LING/C SC/PSYC 438/538

Lecture 10

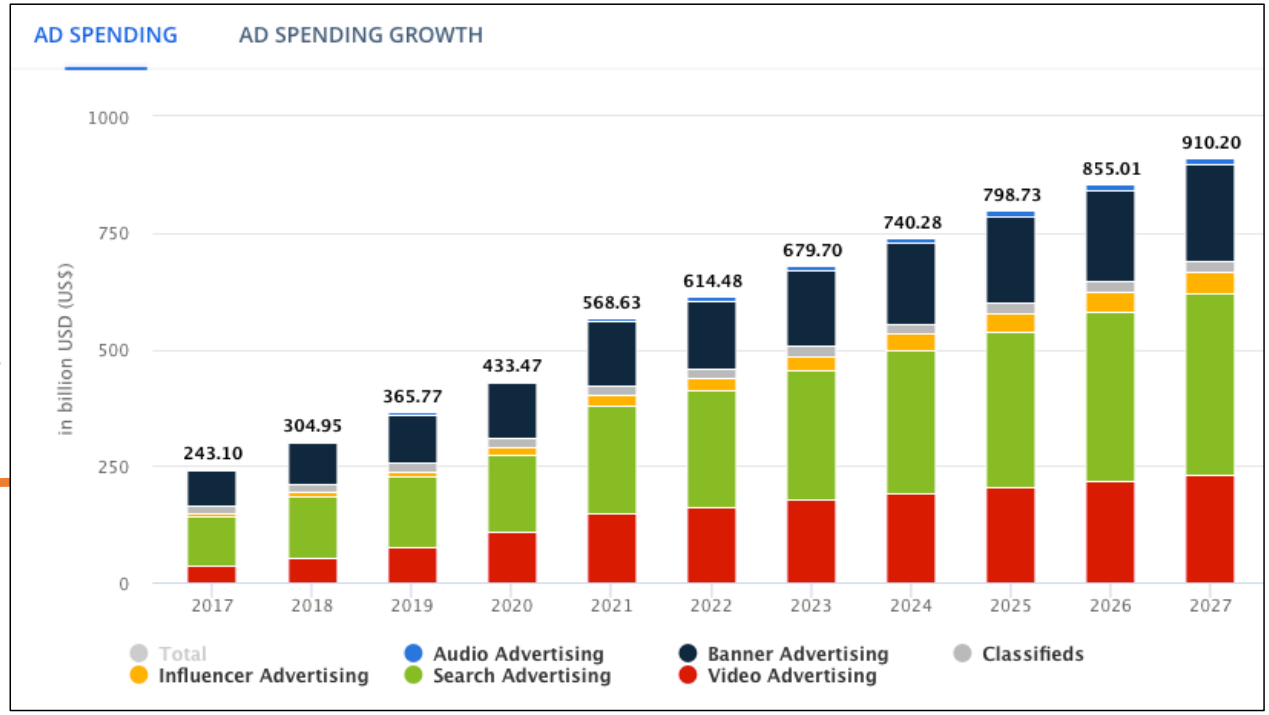
Sandiway Fong

Today's Topics

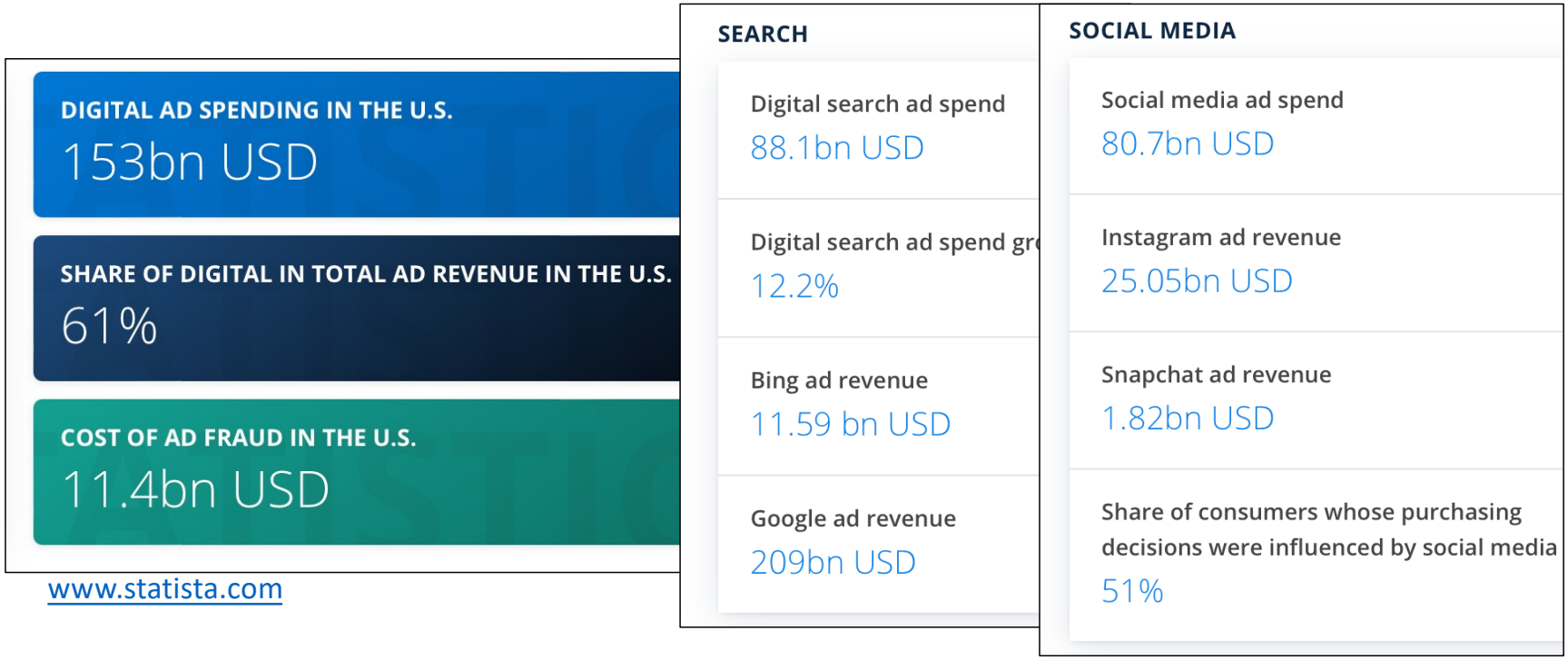
- Homework 7: *a fun one*
- *Perl regex*

