In [1]:
```
# Name of Creator
CREATOR_NAME = "Jingheng Wang"
```

This file is intended to make some observations on data generated from Jisho, focusing on the following topics:

1. Number of strokes (X) - rank of frequency in news (Y)
2. Difficulty Level (X) - rank of frequency in news (Y)
3. Group of most frequently used 20 Radicals (X) - rank of frequency in news (Y)
4. Difficulty Level (grades and JLPT, X) - number of Kanjis in that level (Y)

In [2]:
```
# initialization, read csv file
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import ast

raw = pd.read_csv("../Question1/cleaned_link.csv")
raw.head()
```

Out[2]:

	kanji	strokes	frequency	grade	jlpt	parts	radicals	on_readings	kun_readings	on_readings_compounds
0	亜	7.0	1509.0	junior high	N1	['一', '｜', '口']	{'二': 'two'}	['ア']	['つ.ぐ']	['亜 【ア】 sub-, -ous (indicating a low oxidati..
1	哀	9.0	1715.0	junior high	N1	['亠', '口', '衣']	{'口': 'mouth, opening'}	['アイ']	['あわ.れ', 'あわ.れむ', 'かな.しい']	['哀悼 【アイトウ】 condolence, regret, tribute sor..
2	挨	10.0	2258.0	junior high	NaN	['ム', '扎', '矢', '乞']	{'手 (扌龵)': 'hand'}	['アイ']	['ひら.く']	['挨拶 【アイサツ】 greeting, greetings salutation,..
3	愛	13.0	640.0	grade 4	N3	['冖', '夂', '心', '爪']	{'心 (忄, ⺗)': 'heart'}	['アイ']	['いと.しい', 'かな.しい', 'め.でる', 'お.しむ', 'まな']	['愛 【アイ】 love affection, care, attachment ..
4	曖	17.0	NaN	junior high	NaN	['冖', '夂', '心', '日', '爪']	{'日': 'sun, day'}	['アイ']	['くら.い']	['曖昧 【アイマイ】 vague ambiguous, unclear fuzzy..

5 rows × 26 columns

In [3]:
```
raw.describe()
```

Out[3]:

	strokes	frequency	wanikani_level	Genki_Level	Number of Appearances on Twitter	Percentage of Appearances on Twitter	Rank of Appearances on Twitter	N App on
count	2136.000000	2037.000000	1977.000000	316.000000	2130.000000	2.130000e+03	2130.000000	2.13
mean	10.468633	1051.812960	29.916540	13.180380	4616.680751	4.614417e-04	1184.940376	3.60
std	3.792999	636.127593	17.251111	6.037638	12486.566946	1.248044e-03	789.329591	8.70
min	1.000000	1.000000	1.000000	3.000000	1.000000	9.995096e-08	1.000000	4.00
25%	8.000000	510.000000	15.000000	8.000000	283.000000	2.828612e-05	536.250000	2.75
50%	10.000000	1021.000000	30.000000	13.000000	1188.000000	1.187417e-04	1087.500000	9.88
75%	13.000000	1558.000000	45.000000	18.000000	3964.750000	3.962806e-04	1739.750000	3.47
max	29.000000	2495.000000	60.000000	23.000000	291692.000000	2.915490e-02	4490.000000	2.10

Number of strokes (X) - rank of frequency, average (Y)

