CISC 372
Text Analytic I

Today

- Text Analysis
  - Classification
  - Preprocessing
  - Representations
    - BOW
    - N-gram
    - character n-gram
    - Part-Of-Speech Tagging
    - Dependency Tree
  - Vanilla RNN

The Web contains a LOT of text.

- What can we do with it?
  - read it, with the help of search engines
  - communicate (email, IM, social media, …)
  - learn about how humans inform and communicate, by reversing engineering language use to infer properties of the writer

- Text Analytics
  - Sentiment Analysis
  - Authorship Analysis
  - Socio-economic characteristics inference
  - As a 'reverse-engineering' problem

Sentiment Analysis

- A sentence/paragraph -> **positive** or **negative**

This film was just brilliant, casting location, scenery story direction everyone's really suited the part they played.

Preprocessing

- Case normalization
  - All lower case (or all higher case)
  - Case may carry critical information depending on the problem
  - E.g. Cases carry writing style

- Stop words & Punctuation removal
  - i, me, my, myself, we, our, ours, ourselves, you, your, yours, yourself, yourselves, he, him, his, himself, she, her, hers, herself, it, its, itself, they, them, their, theirs, …
  - Domain-driven
    - carries writing style
    - carries semantic information
    - "few" vs "a few"

Preprocessing

- Stemming
  - Stemmers remove morphological affixes from words, leaving only the word stem. (nltk)
  - print(stemmer.stem("running"))
    - run
  - print(stemmer.stem("generously"))
    - generous

Representation

- How can we represent sequential text data as a numeric vector (from unstructured data to structured data)

This film was just brilliant, casting location, scenery story direction everyone's really suited the part they played.

Bag-of-Words model

- Represent each unique word as a **feature**, and the value can be
  - Term frequency (TF)
  - Term_frequency / document_frequency (TF-IDF)

- "This film was just brilliant"

  -> {"this": 1, "film":1, "was": 1, "just", 1, "brilliant":1}

  -> [1,1,1,1]

Document-Word Matrix

- doc1: "this is an apple "
- doc2: "this is an orange"

	an	this	is	apple	orange
doc1	1	1	1	1	0
doc2	1	1	1	0	1

Document-Word Matrix

- doc1: "The dog bit the man"
- doc2: "The man bit the dog"

	the	dog	bit	man
doc1	2	1	1	1
doc2	2	1	1	1

Bag-of-***n***-gram model

- Represent unique *word sequence of length **n** as feature*, value can be:
  - Term frequency (TF)
  - Term_frequency / document_frequency (TF-IDF)

- "This film was just brilliant"

  -> {"this_film": 1, "film_was":1, "was_just": 1, "just_brilliant": 1}

  -> [1,1,1]

Document-Word Matrix (n-gram model)

- doc1: "The dog bit the man"
- doc2: "The man bit the dog"

	the-dog	dog-bit	bit-the	the-man	man-bit
doc1	1	1	1	1	0
doc2	2	0	1	1	1

Bag-of-***n***-perm model

- Represent unique *unordered word sequence of length **n** as feature*, value can be:
  - Term frequency (TF)
  - Term_frequency / document_frequency (TF-IDF)

- "It is an apple. Is it?"

  -> {"it-is": 2, …}

  -> [2, …]

Bag-of-character-***n***-gram model

- Represent unique *character sequence of length **n** as feature*, value can be:
  - Term frequency (TF)
  - Term_frequency / document_frequency (TF-IDF)

- "This film was just brilliant"

  -> {"th": 1, "hi":1, "is": 1, "s_f": 1, …}

  -> [1,1,1, …]

