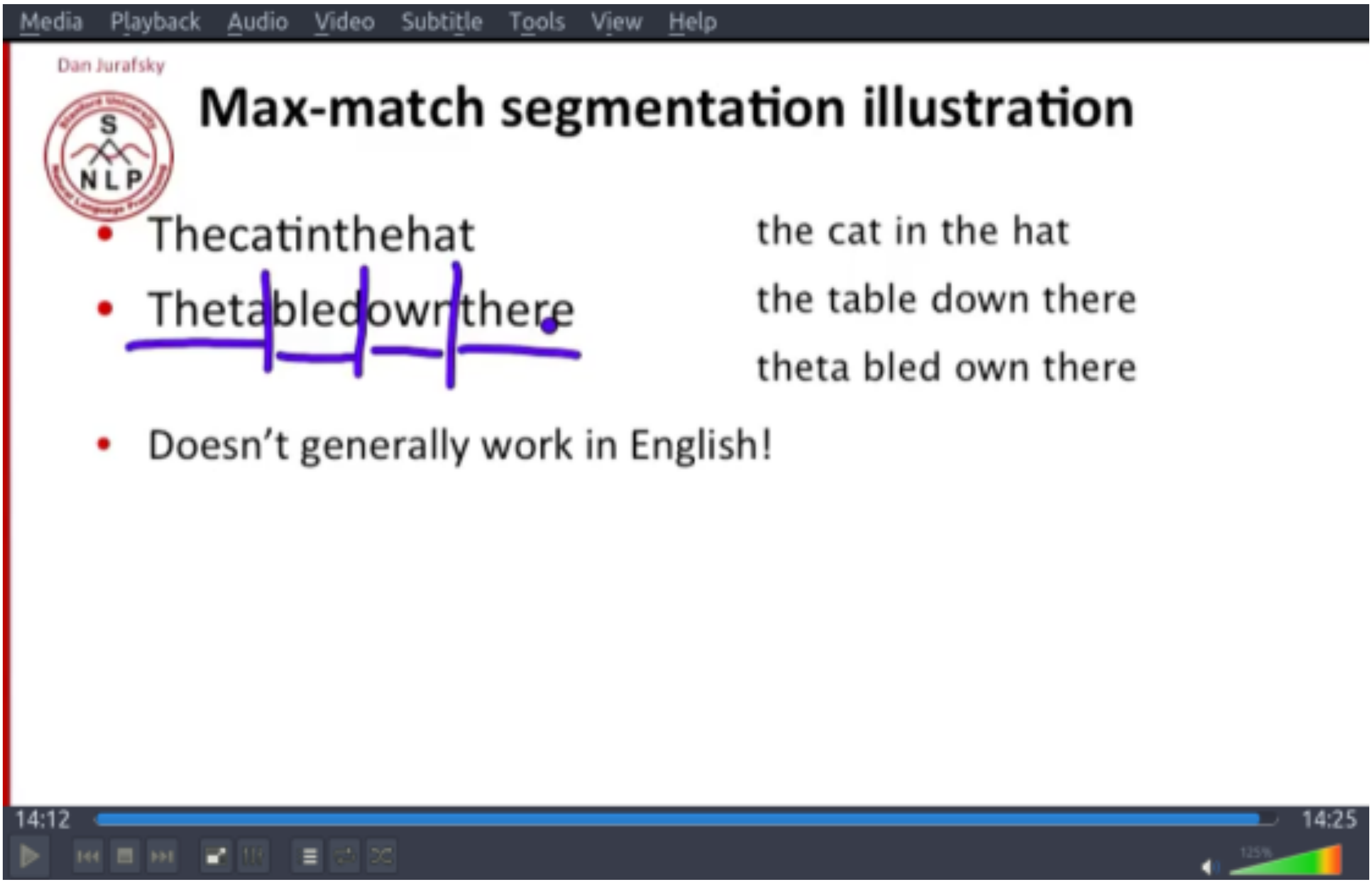
Church and Gale (1990): $|V| > O(N^{½})$

	Tokens = N	Types = |V|
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
Google N-grams	1 trillion	13 million