Digital advertising

Spectacular growth

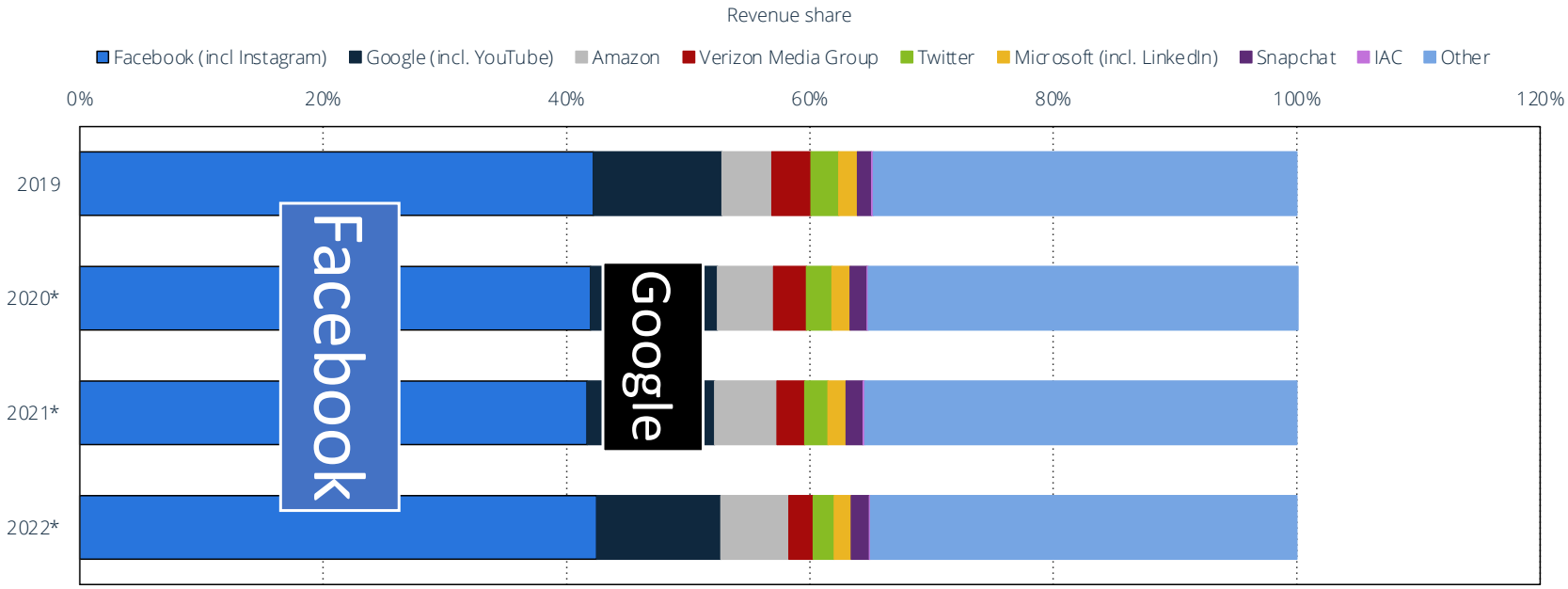


Digital advertising



www.statista.com

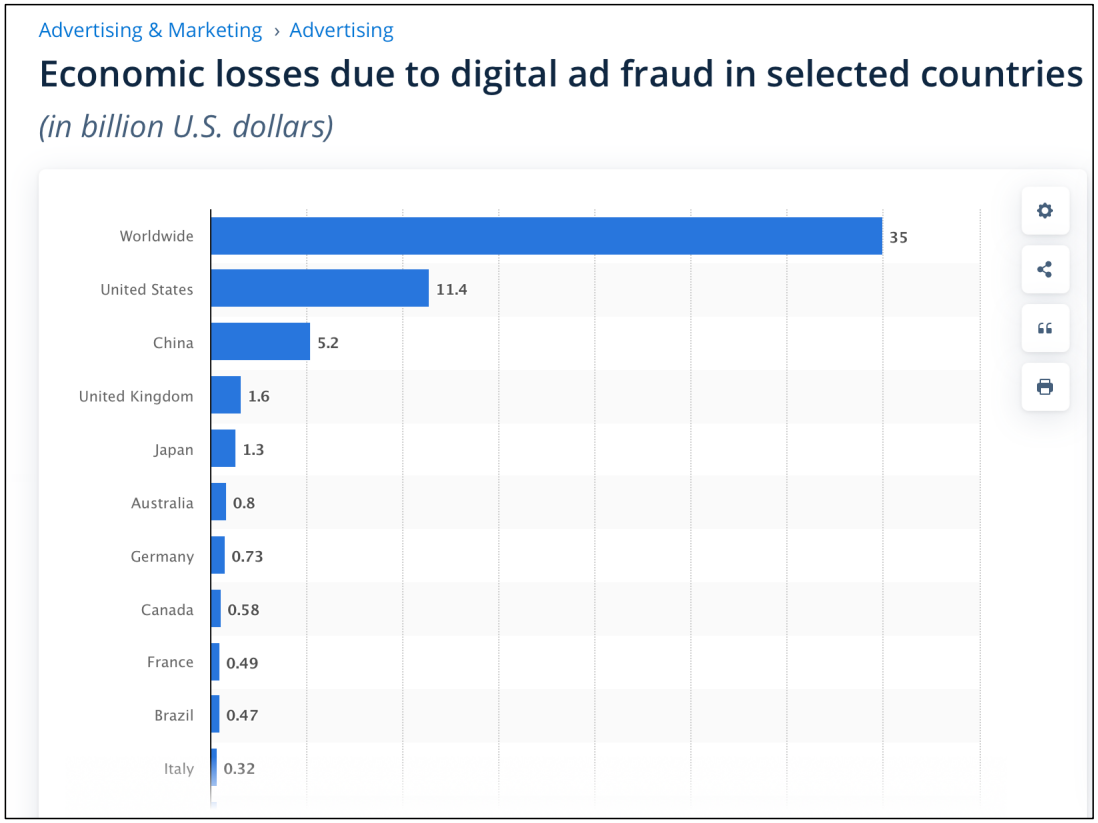
Digital advertising: biggest sellers



Note(s): United States; 2019
Further information regarding this statistic can be found on page 8.
Source(s): eMarketer; ID 237208

Digital advertising

- digital ad fraud