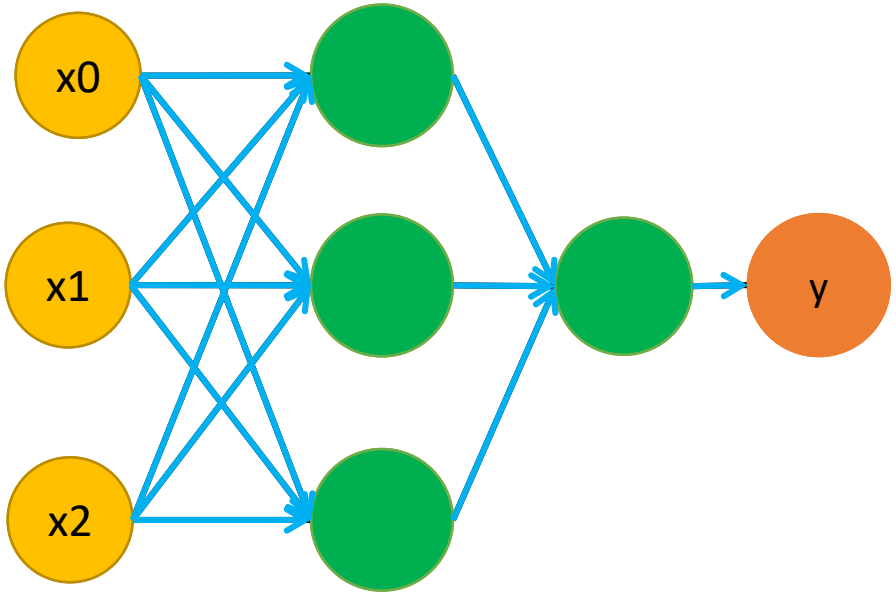
Term weighting

- Similar to feature weighting
- The more frequent a term occur in a document, the less important it is:
  - Term is weighted by IDF (inversed document frequency)
- Information gain
- …

Recurrent Neural Network

- Treating text as a sequence of discrete signals

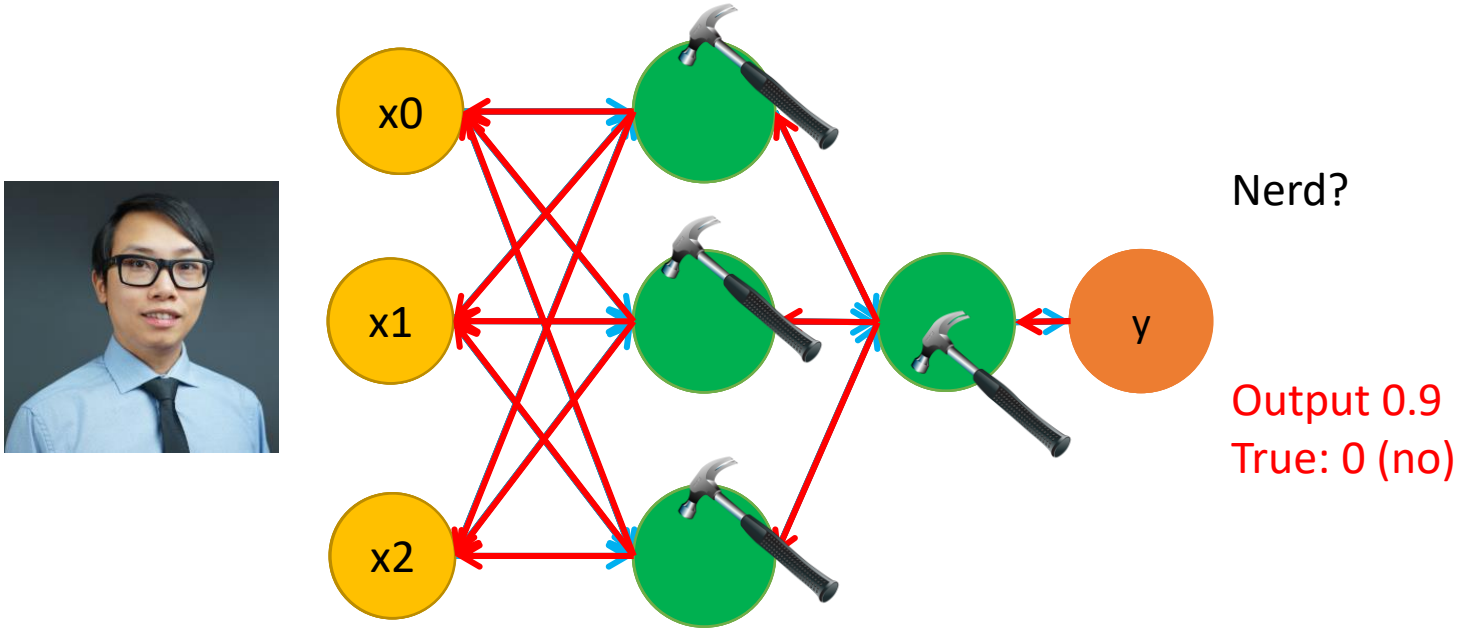
Neural Network - Forward



Nerd?