Size of the vocabulary grows greater than the square root of number of tokens

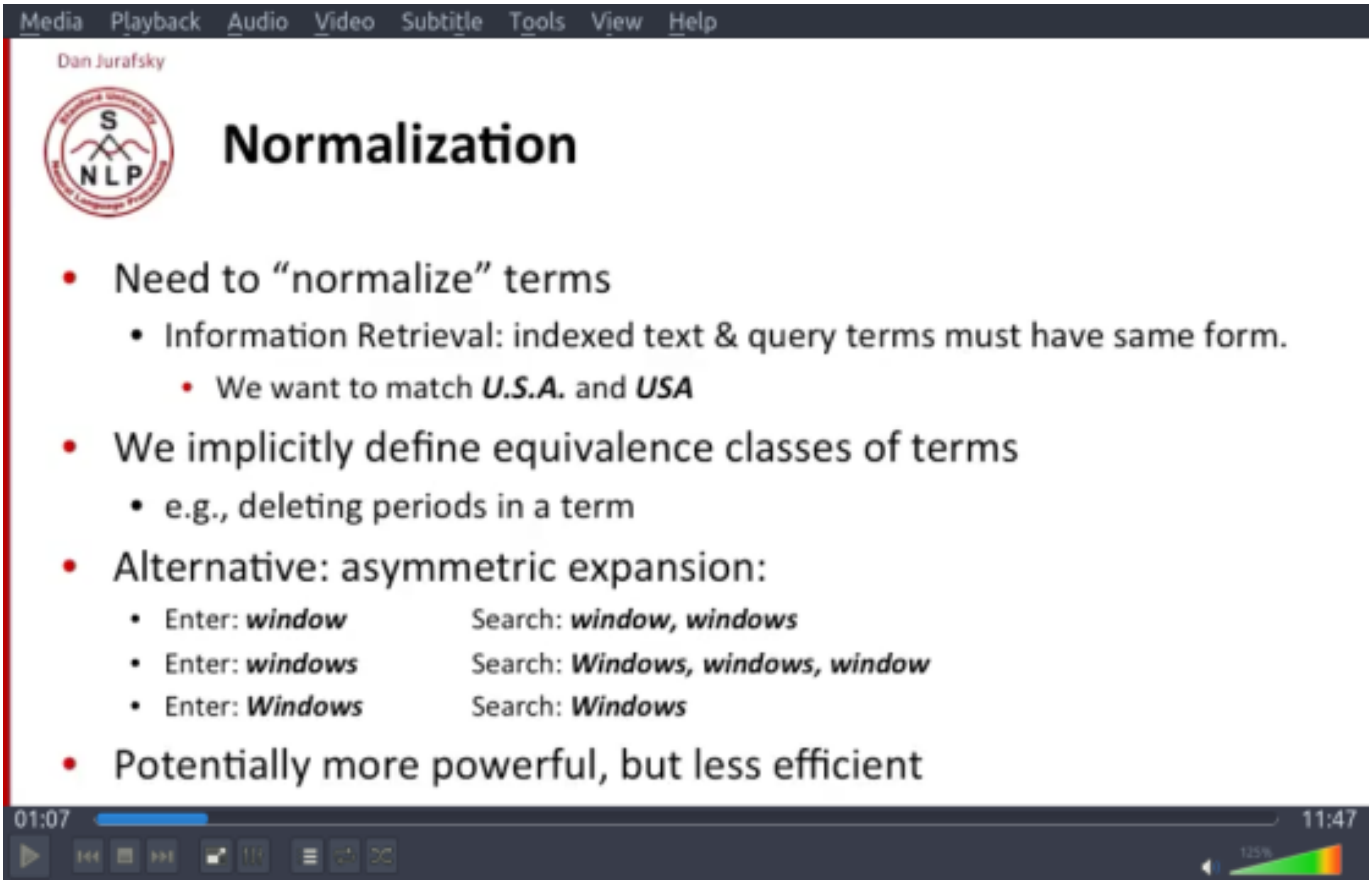






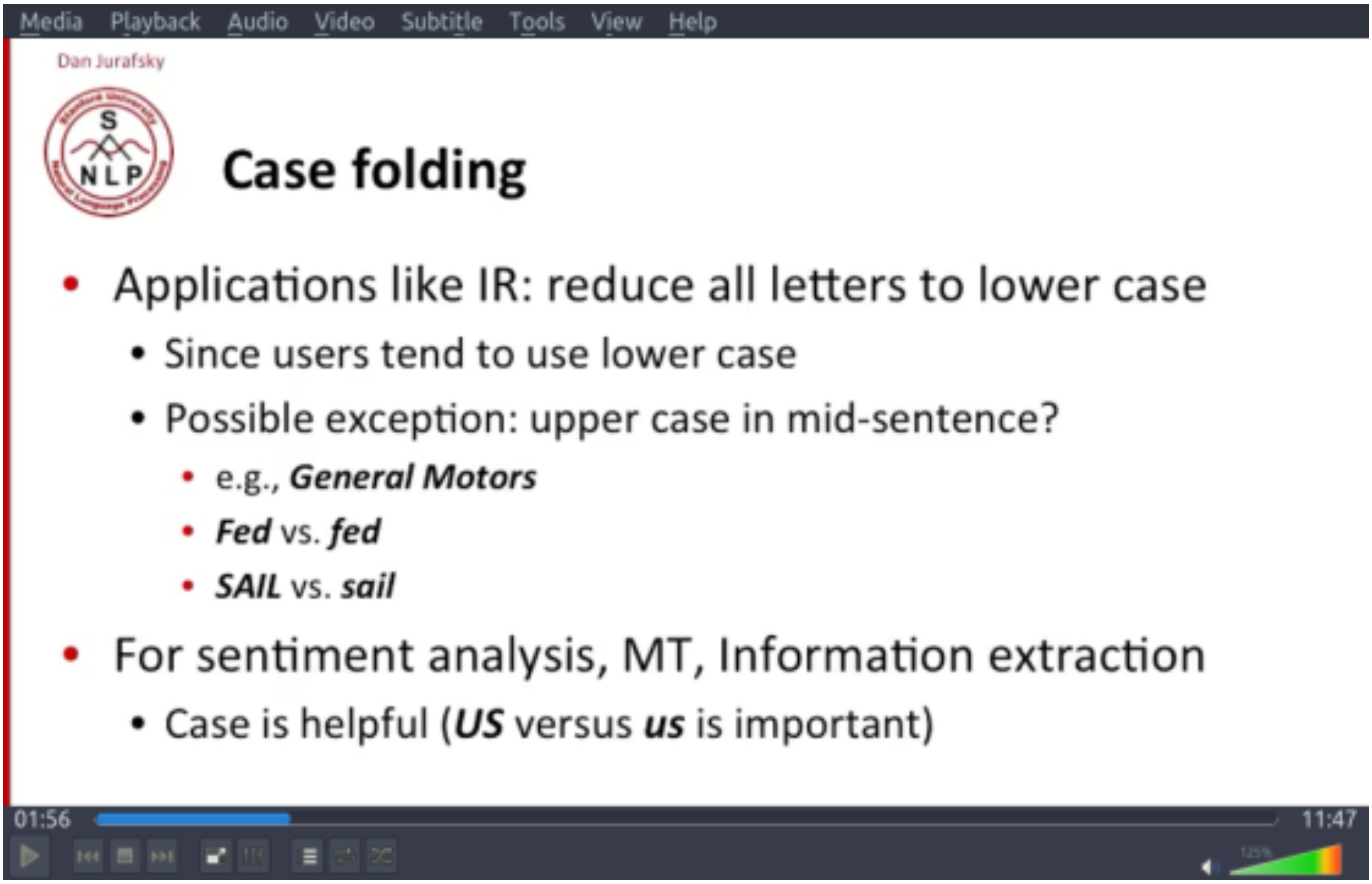
This algorithm doesn't work for English

Normalization

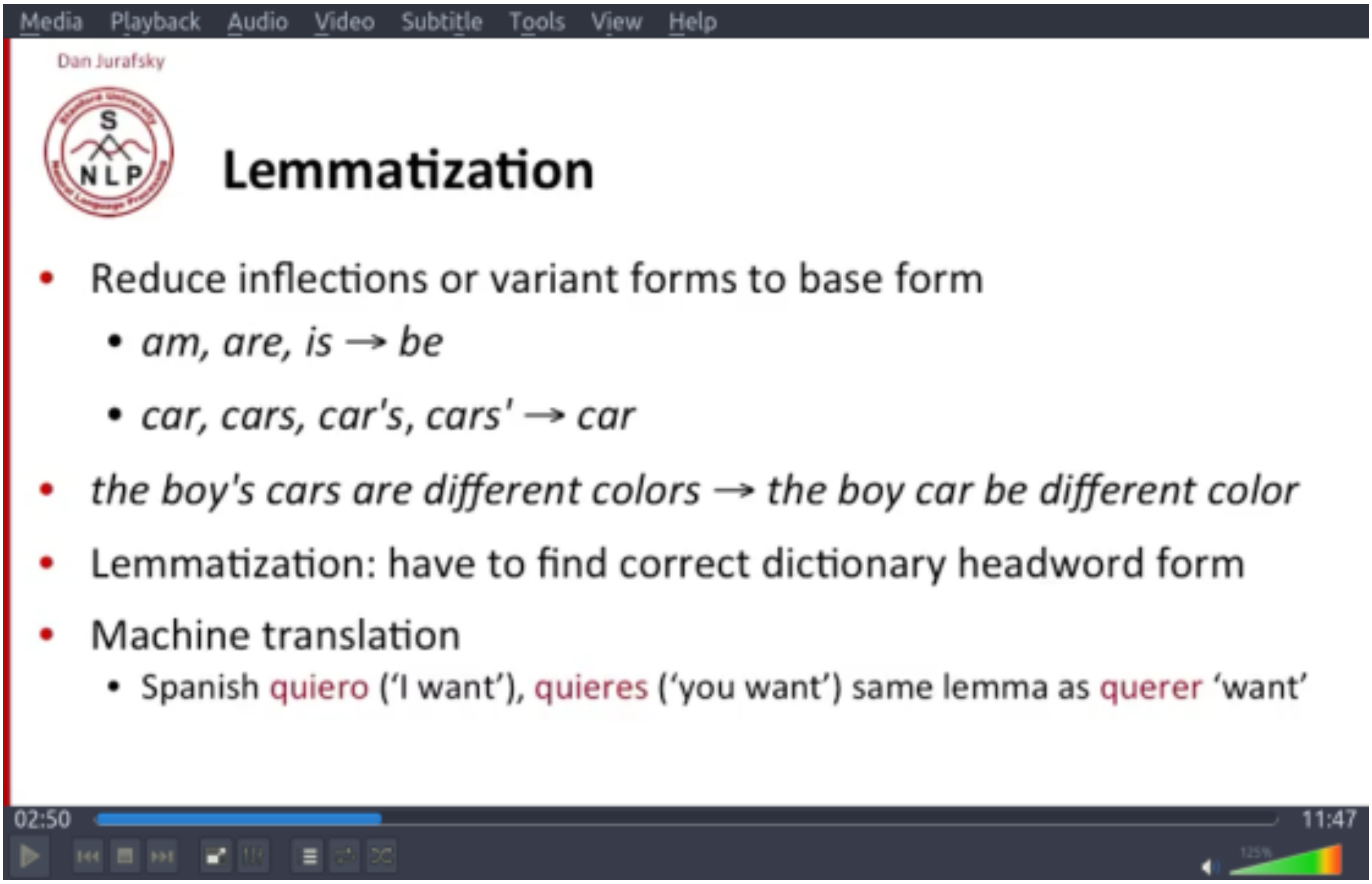


Generally it is done to make SEARCHING easy

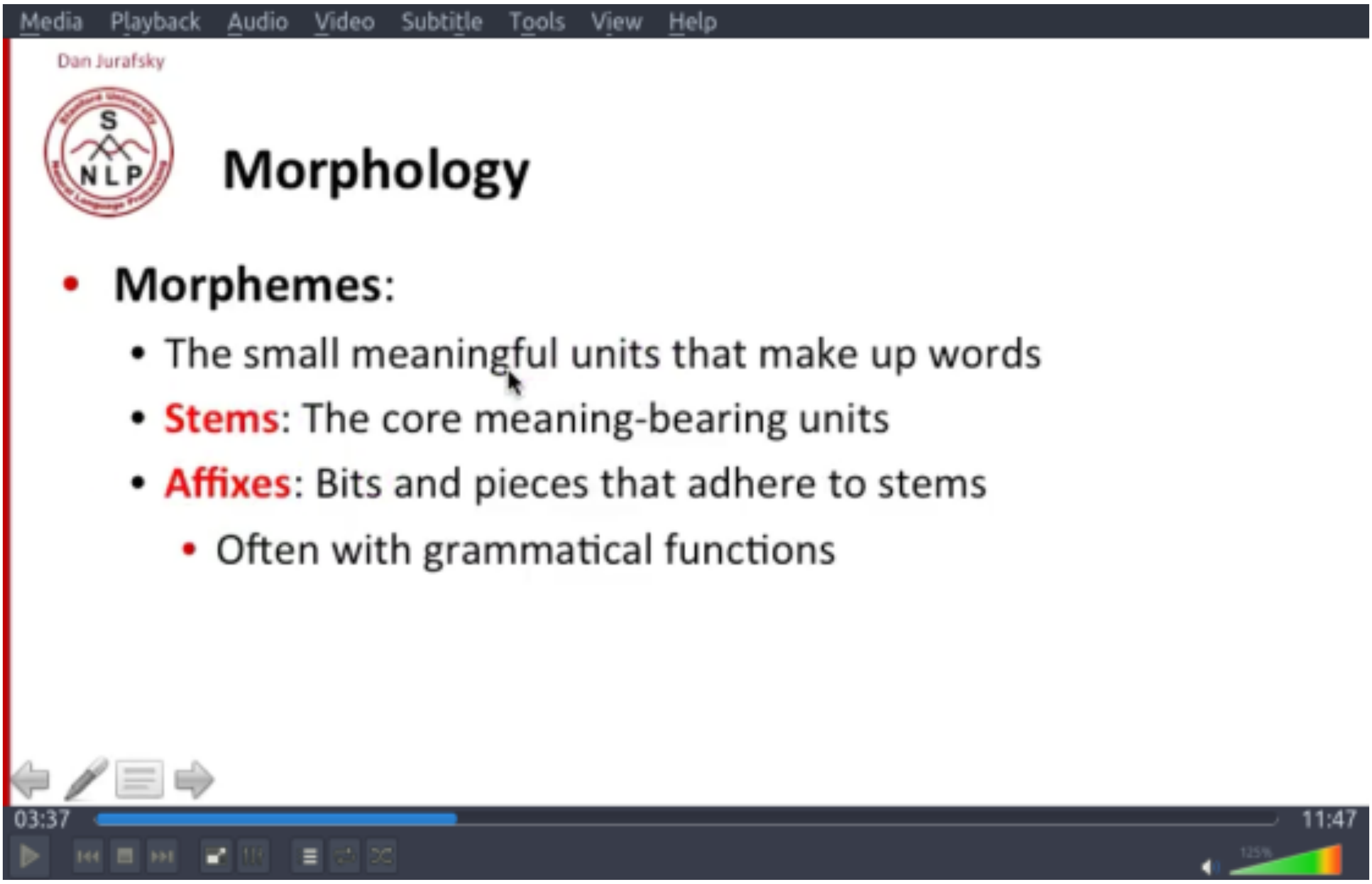
Case Folding



Lemmatization


Media Playback Audio Video Subtitle Tools View Help
Dan Jurafsky
S NLP
Lemmatization
• Reduce inflections or variant forms to base form
• am, are, is → be
• car, cars, car's, cars' → car
• the boy's cars are different colors → the boy car be different color
• Lemmatization: have to find correct dictionary headword form
• Machine translation
• Spanish quiero ('I want'), quieres ('you want') same lemma as querer 'want'
02:50
11:47

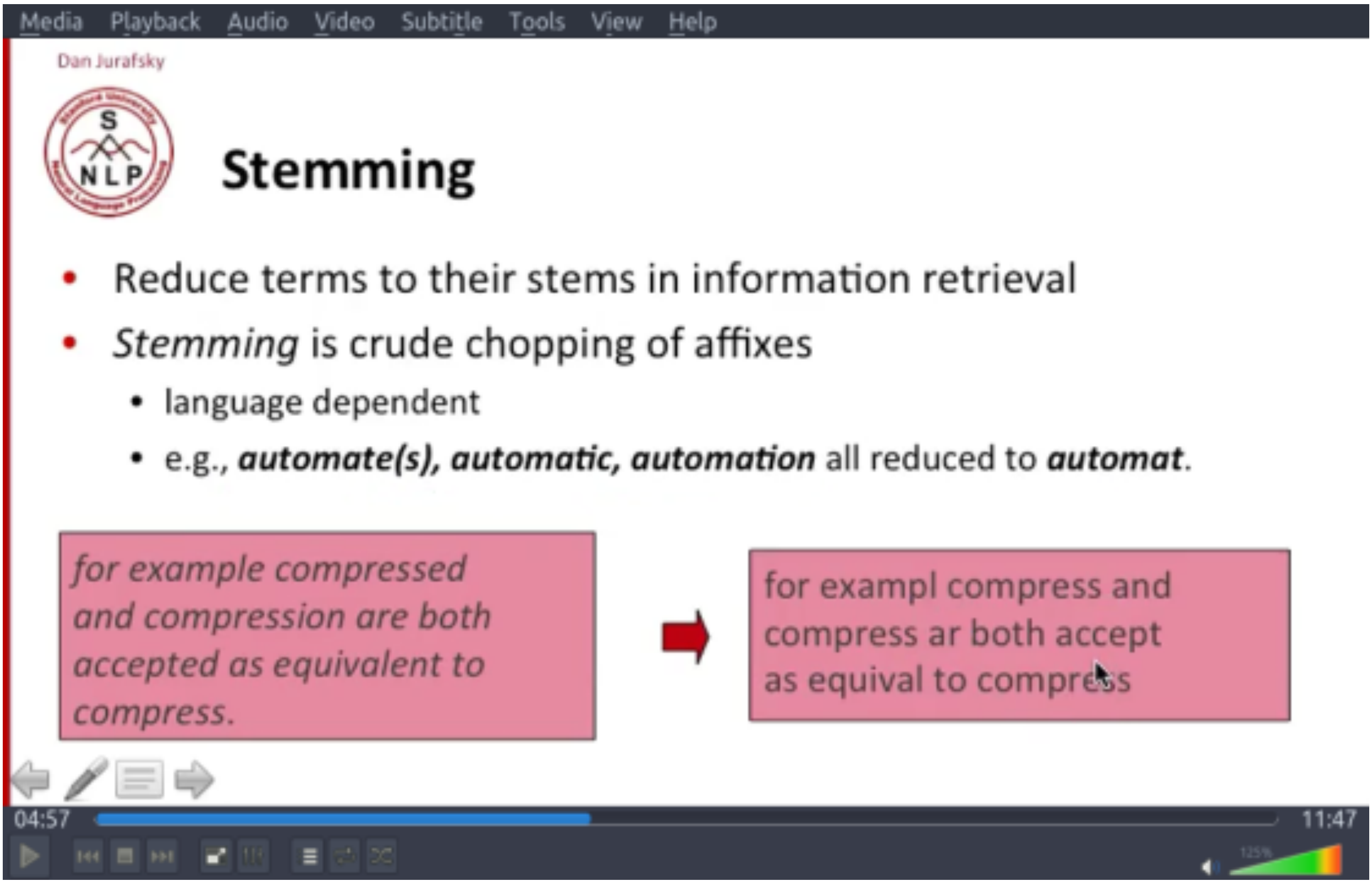
Morphology


Media Playback Audio Video Subtitle Tools View Help
Dan Jurafsky
S NLP
Morphology
• Morphemes:
• The small meaningful units that make up words
• Stems: The core meaning-bearing units
• Affixes: Bits and pieces that adhere to stems
• Often with grammatical functions
03:37
11:47

For example:

In the string “Stems” - “stem” is a stem and “s” is an affix

Stemming



Stemming is a simplified version of Lemmatization

Porter's Algorithm

Dan Jurafsky

Porter's algorithm
The most common English stemmer

Step 1a

```
sses → ss   caresses → caress
ies  → i    ponies   → poni
ss   → ss   caress   → caress
s    → ø    cats     → cat
```

Step 1b

```
(*v*)ing → ø  walking   → walk
              sing      → sing
(*v*)ed  → ø  plastered → plaster
…
```

Step 2 (for long stems)

```
ational→ ate  relational→ relate
izer→ ize     digitizer → digitize
ator→ ate     operator  → operate
…
```

Step 3 (for longer stems)

```
al   → ø  revival    → reviv
able → ø  adjustable → adjust
ate  → ø  activate   → activ
…
```

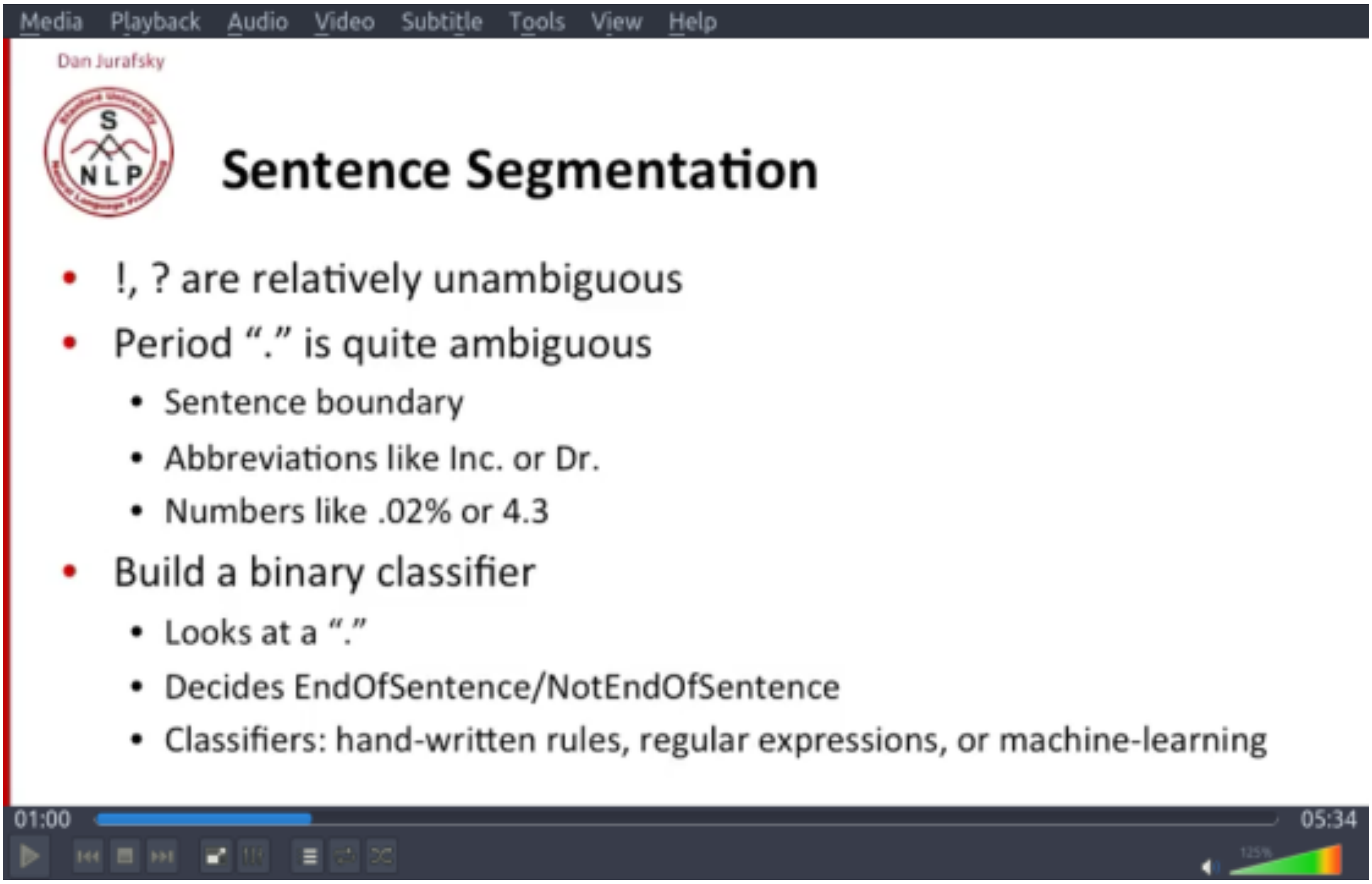

Dan Jurafsky

Viewing morphology in a corpus
Why only strip –ing if there is a vowel?

```
(*v*)ing → ø  walking → walk
              sing    → sing
```

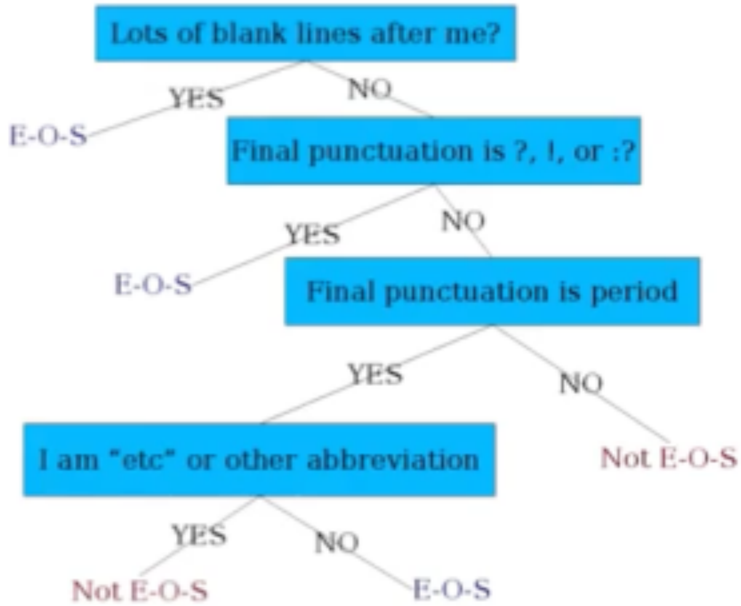
33

Sentence Segmentation



Dan Jurafsky

Determining if a word is end-of-sentence: a Decision Tree



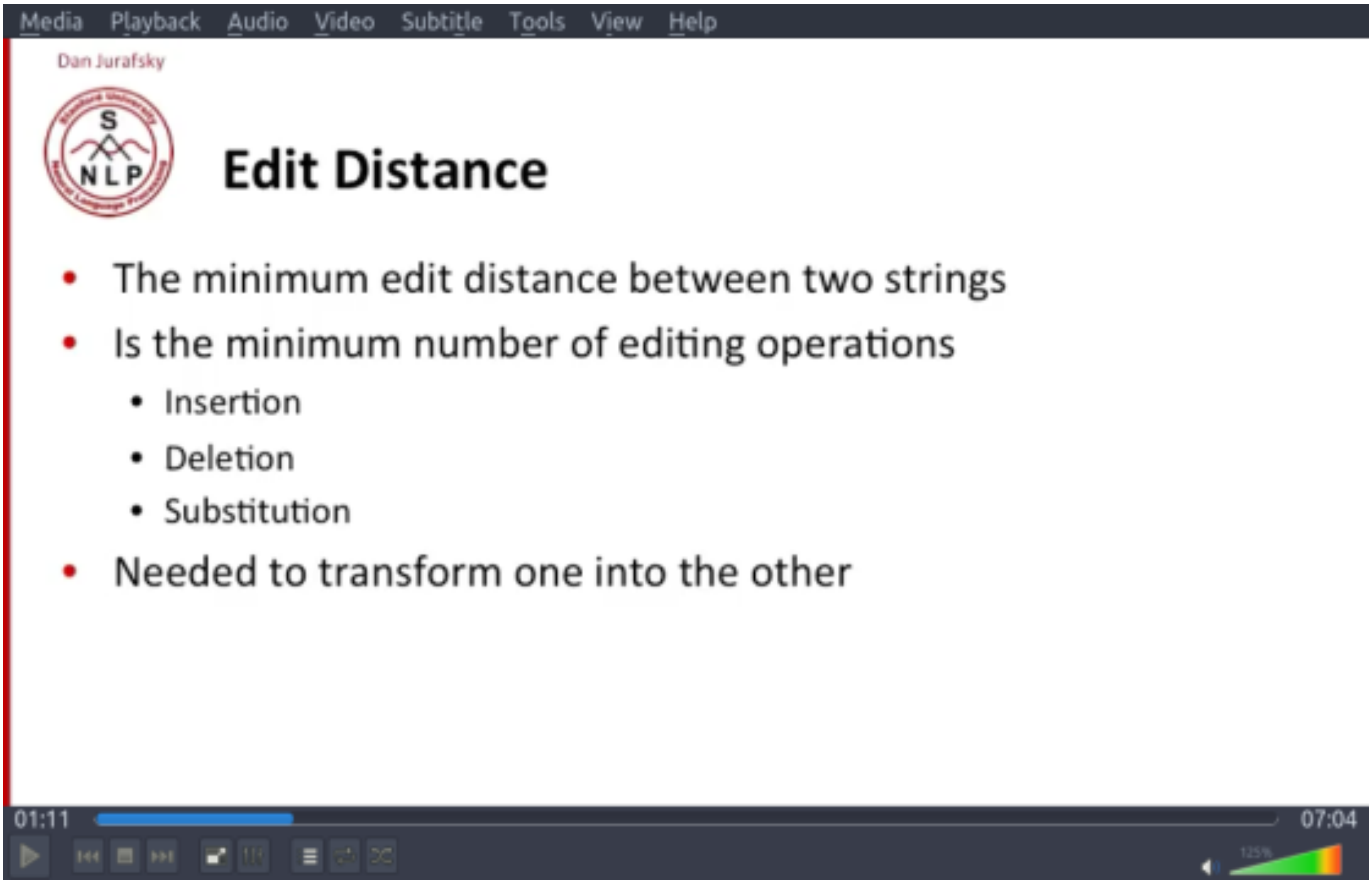
Dan Jurafsky

More sophisticated decision tree features

- Case of word with ".": Upper, Lower, Cap, Number
- Case of word after ".": Upper, Lower, Cap, Number

- Numeric features
  - Length of word with "."
  - Probability(word with "." occurs at end-of-s)
  - Probability(word after "." occurs at beginning-of-s)

Minimum Edit Distance



Media Playback Audio Video Subtitle Tools View Help

Dan Jurafsky

Defining Min Edit Distance (Levenshtein)

- Initialization

  D(i,0) = i

  D(0,j) = j

- Recurrence Relation:

  For each i = 1…M

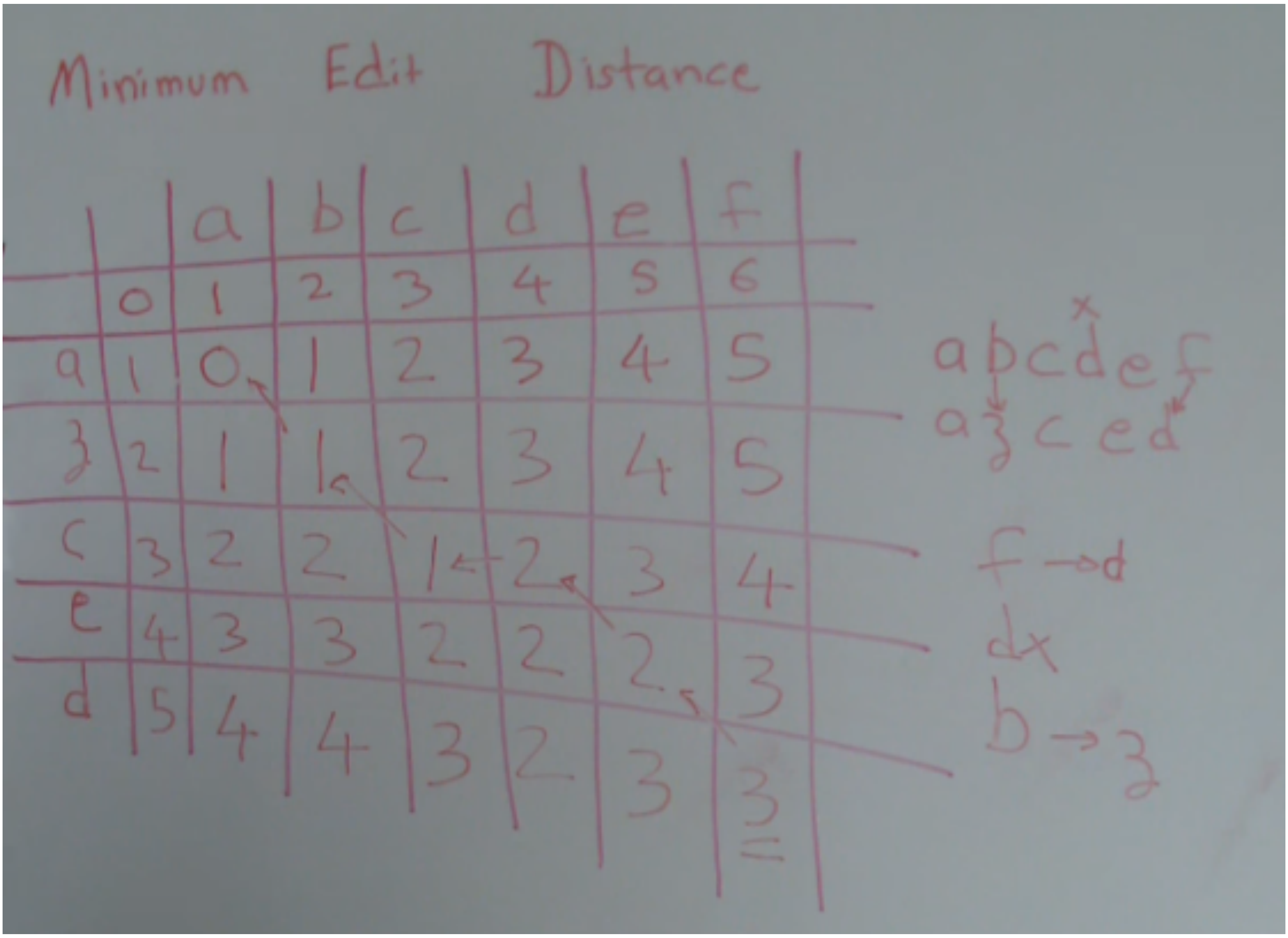
  For each j = 1…N

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } X(i) \neq Y(j) \\ 0; & \text{if } X(i) = Y(j) \end{cases} \end{cases}$$

- Termination:

  D(N,M) is distance

01:04 05:54



<u>Note:</u>

• If the comparing characters are different then the value will be minimum of (left value or top value or left diagonal value) plus 1
• if the comparing characters are same then the value will be equal to the left diagonal value

Language Modelling

WHY?

Media Playback Audio Video Subtitle Tools View Help

Dan Jurafsky

Probabilistic Language Models

- Today's goal: assign a probability to a sentence
  - Machine Translation:
    - P(**high** winds tonite) > P(**large** winds tonite)
  - Spell Correction
    - The office is about fifteen **minuets** from my house
      - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
  - Speech Recognition
    - P(I saw a van) >> P(eyes awe of an)
  - + Summarization, question-answering, etc., etc.!!

Why?

01:02 08:41

Media Playback Audio Video Subtitle Tools View Help

Dan Jurafsky

Probabilistic Language Modeling

- Goal: compute the probability of a sentence or sequence of words:

  $P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$

- Related task: probability of an upcoming word:

  $P(w_5|w_1,w_2,w_3,w_4)$ $P(w_1 w_2 v_3 w_4 w_5 -)$

- A model that computes either of these:

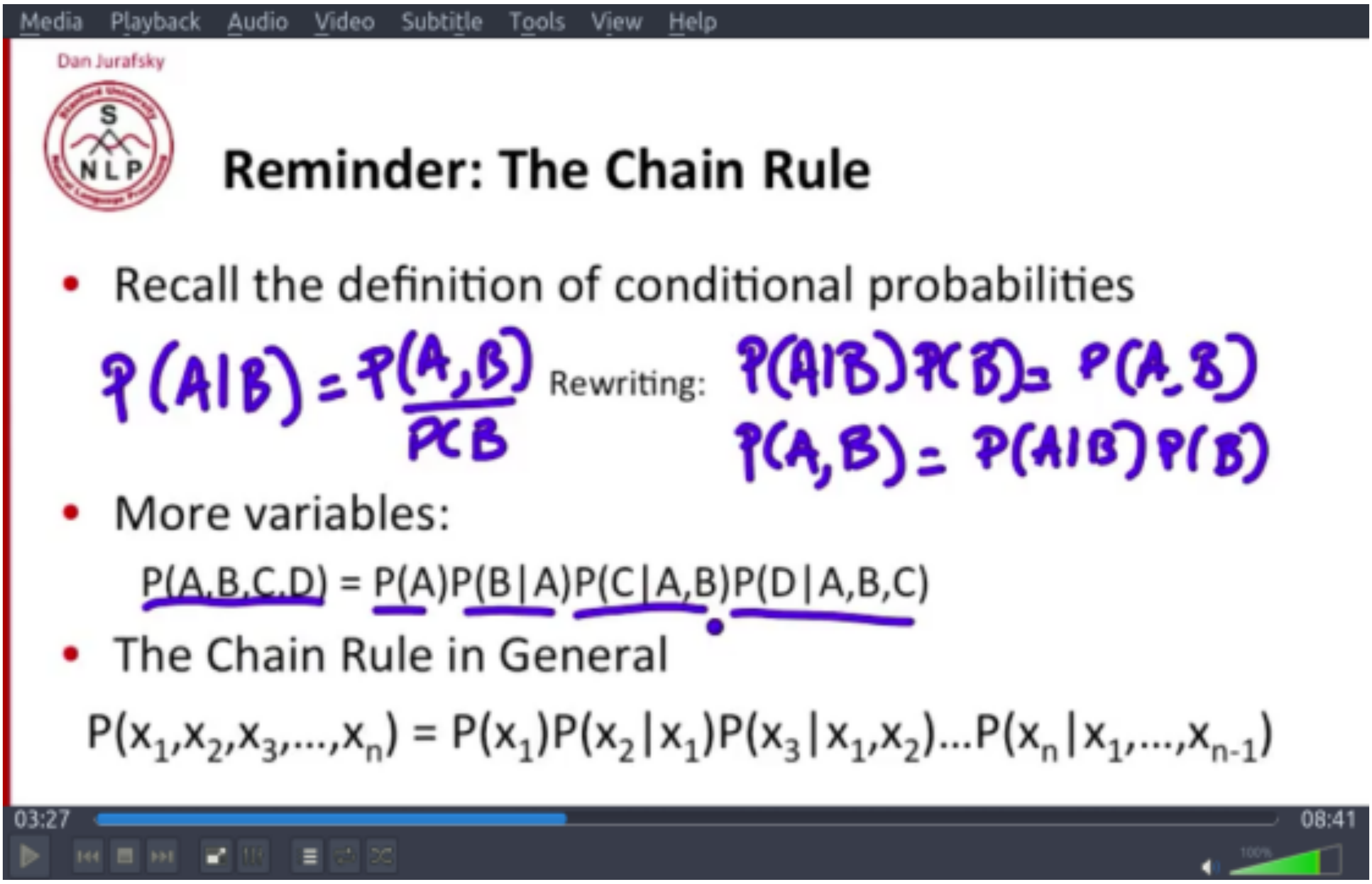
  $P(W)$ or $P(w_n|w_1,w_2 \ldots w_{n-1})$ is called a **language model**.