  - click fraud
  - (bots)
  - Hidden ads: the user doesn't actually see it. This kind of fraud targets ad networks that pay based on impressions (views), not clicks.

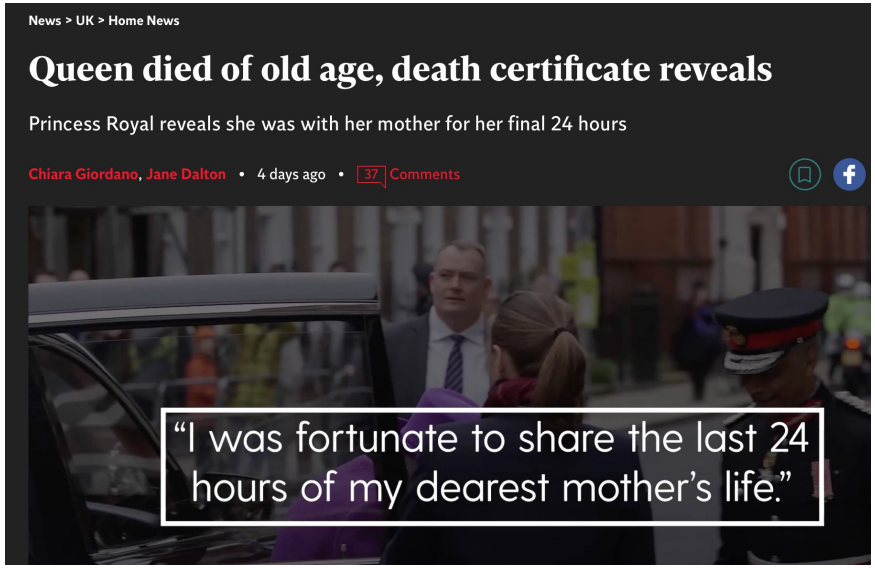


Clickbait headlines: *all about the curiosity gap*

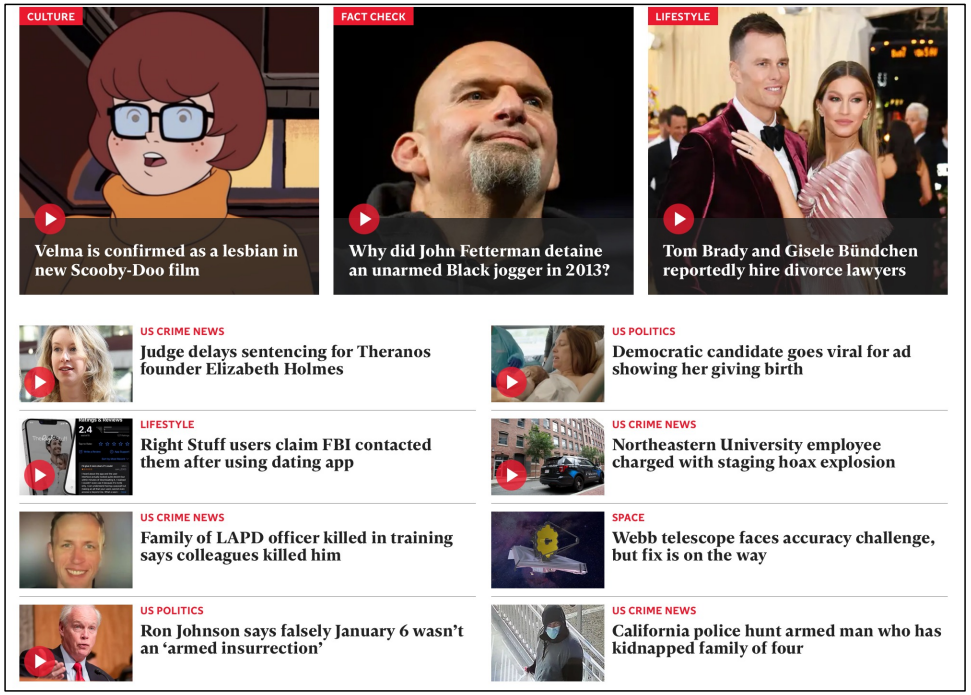
uses the word
revealed
without *revealing*
anything of note …