In [4]:

```
#USE BOX PLOT, CALCULATE MAX/MIN

# Initialization
container1 = [np.nan]

stroke_char_count = [np.nan]

# For each # of stroke (1~29)
for strnum in np.arange(1,30):

    # Find all rows where strokes equal to the current loop number
    raw_in_strnum = raw[raw["strokes"] == strnum]

    # Calculate the number of kanjis with that stroke (where frequency data != Na
N), append to count array
    stroke_char_count.append(raw_in_strnum["frequency"].count())

    # Append those kanji's frequency ranks to a container
    container1.append(list(raw_in_strnum["frequency"].dropna()))

# The container has (ideally) 29 lists, each contains the rank of frequencies of ka
njis with corresponding strokes.
# So container[1] is kanjis with 1 stroke, their frequency ranks
container1
```

```
Out[4]: [nan,
 [2.0, 1841.0],
 [55.0, 115.0, 8.0, 5.0, 1312.0, 1794.0, 9.0, 56.0, 92.0, 1874.0, 792.0, 62.0],
 [97.0,
  1349.0,
  542.0,
  688.0,
  544.0,
  1802.0,
  1098.0,
  284.0,
  299.0,
  2478.0,
  1497.0,
  14.0,
  131.0,
  526.0,
  72.0,
  151.0,
  114.0,
  35.0,
  1375.0,
  1763.0,
  1669.0,
  924.0,
  195.0,
  181.0,
  7.0,
  307.0,
  661.0,
  1730.0,
  375.0,
  308.0],
 [218.0,
  69.0,
  684.0,
  89.0,
  574.0,
  617.0,
  1738.0,
  1202.0,
  1673.0,
  137.0,
  860.0,
  23.0,
  1326.0,
  192.0,
  1564.0,
  575.0,
  31.0,
  914.0,
  154.0,
  118.0,
  2052.0,
  49.0,
  159.0,
  310.0,
  84.0,
  1940.0,
  60.0,
  337.0,
  2077.0,
  287.0,
  1782.0,
```

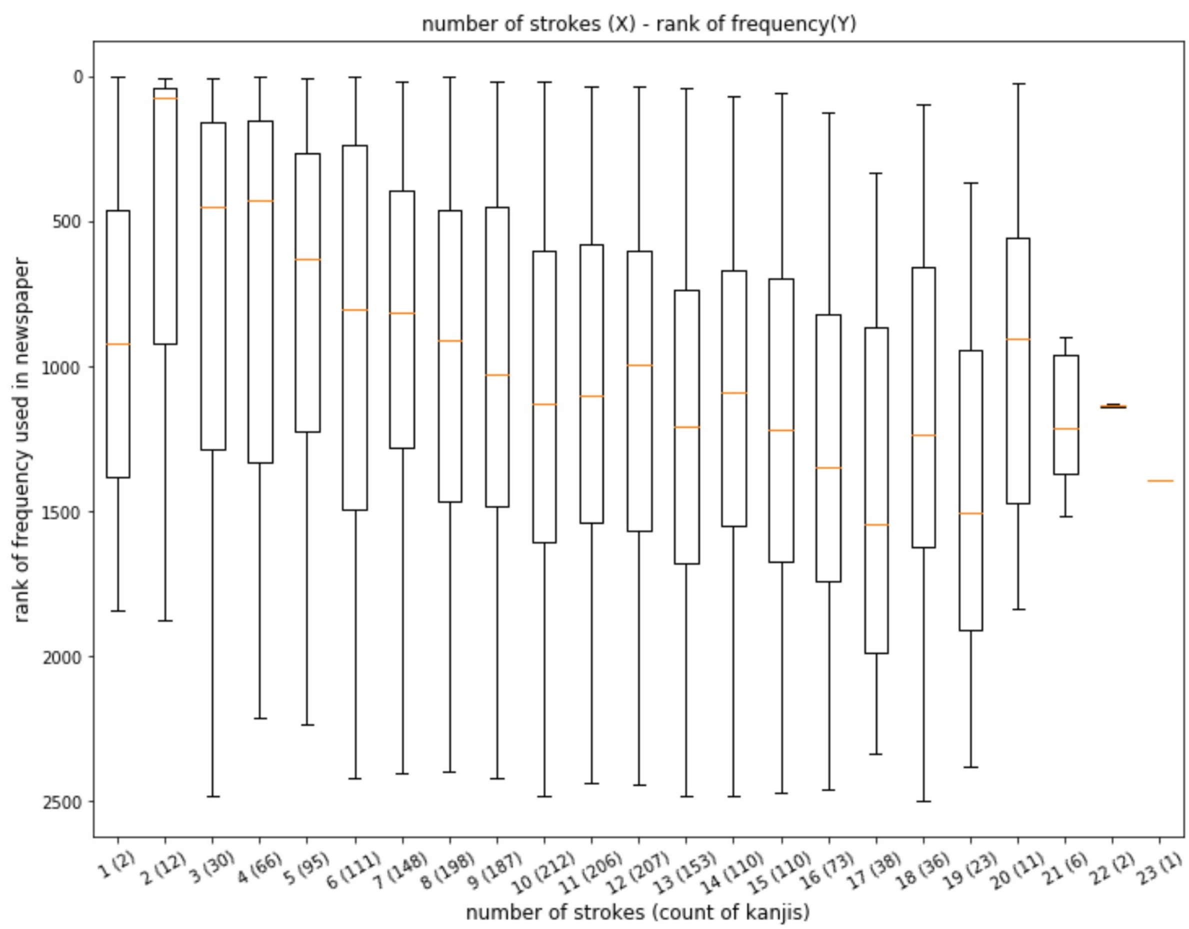
In [5]:
```python
# some of the characters do not have a frequency number. That's why although the ma
ximum is 29, some of them are NaNs.
raw[raw["strokes"] == 29]

# So focus on the meaningful values, strokes from 1 to 23
```