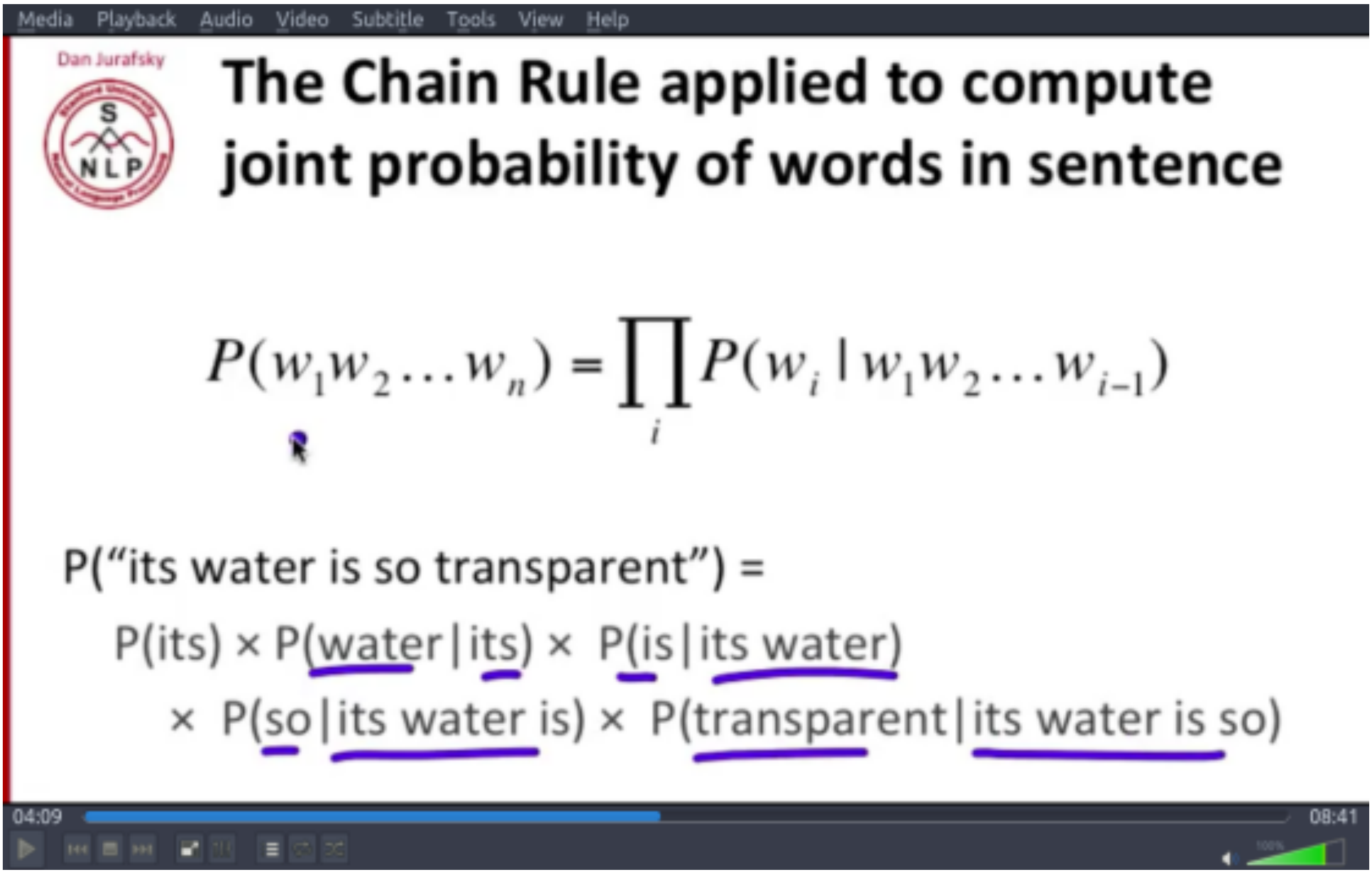
01:50 08:41

Compute the Joint Probability of a sentence or a sequence of words

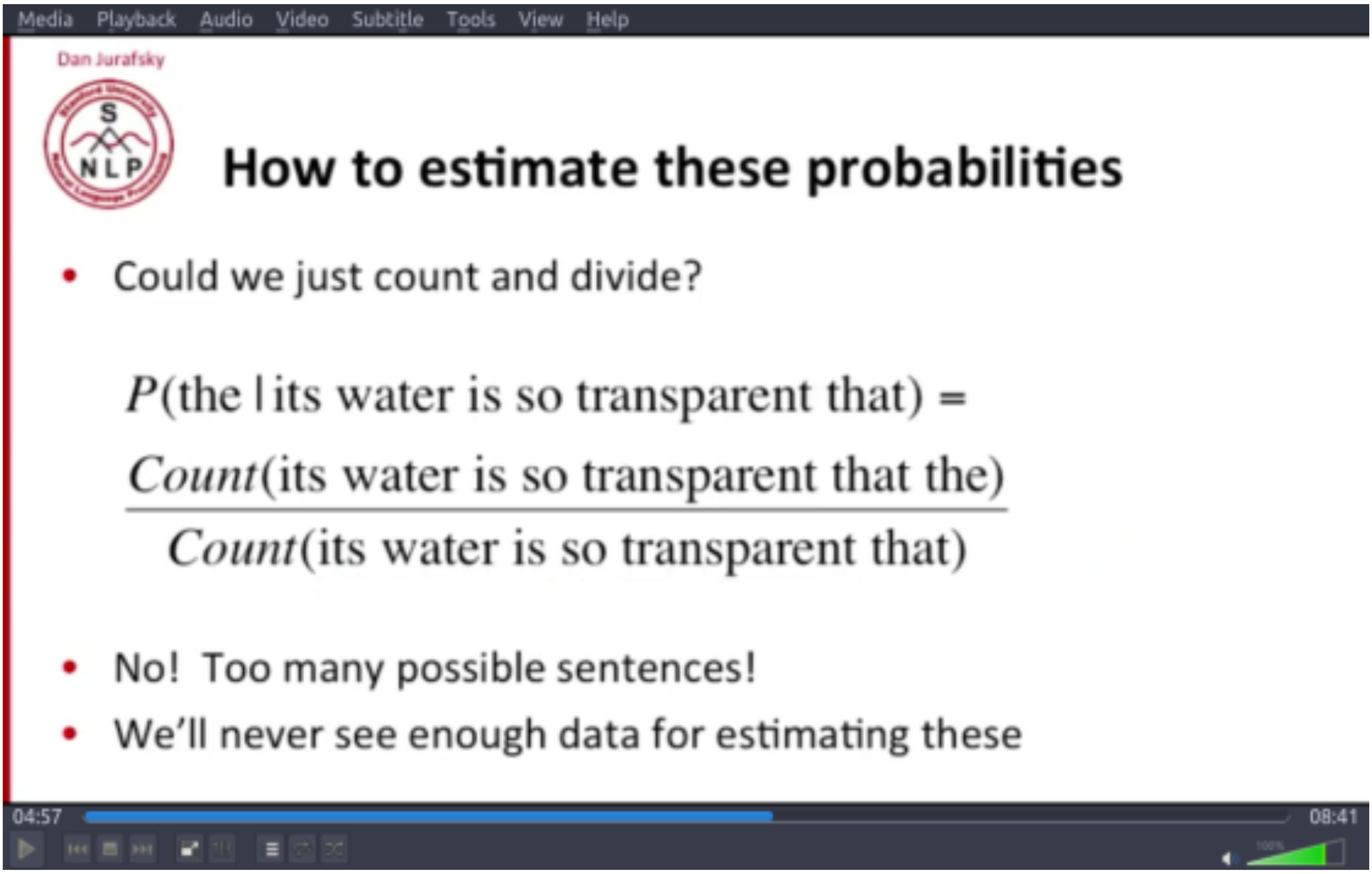
P (its, water, is, so, transparent, that)

This can be calculated using **Chain Rule of Probability**



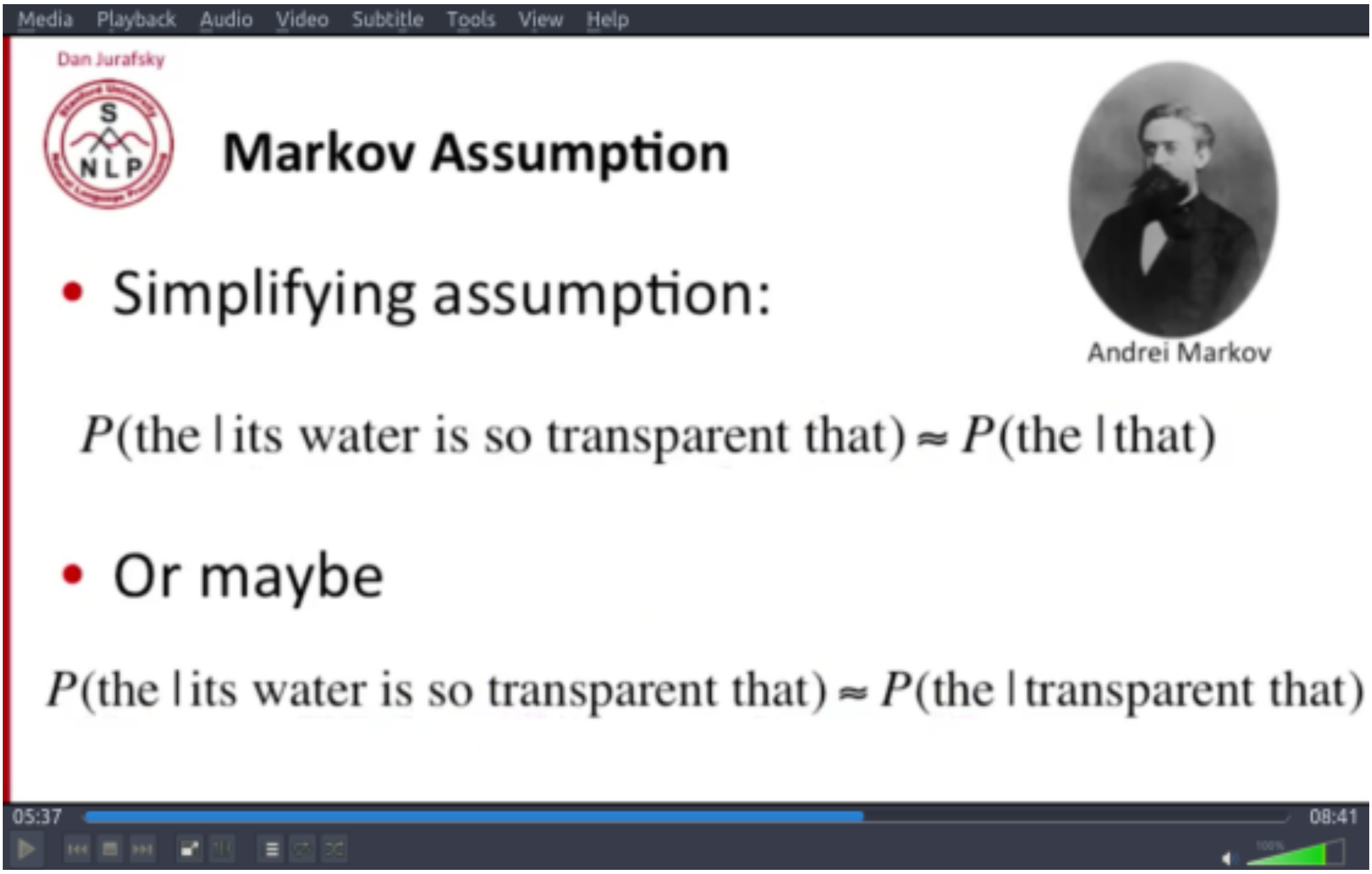


How to calculate this probability?



Markov Assumption is used to calculate this probability

Markov Assumption



Media Playback Audio Video Subtitle Tools View Help

Dan Jurafsky

Markov Assumption


Andrei Markov

- Simplifying assumption:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$$

- Or maybe

$$P(\text{the} \mid \text{its water is so } \boxed{\text{transparent that}}) \approx P(\text{the} \mid \text{transparent that})$$

05:34 08:41

Media Playback Audio Video Subtitle Tools View Help

Dan Jurafsky

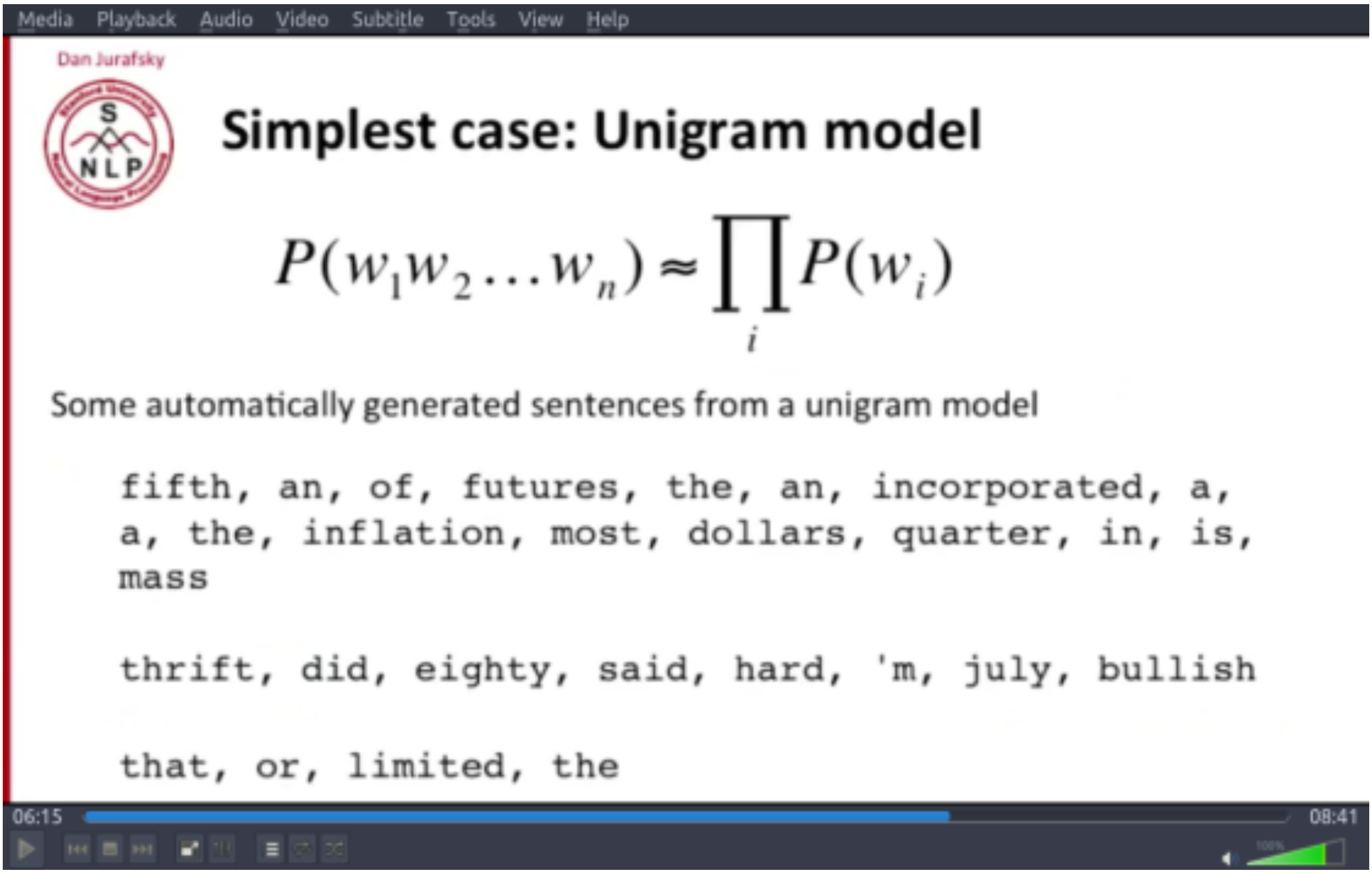
Markov Assumption

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i \mid w_{i-k} \ldots w_{i-1})$$