Clickbait headlines: *all about the curiosity gap*

- The Independent:
  - https://www.independent.co.uk/news/uk/home-news/queen-elizabeth-death-age-cause-b2178021.html

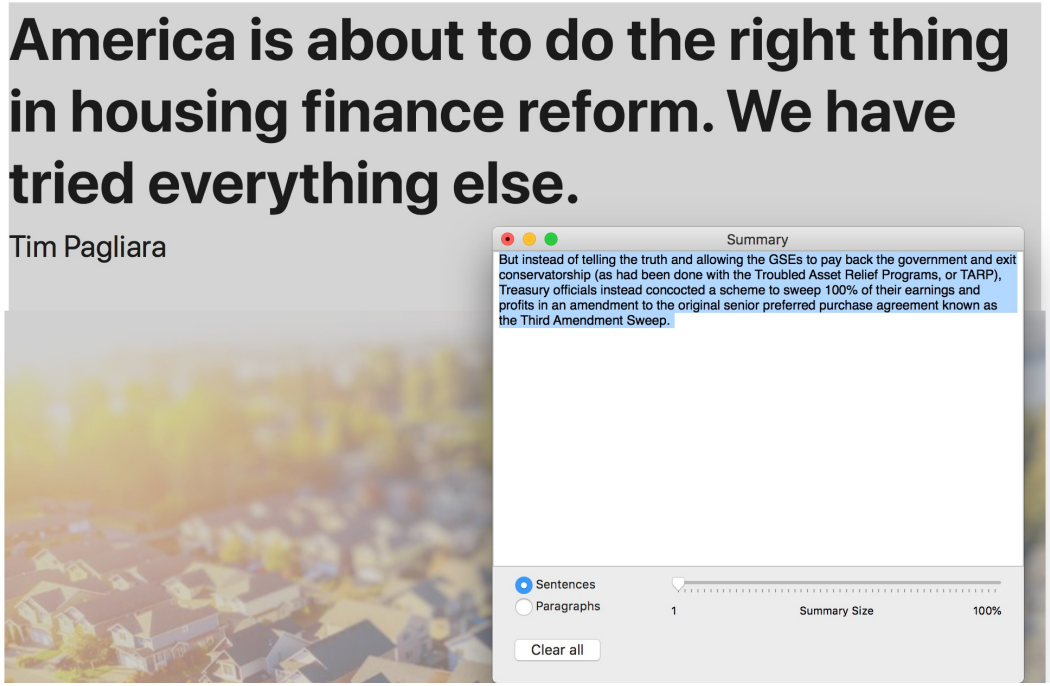


Generally not clickbait



same newspaper

Clickbait headlines: *all about the curiosity gap*



Summarization Ratio

5% 10% 20% 30% 40% 50% 60% 70% 80%

Language

en

Submit

During the time Calabria worked with Senator Richard Shelby on the Senate Banking Committee, they were guided by the belief that “well run, adequately capitalized, properly regulated financial institutions, do not fail.”

Despite President Trump’s executive order directing the Treasury to reprivatize the companies and end 11 years of U.S. government control and the emphatic smack down from the Fifth Circuit, there continues to be rogue arguments for the shut-down of the GSEs – more Jim Parrott.

Clickbait Example

- *Fannie Mae* and *Freddie Mac* are government-sponsored entities (GSEs).
- Fannie and Freddie don't make loans themselves. Instead, they … [buy] mortgages from lenders and [package] them into bonds that are sold to investors with guarantees of interest and principal.
- The GSEs are among the *most profitable companies* in the world.

https://www.americanbanker.com/articles/fannie-mae-freddie-mac-investors-fighting-profit-sweep-get-key-court-win

Clickbait Example

Easy answer

Top stories

Fannie Mae, Freddie Mac can keep their profits now

Curbed

22 hours ago

U.S. allows Fannie Mae, Freddie Mac to start keeping profits

Reuters

1 day ago

Fannie, Freddie to Retain Earnings

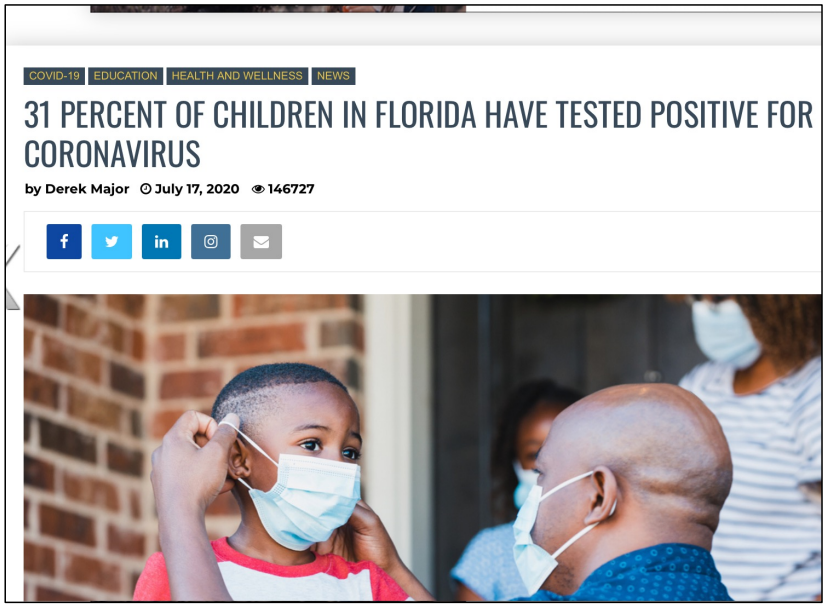
The Wall Street Journal

21 hours ago

Clickbait Examples

Others:

- *Before you do* X*, read this.*
- *You won't believe what* X *said* (*about Y*).
- *The real reason why …*
- *N ways to …*
- *A statistically sensationalized headline* ☞
- etc.