Out[5]:

	kanji	strokes	frequency	grade	jlpt	parts	radicals	on_readings	kun_readings	on_readings_compound
62	鬱	29.0	NaN	junior high	NaN	['冖', '凵', '匕', '彡', '木', '缶', '鬯']	{'鬯': 'herbs, sacrificial wine'}	['ウツ']	['うっ.する', 'ふさ.ぐ', 'しげ.る']	['鬱 【ウツ】 depressio low spirits', 'うつ病 【ツ

1 rows × 26 columns

In [6]:
```
# Plotting

# Add figure, axis
fig = plt.figure(figsize=(12,9))
axis = fig.add_subplot(1,1,1)

# Box plot
axis.boxplot(container1[1:24])

# Title, xlabel, ylabel
axis.set_title("number of strokes (X) - rank of frequency(Y)", fontsize="large")
axis.set_xlabel("number of strokes (count of kanjis)", fontsize="large")
axis.set_ylabel("rank of frequency used in newspaper", fontsize="large")

# Generate xticks and its labels
xticks = []
for i in np.arange(1,24):
    xticks.append("{} ({})".format(i, stroke_char_count[i]))

axis.set_xticks(np.arange(1,24))
axis.set_xticklabels(xticks,rotation=30)

# Flip the box y-axis
axis.set_ylim(axis.get_ylim()[::-1])

# useless, avoid output from set_xticklabels
print("")
```



Conclusion

In the box plot, the box indicates the Q1 and Q3 quartiles. Although the range of Kanjis frequencies don't have big difference, the Q1 and Q3 quartiles shows some interesting data: for kanjis that have less strokes, that box seems to be higher than those with more strokes, indicating that they are more frequently used. So, generally, kanjis with less strokes seems to appear more frequent than those of more strokes.

The special case for "stroke = 1" only contains 2 kanjis, the sample size is too small, thus we can ignore.

Difficulty Level (X) - rank of frequency, average (Y)

In [7]:

```
# how many different grades in our DF
raw_grades = raw["grade"].unique()
# sort every value except NaN, which happens to be the last element
grades = np.sort(raw_grades[:-1])
grades
```