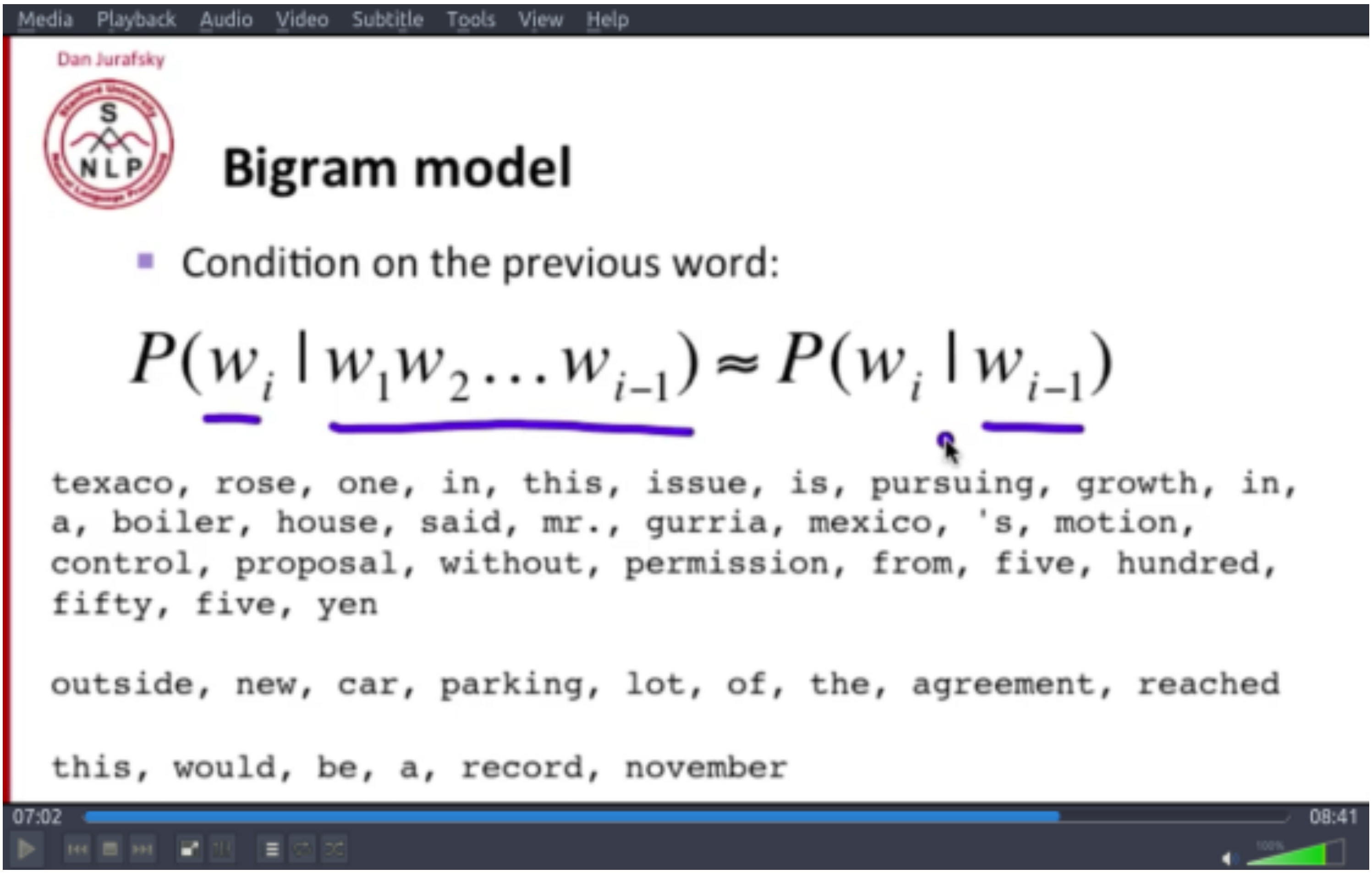
- In other words, we approximate each component in the product

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-k} \ldots w_{i-1})$$

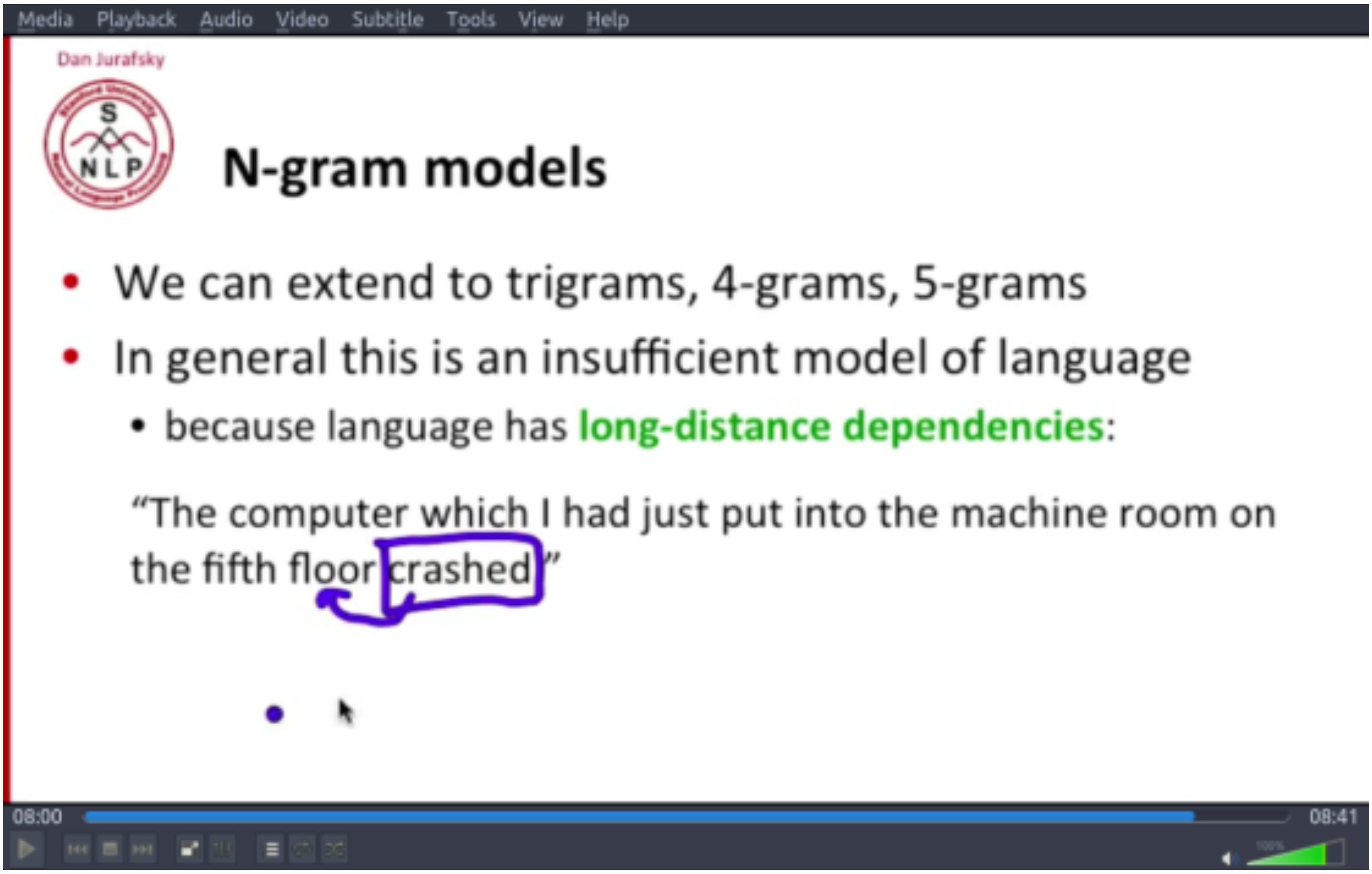
05:39 08:41



In Unigram Model, the words are independent of each other. So the sentence apparently doesn't make sense.



In Bigram Model the word is dependent on its previous word.



Like in the above example, N-gram model can fail with sentences having long distance dependencies.
Because the word "crashed" is not related to floor. Here the actual subject is computer. Hence the N-gram model will fail here.
But more or less it works OK for most of the sentences.

Dan Jurafsky

An example

Estimating Bigram Probabilities

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

$P(\texttt{I} \mid \texttt{<s>}) = \frac{2}{3} = .67$ $\quad P(\texttt{Sam} \mid \texttt{<s>}) = \frac{1}{3} = .33$ $\quad P(\texttt{am} \mid \texttt{I}) = \frac{2}{3} = .67$

$P(\texttt{</s>} \mid \texttt{Sam}) = \frac{1}{2} = 0.5$ $\quad P(\texttt{Sam} \mid \texttt{am}) = \frac{1}{2} = .5$ $\quad P(\texttt{do} \mid \texttt{I}) = \frac{1}{3} = .33$

Count (am, Sam) / Count (am) = 1 / 2

Dan Jurafsky

Practical Issues

- We do everything in log space
  - Avoid underflow Multiplying Numbers like 0.002 * 0.00001
  - (also adding is faster than multiplying)

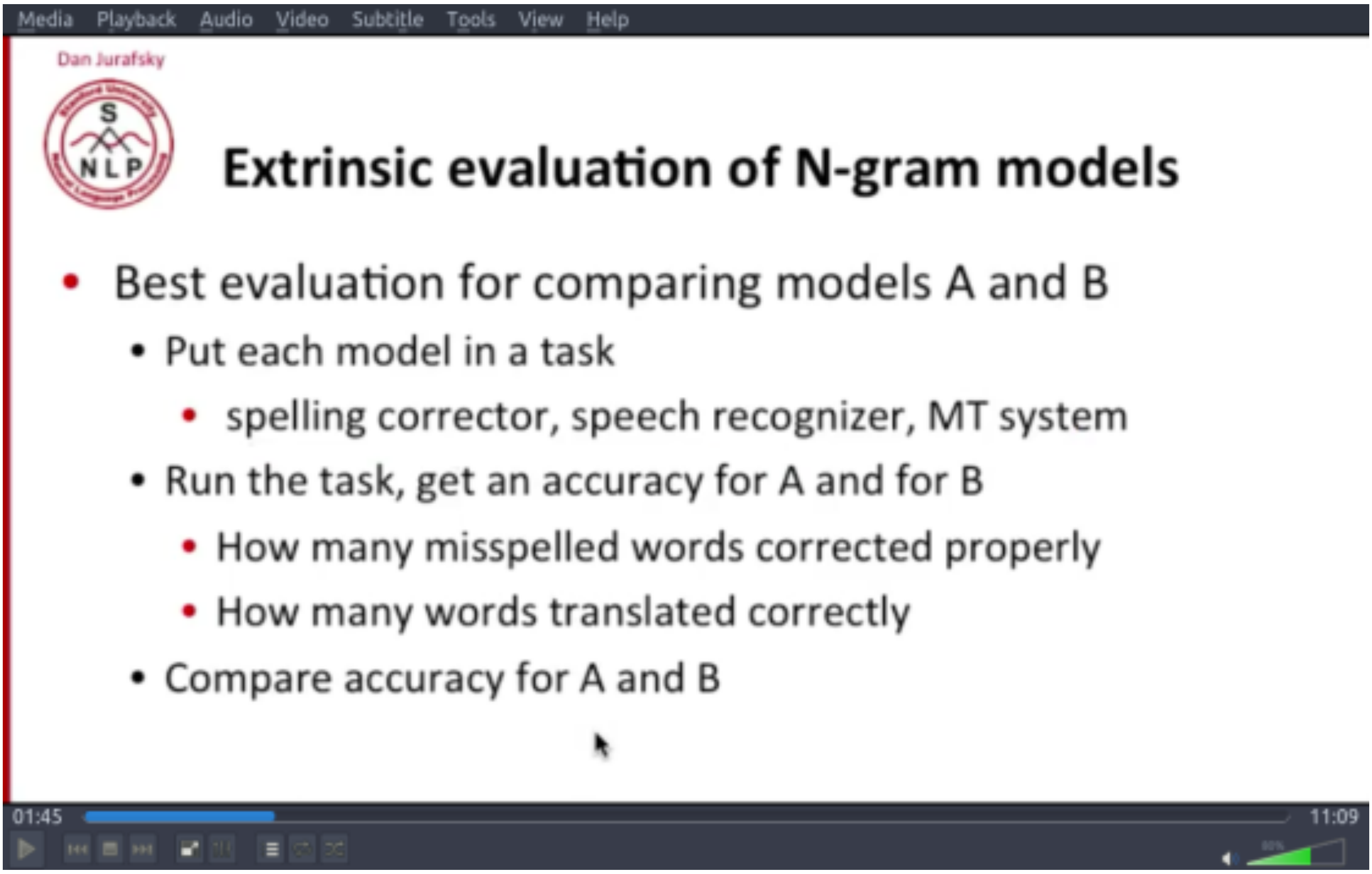
$$p_1 \times p_2 \times p_3 \times p_4 = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