Lots of recent interest on spotting clickbait

2016

Clickbait Detection

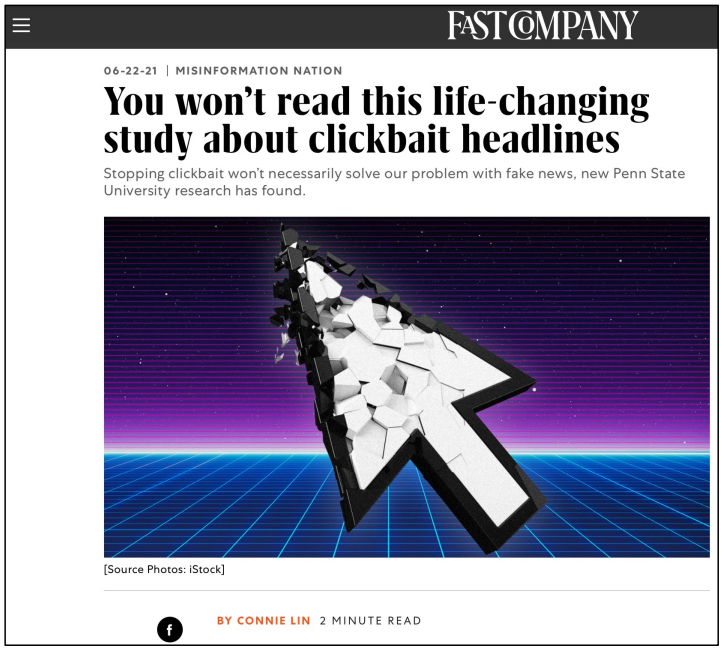
Martin Potthast, Sebastian Köpsel, Benno Stein, and Matthias Hagen

Bauhaus-Universität Weimar

<first name>.<last name>@uni-weimar.de

Abstract This paper proposes a new model for the detection of clickbait, i.e., short messages that lure readers to click a link. Clickbait is primarily used by online content publishers to increase their readership, whereas its automatic detection will give readers a way of filtering their news stream. We contribute by compiling the first clickbait corpus of 2992 Twitter tweets, 767 of which are clickbait, and, by developing a clickbait model based on 215 features that enables a random forest classifier to achieve 0.79 ROC-AUC at 0.76 precision and 0.76 recall.

- https://www.fastcompany.com/90649160/you-wont-read-this-life-changing-study-about-clickbait-headlines



FAST COMPANY

06-22-21 | MISINFORMATION NATION

You won't read this life-changing study about clickbait headlines

Stopping clickbait won't necessarily solve our problem with fake news, new Penn State University research has found.

[Source Photos: iStock]

BY CONNIE LIN 2 MINUTE READ

Homework 7

Visit mashable.com, vox.com and slate.com

- Question 1: which site do you think is more clickbaity? Why?
- Question 2: find 3 clickbait headlines
  - Carefully:
  1. Explain why they are clickbait headlines.
  2. Figure out the crucial missing information from the article.
  3. Can text summarization help bridge the gap?
     - (Use Open Text Summarizer tool or Summarize on your Mac or any other summarization tool you can find.)
  4. Propose a way that a computer program can figure whether the article should be flagged as clickbait. Or argue that it can't be done.

https://xkcd.com/208/

Regular Expressions to the rescue

Perl regex

- Python `re`
  - `import re`
  - slightly complicated string handling: use raw `r'…'` format
  - https://docs.python.org/3/library/re.html
  - (*there's also a 3rd party `regex` module*)

`re` — Regular expression operations

Source code: Lib/re.py

This module provides regular expression matching operations similar to those found in Perl.

Both patterns and strings to be searched can be Unicode strings (`str`) as well as 8-bit strings (`bytes`). However, Unicode strings and 8-bit strings cannot be mixed: that is, you cannot match a Unicode string with a byte pattern or vice-versa; similarly, when asking for a substitution, the replacement string must be of the same type as both the pattern and the search string.

Regular expressions use the backslash character (`'\'`) to indicate special forms or to allow special characters to be used without invoking their special meaning. This collides with Python's usage of the same character for the same purpose in string literals; for example, to match a literal backslash, one might have to write `'\\\\'` as the pattern string, because the regular expression must be `\\`, and each backslash must be expressed as `\\` inside a regular Python string literal.

The solution is to use Python's raw string notation for regular expression patterns; backslashes are not handled in any special way in a string literal prefixed with `'r'`. So `r"\n"` is a two-character string containing `'\'` and `'n'`, while `"\n"` is a one-character string containing a newline. Usually patterns will be expressed in Python code using this raw string notation.

Perl regex

- https://www.pcre.org

PCRE - Perl Compatible Regular Expressions

The PCRE library is a set of functions that implement regular expression pattern matching using the same syntax and semantics as Perl 5. PCRE has its own native API, as well as a set of wrapper functions that correspond to the POSIX regular expression API. The PCRE library is free, even for building proprietary software.