Output 0.9
True: 0 (no)

Neural Network - Backward



Sentiment Analysis

- A sentence -> **positive** or **negative**

This film was just brilliant, casting location, scenery story direction everyone's really suited the part they played.

Sentiment Analysis

1. Start with an empty memory
2. Read next word
3. Interpret its meaning (lookup)
4. Add it to the memory (memorize)
5. Go back to 2

This film was just brilliant, casting location, scenery story direction everyone's really suited the part they played.

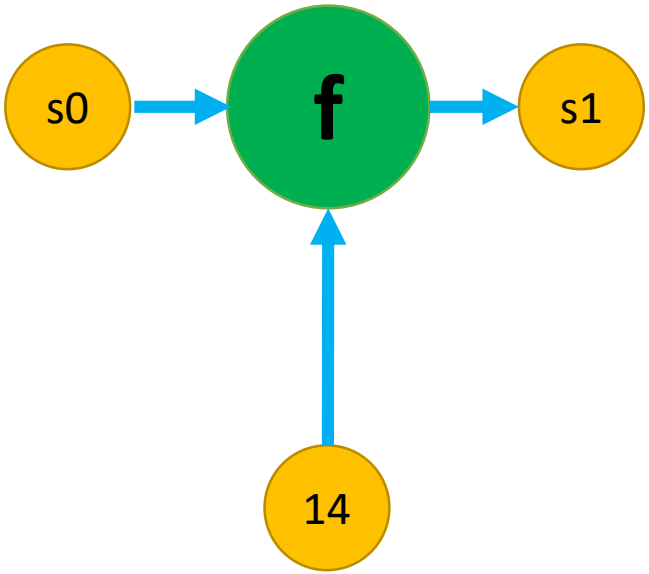
Building vocabulary

Transform tokens to their corresponding IDs (ranked by frequency).

this film was just brilliant casting location scenery story

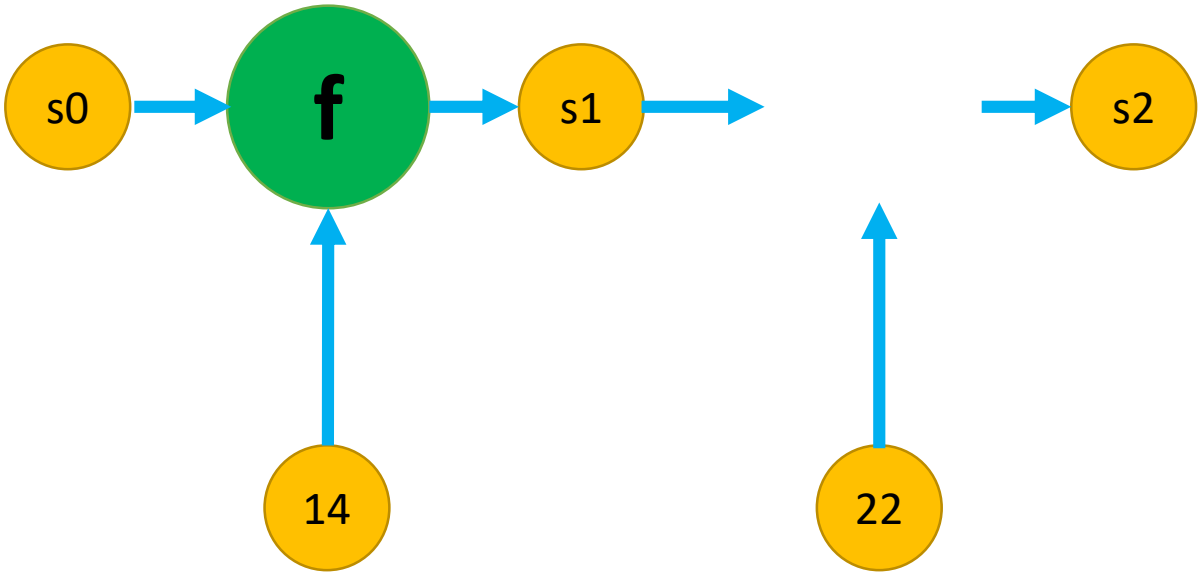
14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 39