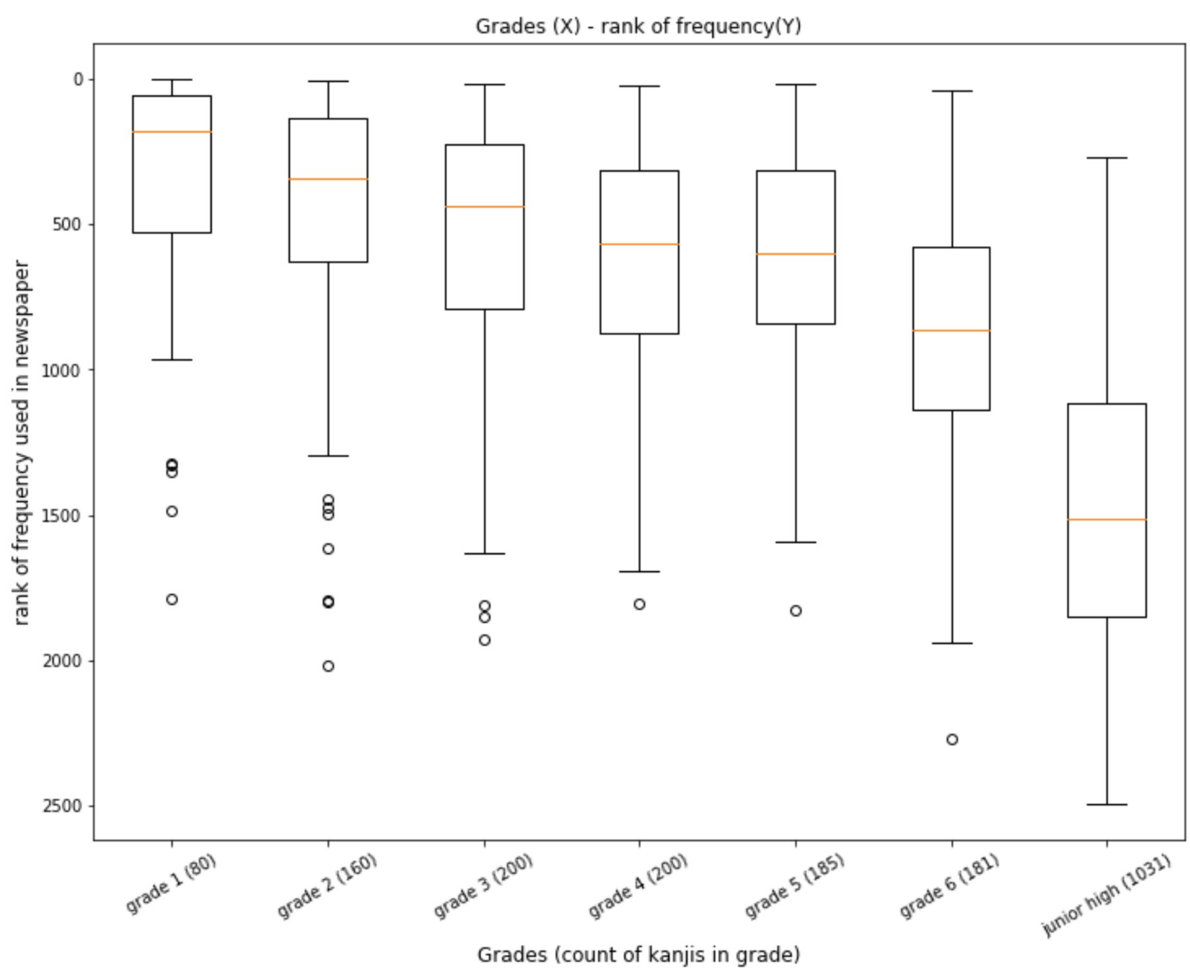
Out[7]:

```
array(['grade 1', 'grade 2', 'grade 3', 'grade 4', 'grade 5', 'grade 6',
       'junior high'], dtype=object)
```

In [8]:
```
# Initialization. Container 2 is the list of "list of frequencies of kanjis taught
in some grade"
container2 = []

grade_char_count = []

# For all grades
for gradenum in np.arange(len(grades)):

    # locate all rows with that grade
    raw_in_gradenum = raw[raw["grade"] == grades[gradenum]]

    # count the number of kanjis taught in that grade
    grade_char_count.append(raw_in_gradenum["frequency"].count())

    # append the list of frequencies to containers
    container2.append(list(raw_in_gradenum['frequency'].dropna()))

container2
```