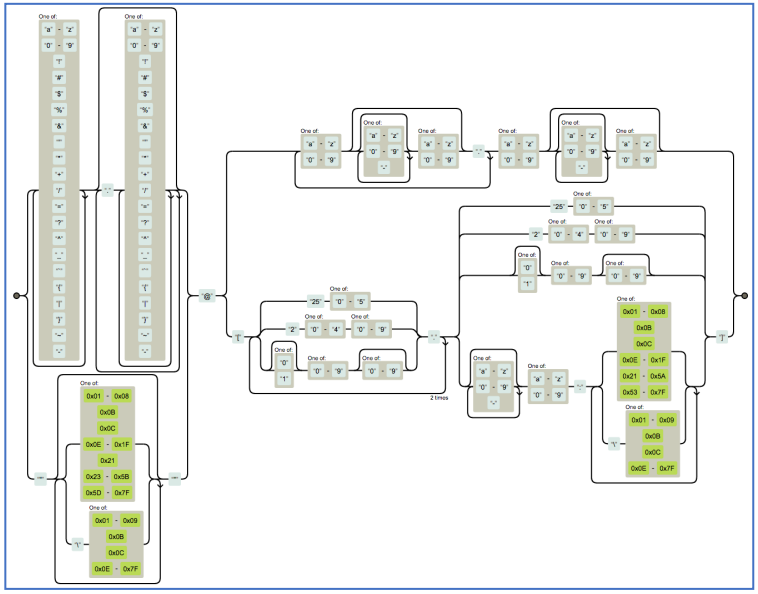
PCRE was originally written for the Exim MTA, but is now used by many high-profile open source projects, including Apache, PHP, KDE, Postfix, and Nmap. PCRE has also found its way into some well known commercial products, like Apple Safari. Some other interesting projects using PCRE include Chicken, Onyx, Hypermail, Leafnode, Askemos, Wenlin, and 8th.

A regex from Hell

Email validation: RFC 5322 (Internet Message Format):

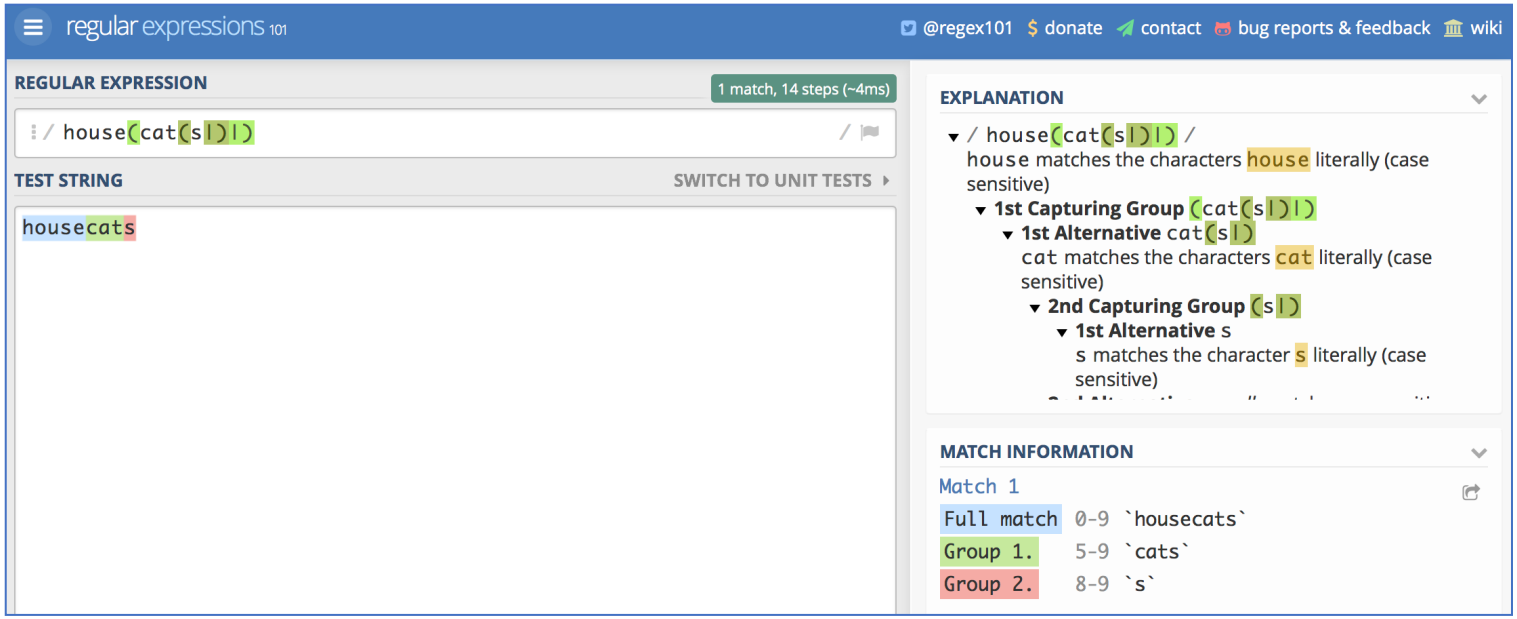
```
(?:[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-
z0-9!#$%&'*+/=?^_`{|}~-]+)*|"(?:[\x01-
\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-
\x7f]|\\[\x01-\x09\x0b\x0c\x0e-
\x7f])*")@(?:(?:[a-z0-9](?:[a-z0-9-]*[a-
z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-
9])?|\[(?:(?:(2(5[0-5]|[0-4][0-9])|1[0-
9][0-9]|[1-9]?[0-9]))\.){3}(?:(2(5[0-
5]|[0-4][0-9])|1[0-9][0-9]|[1-9]?[0-
9])|[a-z0-9-]*[a-z0-9]:(?:[\x01-
\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-
\x7f]|\\[\x01-\x09\x0b\x0c\x0e-
\x7f])+)\])
```

https://www.rfc-editor.org/rfc/rfc5322



Online regex tester

https://regex101.com



regex

- Read JM textbook chapter 2: section 1 on regex (*if you have it*)