Recurrent layer



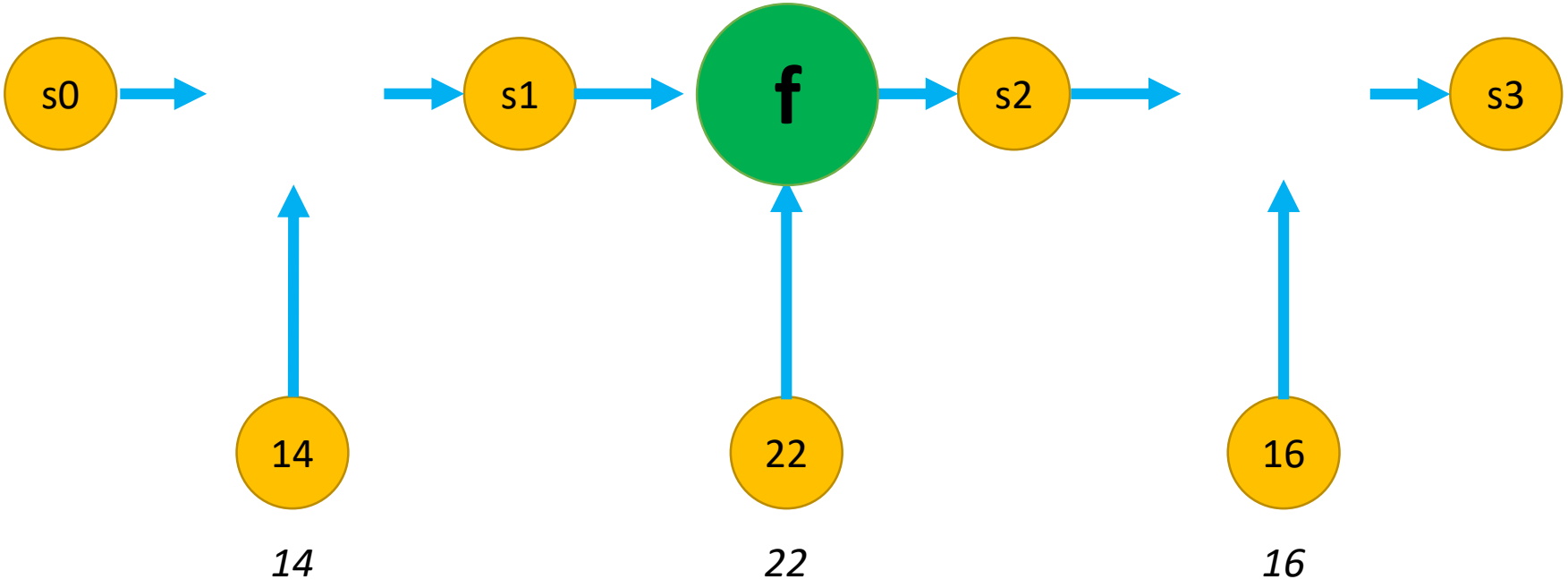
14 *22* *16*

Recurrent layer