Evaluation of Model

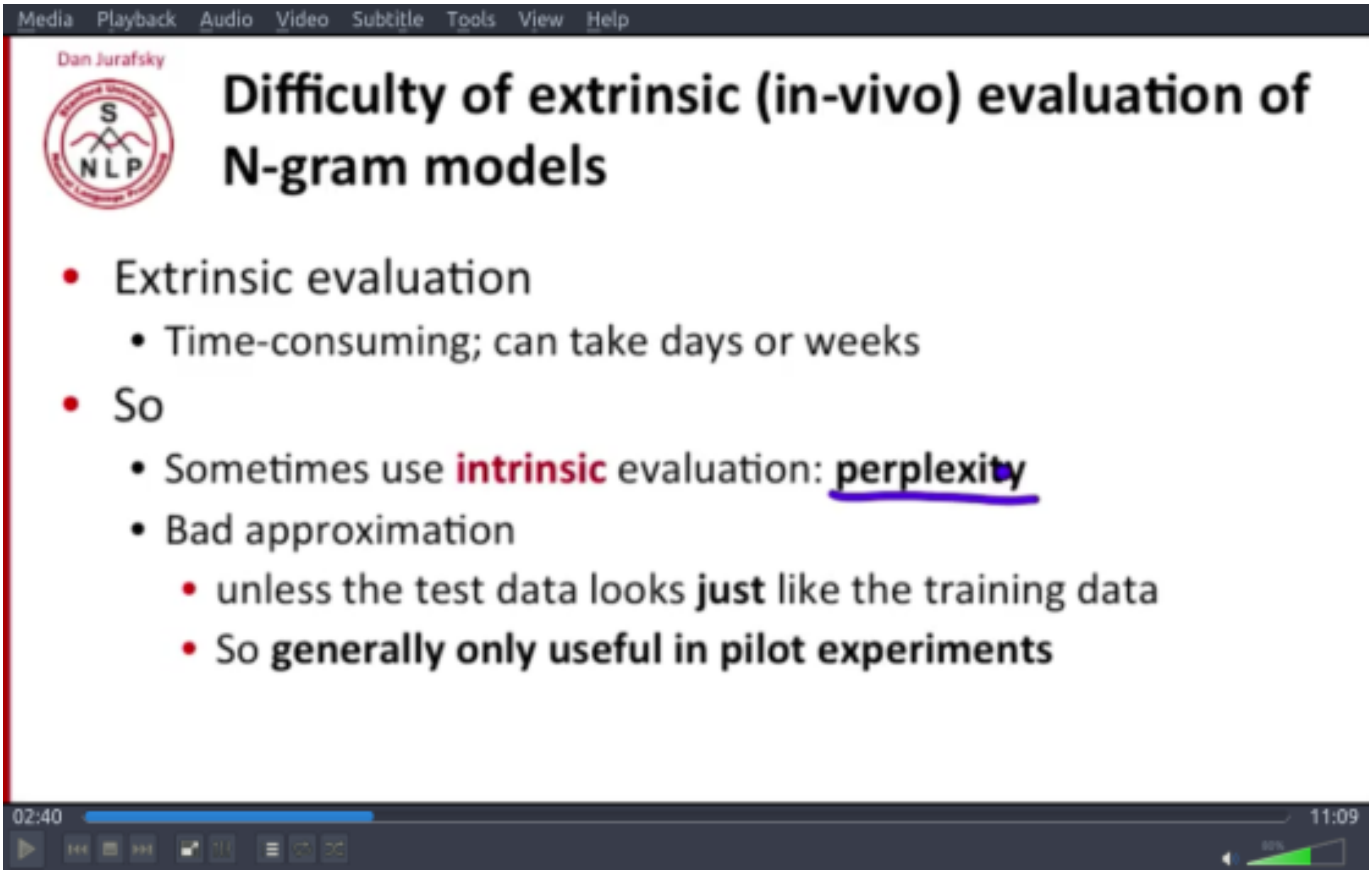
Model can be evaluated in two ways

1. Extrinsic
2. Intrinsic

Extrinsic evaluation is done by using external data and comparing the accuracy. Like using the unseen test data and checking the accuracy on it.



Intrinsic evaluation is the evaluation of the model itself rather than any external application



A type of intrinsic evaluation is Perplexity
Perplexity is the probability of the test set, normalized by the number of words.

Media Playback Audio Video Subtitle Tools View Help

Dan Jurafsky

Perplexity

The best language model is one that best predicts an unseen test set

- Gives the highest P(sentence)

Perplexity is the probability of the test set, normalized by the number of words:

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Chain rule:

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \ldots w_{i-1})}}$$

For bigrams:

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_{i-1})}}$$

Minimizing perplexity is the same as maximizing probability

06:56 11:09

Media Playback Audio Video Subtitle Tools View Help

Dan Jurafsky

The Shannon Game intuition for perplexity

- From Josh Goodman
- How hard is the task of recognizing digits '0,1,2,3,4,5,6,7,8,9'
  - Perplexity 10
- How hard is recognizing (30,000) names at Microsoft.
  - Perplexity = 30,000
- If a system has to recognize
  - Operator (1 in 4)
  - Sales (1 in 4)
  - Technical Support (1 in 4)
  - 30,000 names (1 in 120,000 each)
  - Perplexity is 54
- Perplexity is weighted equivalent branching factor

average branching factor

The perplexity is actually 53, not 54

08:45 11:09

Dan Jurafsky

Perplexity as branching factor

- Let's suppose a sentence consisting of random digits
- What is the perplexity of this sentence according to a model that assign P=1/10 to each digit?

$$
\begin{aligned}
\mathrm{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\
&= \left(\frac{1}{10}^{N}\right)^{-\frac{1}{N}} \\
&= \frac{1}{10}^{-1} \\
&= 10
\end{aligned}
$$

$$P(w_1 w_2 \cdot w_N)^{-\frac{1}{N}}$$

$$P\left(\frac{1}{10} \cdot \frac{1}{10} \cdot \frac{1}{10} \cdot \frac{1}{10}\right.$$