> Imagine that you have become a passionate fan of woodchucks and have recently learned that *groundhog* and *woodchuck* are different names for the same animal. Since you are writing a term paper on woodchucks, you now need to search through your paper for every occurrence of the term *woodchuck* and replace *woodchucks* with *woodchucks (groundhogs)*. But you also need to replace singular *woodchuck* with *woodchuck (groundhog)*. Instead of having to do this search twice, you would prefer to perform a single command for something like *woodchuck with an optional final s*. Or perhaps you might want to search for all the prices in some document; you might want to see all strings that look like *$199* or *$25* or *$24.99* in order to automatically extract a table of prices. In this chapter we introduce the **regular expression**, the standard notation for characterizing text sequences. The regular expression is used for specifying text strings in all sorts of text processing and information extraction applications.

Perl regex

- Read up on the syntax of Perl regular expressions
- Online tutorials
  - http://perldoc.perl.org/perlrequick.html
  - http://perldoc.perl.org/perlretut.html

Perl regex

- Perl regex matching:
  - `$s =~ /foo/` (`/…/` contains a regex)
  - can use in a conditional:
    - e.g. `if ($s =~ /foo/) …`
    - evaluates to true/false depending on what's in `$s`
  - can also use as a statement:
    - e.g. `$s =~ /foo/;`
    - global variable `$&` contains the match
- **Examples**:

```
~$ perl -le 'print $ARGV[0] =~ /[Tt]he/' The man
1
~$ perl -le 'print $ARGV[0] =~ /[Tt]he/' the man
1
~$ perl -le 'print $ARGV[0] =~ /[Tt]he/' A man

~$
```

Perl regex

- Match is not exact:

```
~$ perl -le 'print $ARGV[0] =~ /[Tt]he/' Theresa ate the cookies
1
~$
```

Perl regex

- Perl regex match and substitute:
  - `$s =~ s/foo/bar/`
  - `s/…match… /…substitute… /` contains two expressions
  - will modify `$s` by looking for a **single** occurrence of *`match`* and replacing that with *`substitute`*
  - `s/…match… /…substitute… /g` global substitution
- **Examples:**

```
~$ perl -le '$s = qq/@ARGV/; $s =~ s/[Tt]he/A/g; print $s' The man ate the cookie
A man ate A cookie
~$ perl -le '$s = qq/@ARGV/; $s =~ s/[Tt]he/[Aa]/g; print $s' The man ate the cookie
[Aa] man ate [Aa] cookie
~$
```

`qq/STRING/`

`"STRING"`

A double-quoted, interpolated string.

https://perldoc.perl.org/perlop#Quote-Like-Operators

Perl regex

- Most useful with `template.perl` for reading in a file line-by-line:

```
open($fh, $ARGV[0]) or die "$ARGV[0] not found!\n";
while ($line = <$fh>) {
  $line =~ /…/
}
close($fh)
```

- Or on the command line in abbreviated form as:

```
perl –le 'open $f, filename; while (<$f>) {while (/regex/g)
{print $&}}'
```

- Let’s practice this after we've introduced the notation …

Chapter 2: JM

spaces matter!

RE	Example Patterns Matched
/woodchucks/	"interesting links to woodchucks and lemurs"
/a/	"Mary Ann stopped by Mona's"
/Claire␣says,/	" "Dagmar, my gift please," Claire says,"
/DOROTHY/	"SURRENDER DOROTHY"
/!/	"You've left the burglar behind again!" said Nori

RE	Match	Example Patterns
/[wW]oodchuck/	Woodchuck or woodchuck	"Woodchuck"
/[abc]/	'a', 'b', *or* 'c'	"In uomini, in soldati"
/[1234567890]/	any digit	"plenty of 7 to 5"

Figure 2.1 The use of the brackets [ ] to specify a disjunction of characters.

character class: Perl lingo

Chapter 2: JM

range: in ASCII table

RE	Match	Example Patterns Matched
/[A-Z]/	an upper case letter	"we should call it 'Drenched Blossoms' "
/[a-z]/	a lower case letter	"my beans were impatient to be hoed!"
/[0-9]/	a single digit	"Chapter 1: Down the Rabbit Hole"

Figure 2.2 The use of the brackets [ ] plus the dash – to specify a range.

backslash lowercase letter for class
Uppercase variant for **all but** class

RE	Expansion	Match	Examples
\d	[0-9]	any digit	Party␣of␣5
\D	[^0-9]	any non-digit	Blue␣moon
\w	[a-zA-Z0-9_]	any alphanumeric/underscore	Daiyu
\W	[^\w]	a non-alphanumeric	!!!!
\s	[␣\r\t\n\f]	whitespace (space, tab)	
\S	[^\s]	Non-whitespace	in␣Concord

Figure 2.6 Aliases for common sets of characters.

Chapter 2: JM

RE	Match (single characters)	Example Patterns Matched
`[^A-Z]`	not an upper case letter	"Oyfn pripetchik"
`[^Ss]`	neither 'S' nor 's'	"I have no exquisite reason for't"
`[^\.]`	not a period	"our resident Djinn"
`[e^]`	either 'e' or '^'	"look up ^ now"
`a^b`	the pattern 'a^b'	"look up a^ b now"

Figure 2.3 Uses of the caret ^ for negation or just to mean ^. We discuss below the need to escape the period by a backslash.