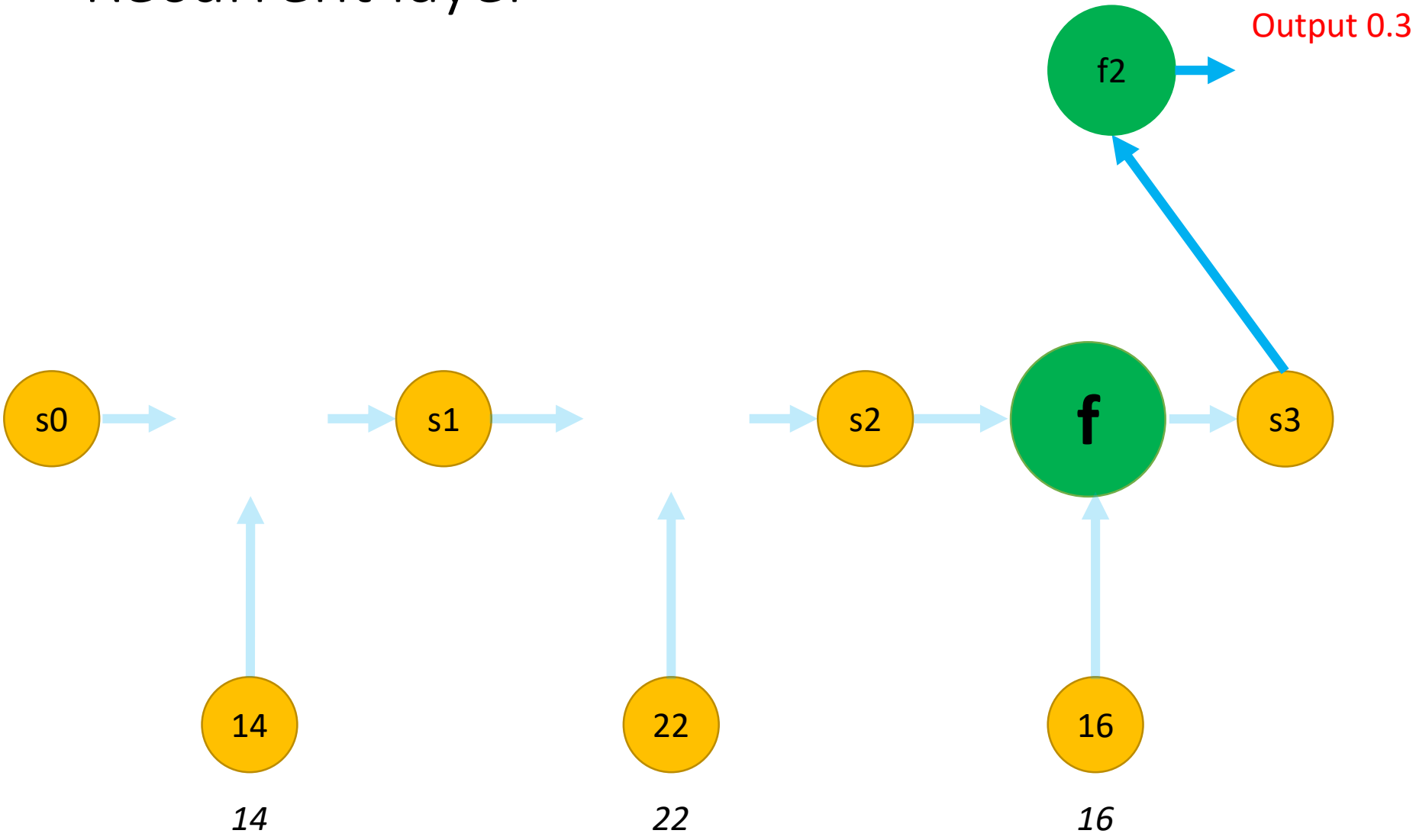


14 *22* *16*

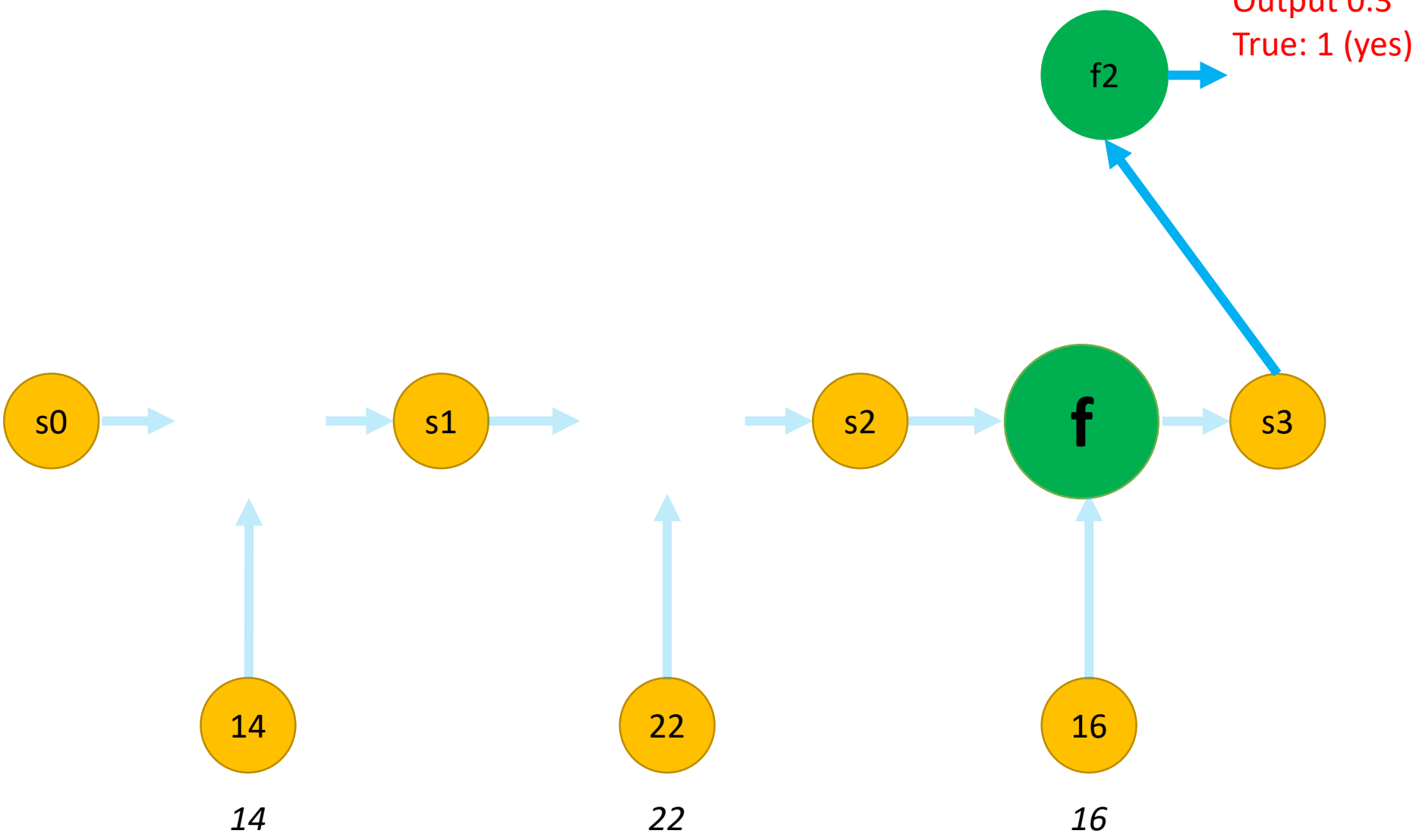