$$\left(\frac{1}{10}^{N}\right)^{-\frac{1}{N}}$$

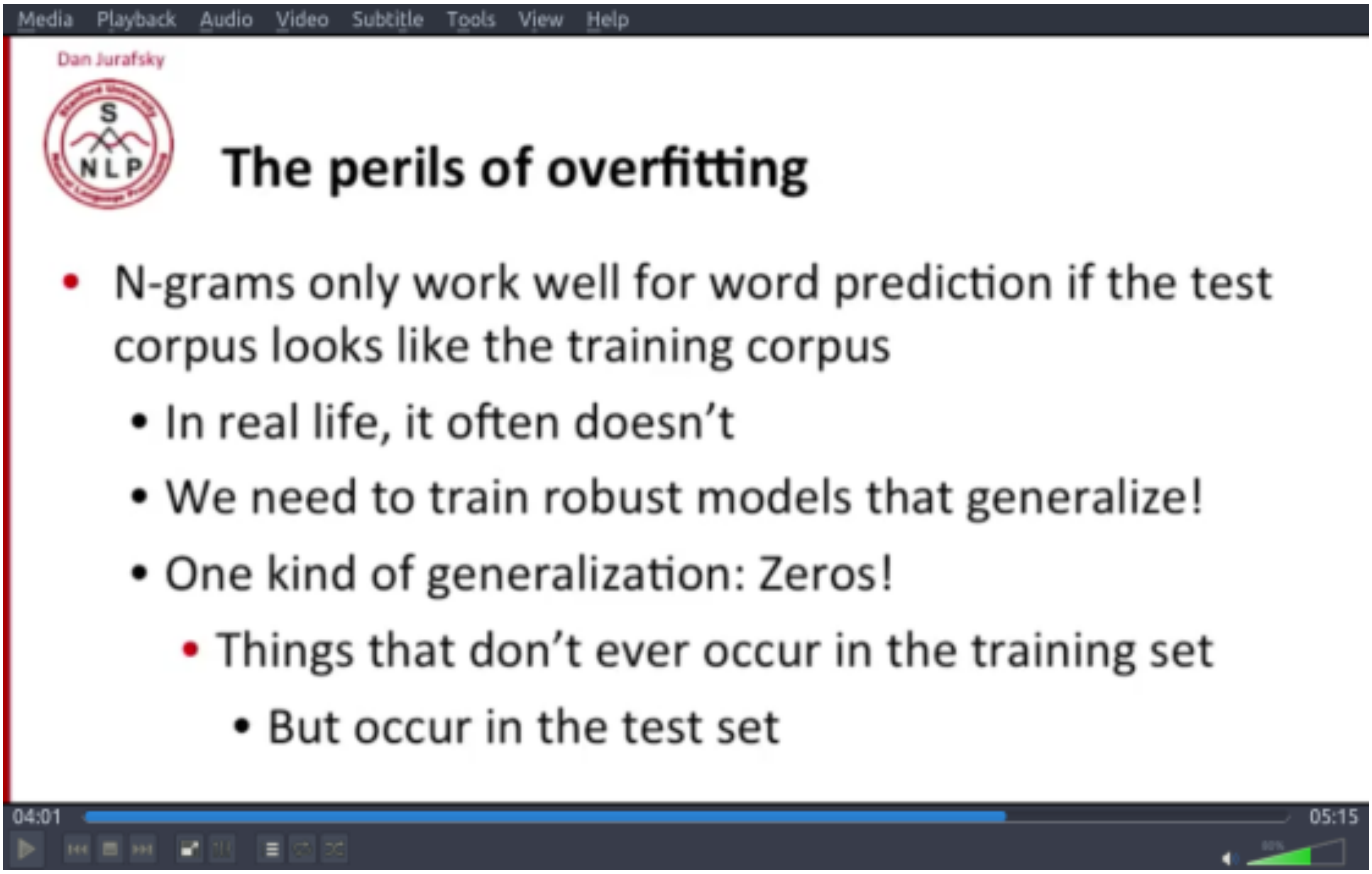
Dan Jurafsky

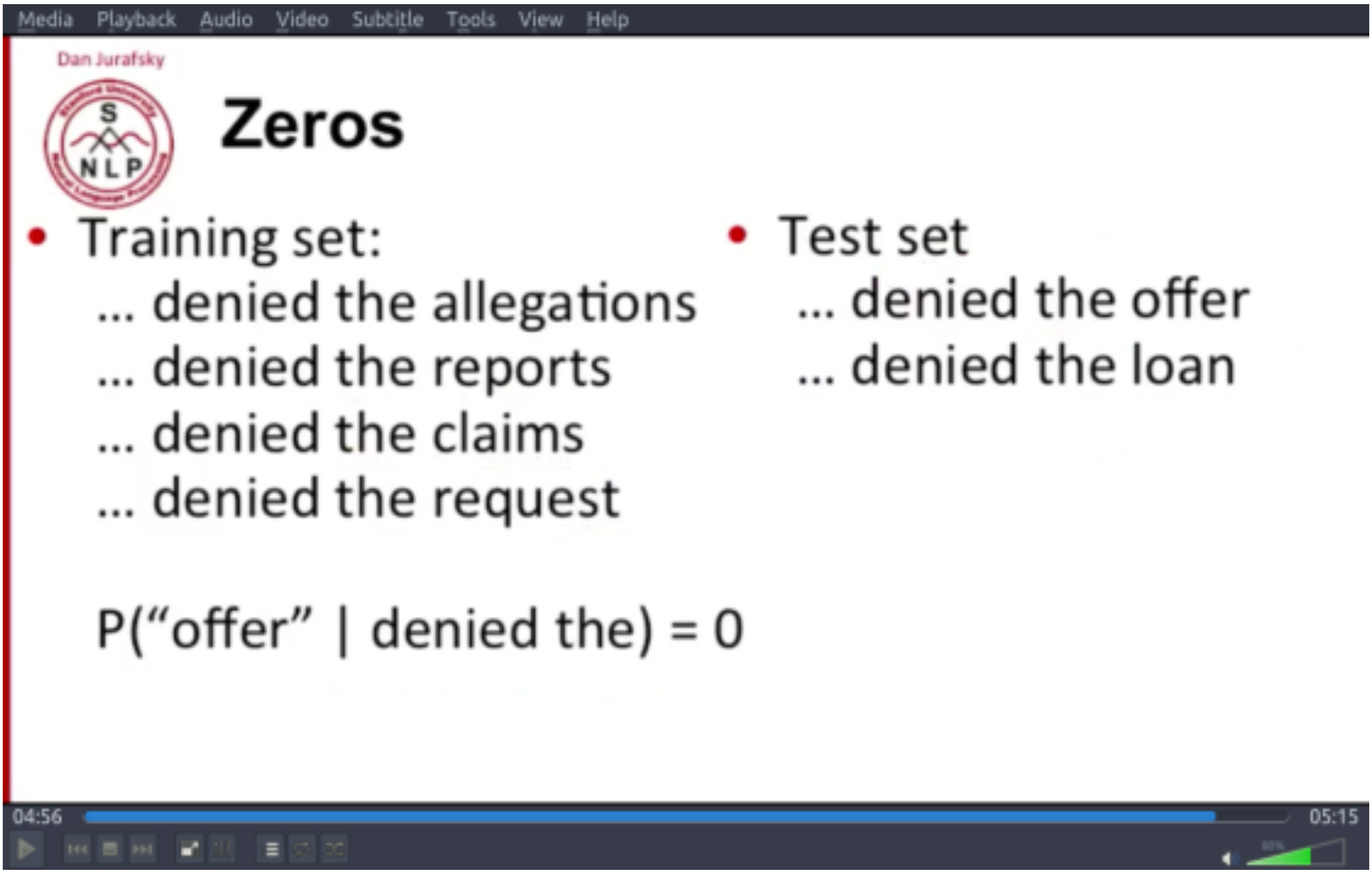
Lower perplexity = better model

- Training 38 million words, test 1.5 million words, WSJ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

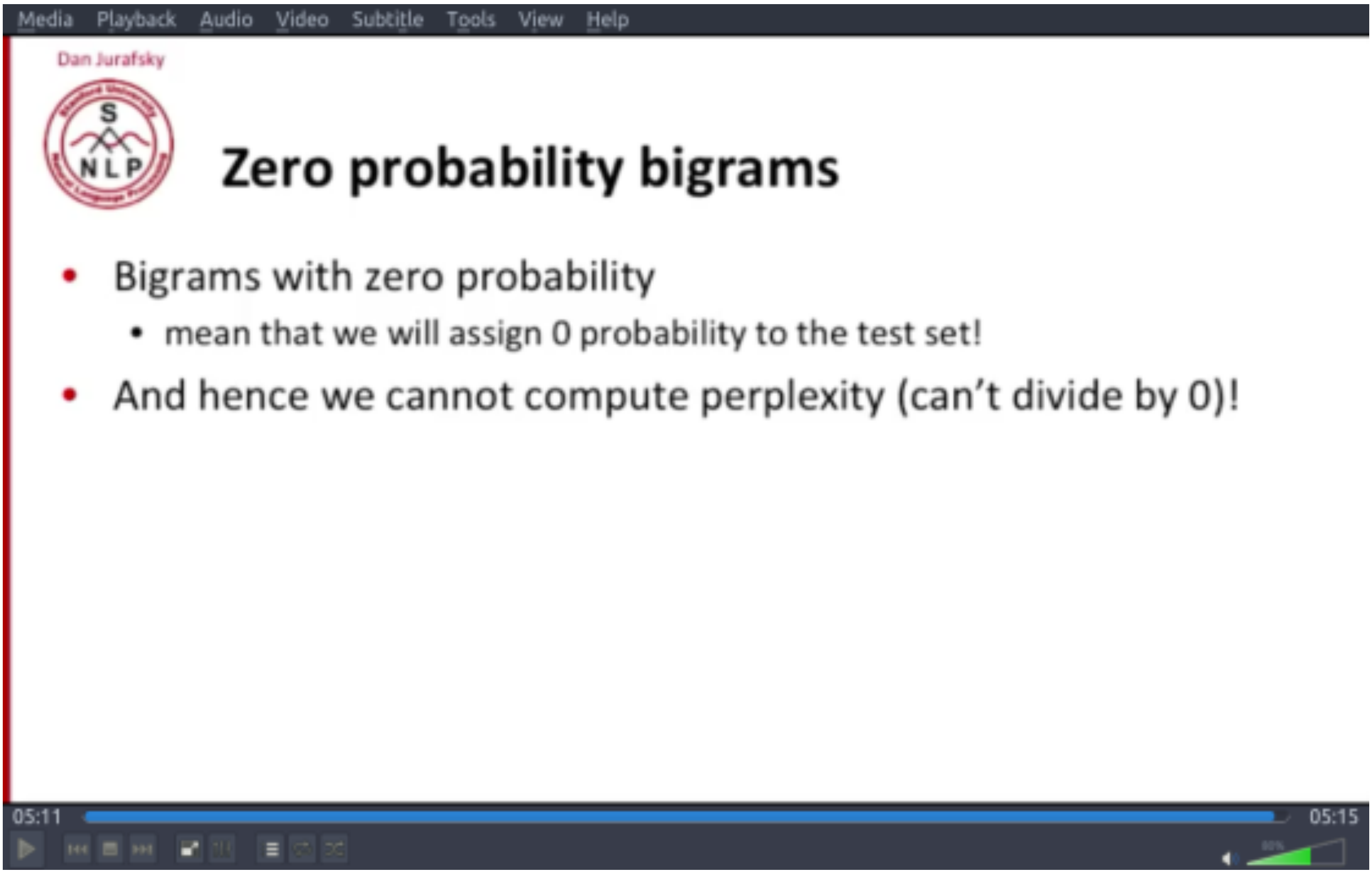
Generalization





Zero Probability Problem

When the sentences in the test set have never appeared in the training set, the probability assigned to it is zero.
This produces Zero Probability Problem.



Smoothing (Add One Estimate)

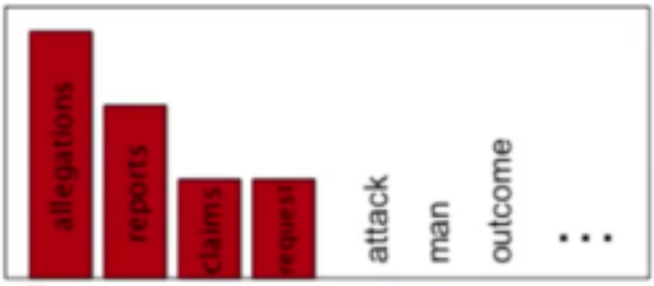
To avoid Zero Probability Problem, Smoothing or Add One Estimate technique is used.

Dan Jurafsky

The intuition of smoothing (from Dan Klein)

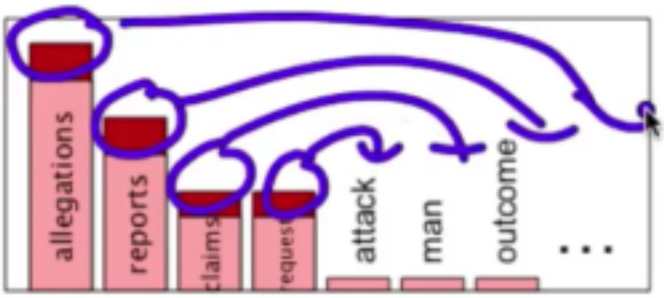
- When we have sparse statistics:

  P(w | denied the)
  3 allegations
  2 reports
  1 claims
  1 request
  7 total



- Steal probability mass to generalize better

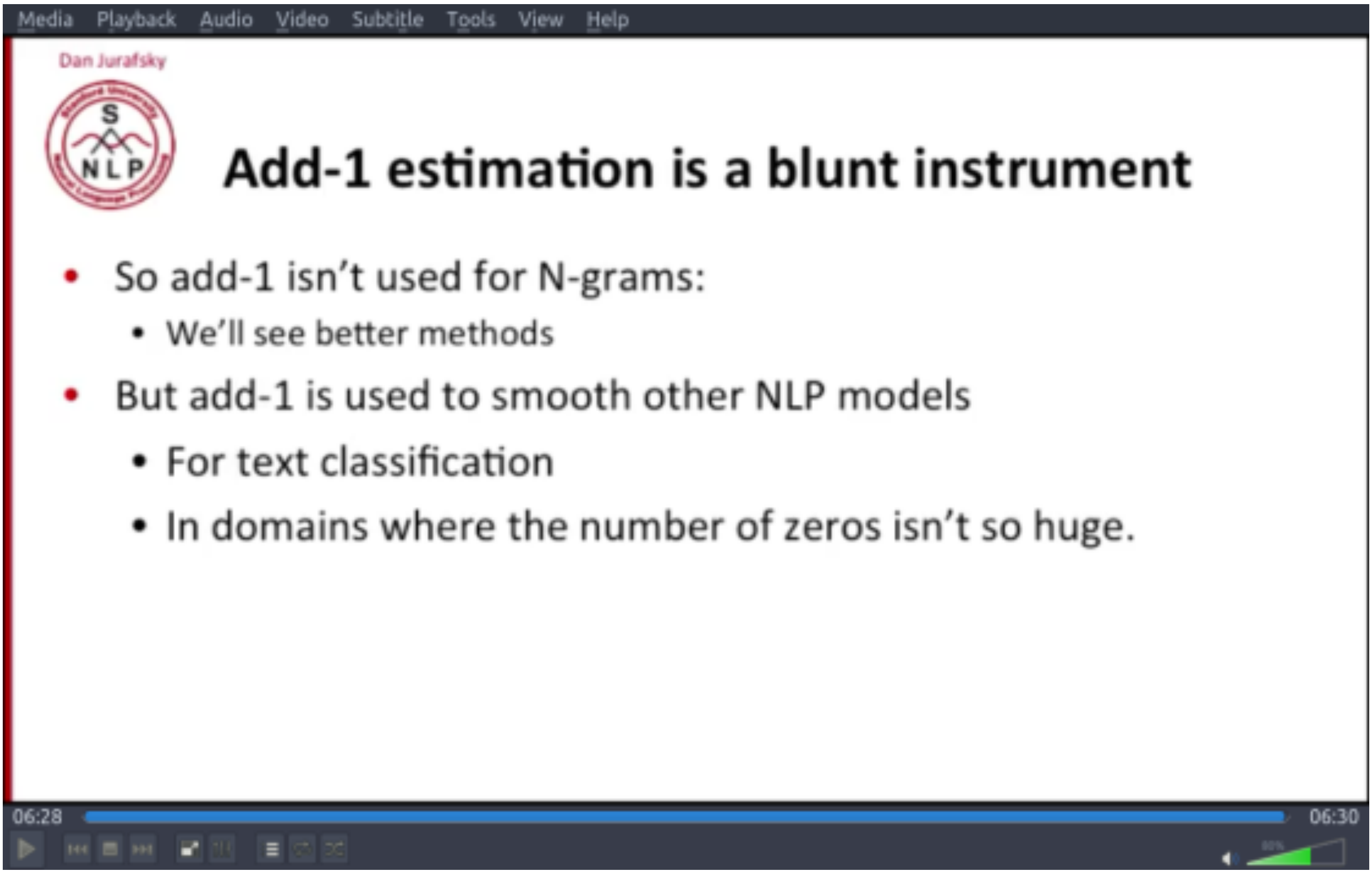
  P(w | denied the)
  2.5 allegations
  1.5 reports
  0.5 claims
  0.5 request
  2 other
  7 total



Dan Jurafsky

Add-one estimation

- Also called Laplace smoothing
- Pretend we saw each word one more time than we did
- Just add one to all the counts!
- MLE estimate: $$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$
- Add-1 estimate: $$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)+1}{c(w_{i-1})+V}$$


Media Playback Audio Video Subtitle Tools View Help
Dan Jurafsky
S
NLP
Add-1 estimation is a blunt instrument
• So add-1 isn't used for N-grams:
• We'll see better methods
• But add-1 is used to smooth other NLP models
• For text classification
• In domains where the number of zeros isn't so huge.
06:28
06:30