RE	Match	Example Patterns
`/beg.n/`	any character between *beg* and *n*	begin, beg'n, begun

Figure 2.5 The use of the period `.` to specify any character.

Chapter 2: JM

RE	Match	Example Patterns Matched
`woodchucks?`	woodchuck or woodchucks	"woodchuck"
`colou?r`	color or colour	"colour"

Figure 2.4 The question mark `?` marks optionality of the previous expression.

RE	Match
`*`	zero or more occurrences of the previous char or expression
`+`	one or more occurrences of the previous char or expression
`?`	exactly zero or one occurrence of the previous char or expression
`{n}`	*n* occurrences of the previous char or expression
`{n,m}`	from *n* to *m* occurrences of the previous char or expression
`{n,}`	at least *n* occurrences of the previous char or expression

Figure 2.7 Regular expression operators for counting.

Can use parentheses (…) to group around a sub-expression if > 1 char
e.g. (ab)* vs. ab*

Perl regex

The regular expression engine provided by the PERL programming language is a powerful tool for defining and locating patterns in unstructured text. Unlike index-based approaches, this strategy does not impose a specific tokenization and thereby a predefined view of the basic entities contained in the corpus. As a consequence, it is possible to formulate patterns based on parts of words and patterns containing optional elements. For example, the expression `\S+ing\b` can be used to retrieve all words ending in *-ing*, or the pattern `\bmusick?\b` can be used to retrieve the spelling variants *music* and *musick*. The bridge version is searched character-by-character. In the search patterns, alpha-numeric characters are interpreted literally, except if they are preceded by a backslash character as in `\b`, which stands for a word-boundary, or `\S`, which stands for any character appearing on screen. Non-alphanumeric characters often have a non-literal interpretation, for example `?`, which, in the pattern `\bmusick?\b`, specifies that the character to its left may be present or

Example: `\S+ing\b`

- \s is a whitespace, so \S is a non-whitespace
- + is repetition (1 or more)
- \b is a word boundary, (words are made up of \w characters)

`\w` is a character class that matches any single *word* character (letters, digits, Unicode marks, and connector punctuation (like the underscore)).

Perl regex

- \b or \b{wb}

`\b{wb}`

This matches a Unicode "Word Boundary", but tailored to Perl expectations. This gives better (though not perfect) results for natural language processing than plain `\b` (without braces) does. For example, it understands that apostrophes can be in the middle of words and that parentheses aren't (see the examples below). More details are at http://www.unicode.org/reports/tr29/.

- Perl global variables set during regex matching:

```
$`    Everything prior to matched string
$&    Entire matched string
$'    Everything after to matched string
```

- other boundary metacharacters: ^ (beginning of line), $ (end of line)

Perl regex: Unicode and \b

\b

```
1$s = "This isn't a U.A.-approved sentence.";¶
2while( $s =~ m/\b(\w.*?)\b/g ) {¶
3  print "$&\n";¶
4}¶
```

```
This
isn
t
a
U
A
approved
sentence
```

Note: global match in while-loop

\b{wb}

```
1$s = "This isn't a U.A.-approved sentence.";¶
2while( $s =~ m/\b{wb}(\w.*?)\b{wb}/g ) {¶
3  print "$&\n";¶
4}¶
```

```
This
isn't
a
U.A
approved
sentence
```

Note: `.*?` is the non-greedy version of `.*`

Perl regex: Unicode and \w

- `\w is [0-9A-Za-z_]`

Definition is expanded for Unicode:

use **pragma:** https://perldoc.perl.org/open.html

```
use utf8;
use open qw(:std :utf8);
my $str = "school école École šola trường स्कूल škole โรงเรียน";
@words = ($str =~ /(\w+)/g);
foreach $word (@words) { print "$word\n" }
```

list context

```
bash-3.2$ perl regex_utf.perl
school
école
École
šola
trường
स्कूल
škole
โรงเรียน
```

```
school
cole
cole
ola
tr
ng
kole
```

Chapter 2: JM

RE	Match	Example Patterns Matched
*	an asterisk "*"	"K*A*P*L*A*N"
\.	a period "."	"Dr. Livingston, I presume"
\?	a question mark	"Why don't they come and lend a hand?"
\n	a newline	
\t	a tab	

Figure 2.8 Some characters that need to be backslashed.

Why is a backslash needed?

- * means zero or more repetitions of the previous char/expr
- . means any single character
- ? means previous char/expr is optional (zero or one occurrence)

Chapter 2: JM

- Precedence of operators
  - Example: Column 1 Column 2 Column 3 …
  - `/Column [0–9]+ *`/
  - `/(Column [0–9]+ *)*`/ space
  - `/house(cat(s|)|)/` (| = disjunction; ? = optional)
- Perl:
  - in a regular expression the pattern matched by within the pair of parentheses is stored in global variables `$1` (and $2 and so on).
  - `(?: … )` group but exclude from $*n* variable storage
- Precedence Hierarchy:

Parenthesis	`( )`
Counters	`* + ? {}`
Sequences and anchors	`the ^my end$`
Disjunction	`|`

Perl regex

Recall last lecture about time?

`http://perldoc.perl.org/perlretut.html`

```
1.  # extract hours, minutes, seconds
2.  if ($time =~ /(\d\d):(\d\d):(\d\d)/) {    # match hh:mm:ss format
3.      $hours = $1;
4.      $minutes = $2;
5.      $seconds = $3;
6.  }
```

returns 1 (true) or "" (empty if false)

A shortcut: **list** context for matching

```
1.  # extract hours, minutes, seconds
2.  ($hours, $minutes, $second) = ($time =~ /(\d\d):(\d\d):(\d\d)/);
```

returns a list