Recurrent layer



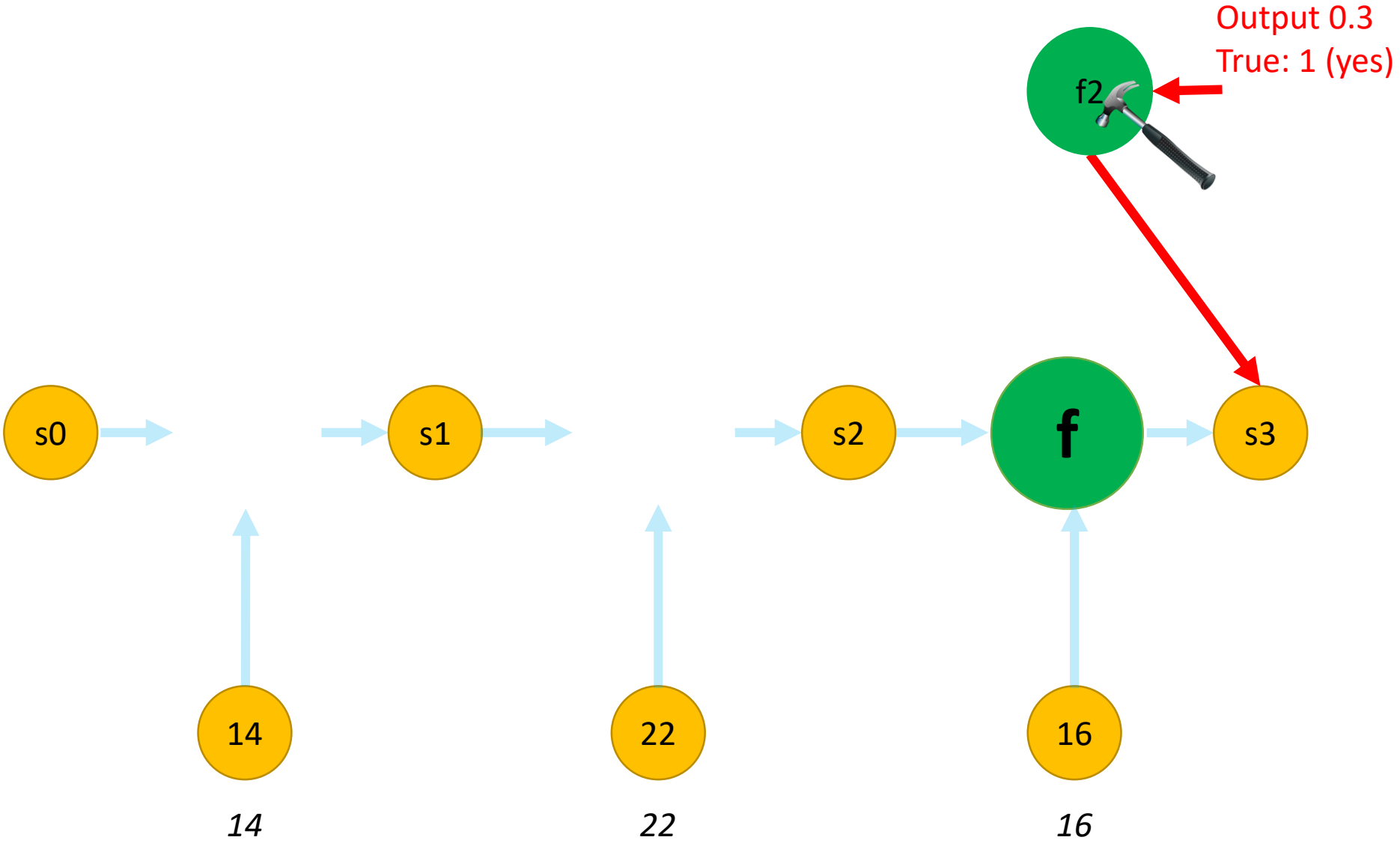
Recurrent layer