Out[8]:
```
[[2.0,
  602.0,
  950.0,
  69.0,
  684.0,
  491.0,
  97.0,
  574.0,
  578.0,
  1787.0,
  63.0,
  113.0,
  55.0,
  642.0,
  737.0,
  53.0,
  304.0,
  23.0,
  1326.0,
  22.0,
  31.0,
  284.0,
  294.0,
  630.0,
  14.0,
  131.0,
  72.0,
  47.0,
  1488.0,
  485.0,
  1328.0,
  115.0,
  333.0,
  60.0,
  8.0,
  13.0,
  151.0,
  114.0,
  35.0,
  609.0,
  5.0,
  223.0,
  143.0,
  29.0,
  589.0,
  924.0,
  342.0,
  584.0,
  195.0,
  181.0,
  173.0,
  402.0,
  967.0,
  343.0,
  253.0,
  7.0,
  240.0,
  593.0,
  11.0,
  1351.0,
  292.0,
  512.0,
  90.0,
  307.0,
```

In [9]:
```
# Initialization, figure and axis
fig2 = plt.figure(figsize=(12,9))
axis21 = fig2.add_subplot(1,1,1)

# Box plot
axis21.boxplot(container2)

# Title, xlabel, ylabel
axis21.set_title("Grades (X) - rank of frequency(Y)", fontsize="large")
axis21.set_xlabel("Grades (count of kanjis in grade)", fontsize="large")
axis21.set_ylabel("rank of frequency used in newspaper", fontsize="large")

# generate and set xticks and labels
xticks21 = []
for i in np.arange(len(grades)):
    xticks21.append("{} ({})".format(grades[i], grade_char_count[i]))

axis21.set_xticks(np.arange(1,len(grades)+1))
axis21.set_xticklabels(xticks21,rotation=30)

# Flip the box y-axis
axis21.set_ylim(axis21.get_ylim()[::-1])

# Avoid output from set_xticklabels
print("")
```



Conclusion

This graph actually is easier to interpret than the previous one. Apparently, as the students go into higher grades, they study kanjis that are much less frequently used.