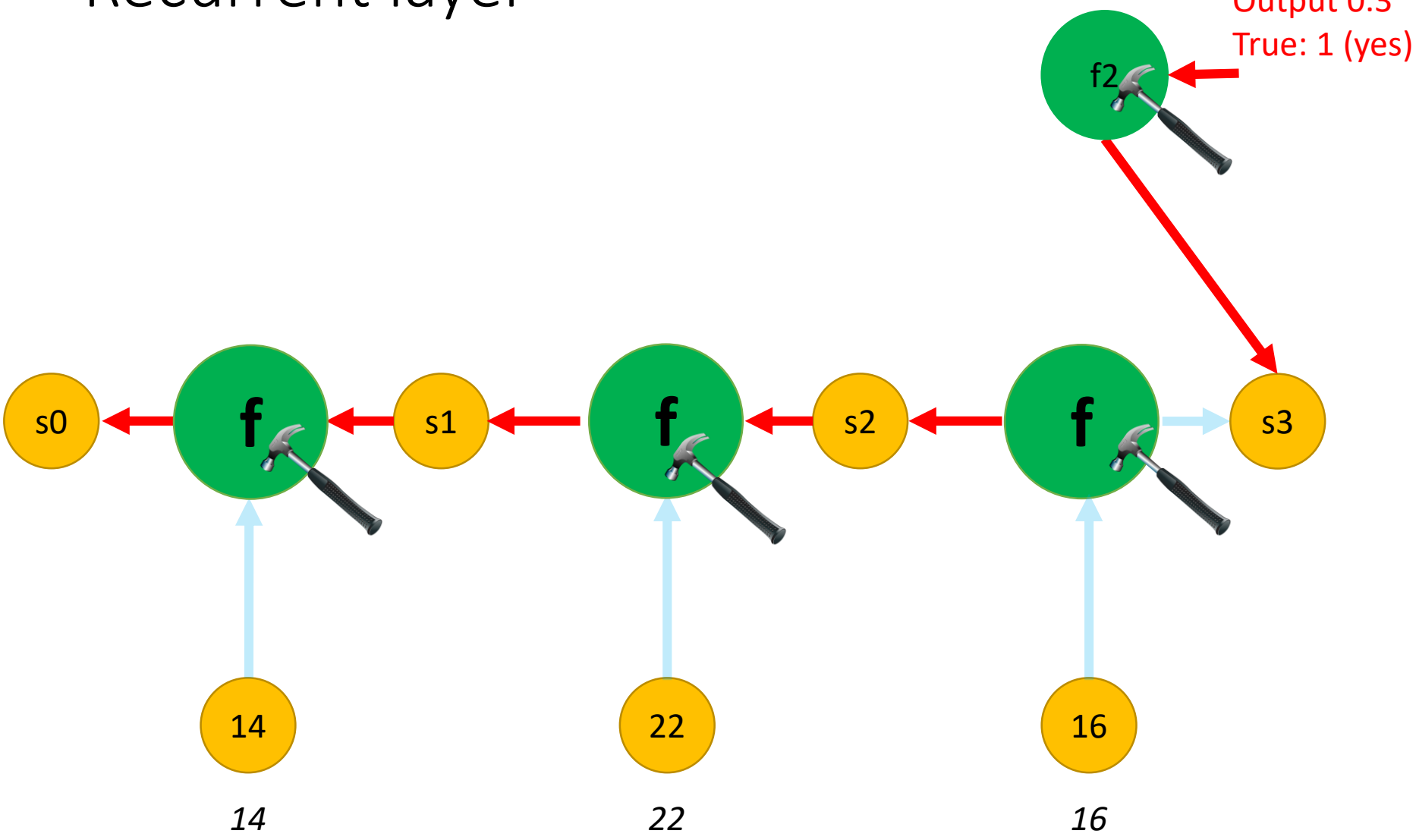
Recurrent layer



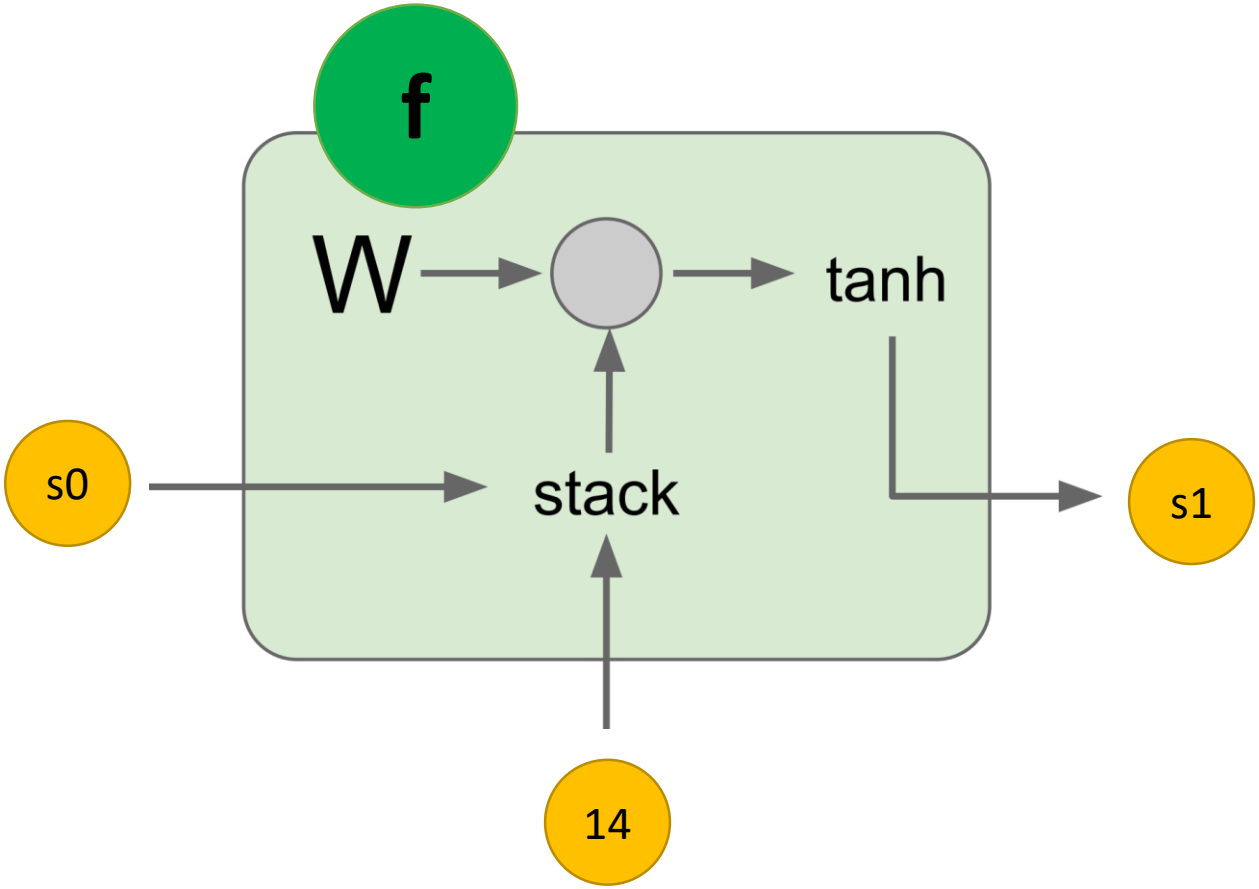
Recurrent layer



Recurrent layer



Cell implementation – Vanilla RNN



Cell implementation – Vanilla RNN