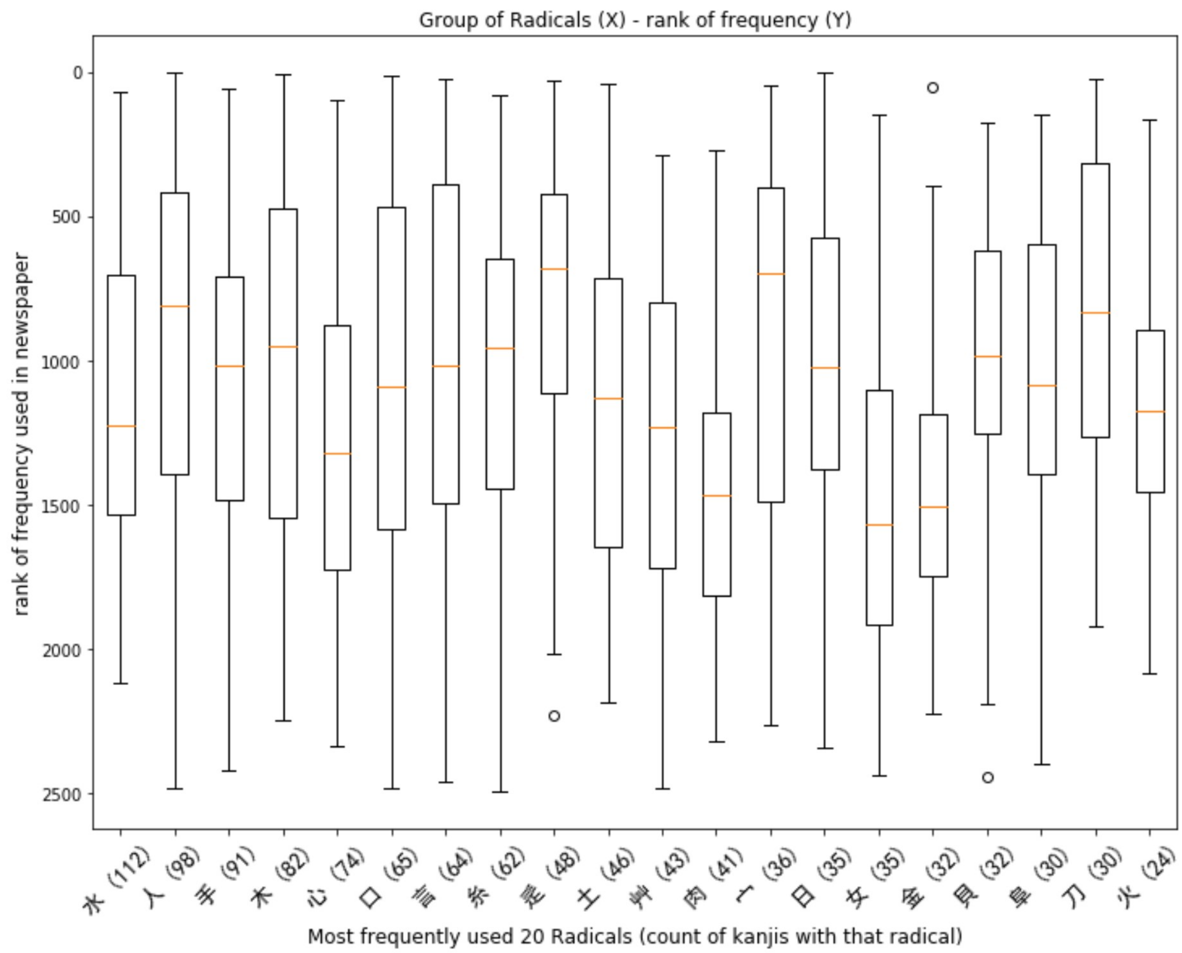
Group of most frequently used 20 Radicals (X) - rank of frequency (Y)

In [10]:
```
# get all radicals from the table
raw_radicals = [ast.literal_eval(res).keys() for res in raw['radicals'].unique()]

# pick only the first character of those radicals
radicals = [list(x)[0][0] for x in raw_radicals]
radicals
```

Out[10]:
```
['二',
 '口',
 '手',
 '心',
 '日',
 '土',
 '⺍',
 '山',
 '木',
 '人',
 '衣',
 '囗',
 '匚',
 '女',
 '爪',
 '田',
 '肉',
 '寸',
 '禾',
 '艸',
 '彐',
 '辵',
 '糸',
 '一',
 '士',
 '弓',
 '卩',
 '阜',
 '水',
 '食',
 '音',
 '羽',
 '雨',
 '邑',
 '小',
 '言',
 '彡',
 '金',
 '行',
 '疒',
 '皿',
 '馬',
 '走',
 '門',
 '冂',
 '廴',
 '火',
 '犬',
 '色',
 '玉',
 '凵',
 '大',
 '彳',
 '欠',
 '殳',
 '尸',
 '虍',
 '乛',
 '匕',
 '力',
 '夊',
 '貝',
 '示',
 '革',
```

In [11]:
```
# Initialization
container3 = {}
radical_char_count = []

# For each radical
for rad in np.arange(len(radicals)):

    # Locate all rows with that radical
    raw_in_radnum = raw[[(x.find(radicals[rad]) != -1) for x in raw['radicals']]]

    # Count the number of kanjis with that radical
    radical_char_count.append(raw_in_radnum["radicals"].count())

    # Append to container dictionary. Key is the radical, value is a list of freque
ncies of kanjis with that radical
    container3[radicals[rad]]=list(raw_in_radnum['frequency'].dropna())

#
# Combine them into 2-tuples
container32 = [(x,container3[x]) for x in container3.keys()]

# Sorting function
def sort_key(x):
    a,b = x
    return -len(b)

f=sort_key

# Sort the list
container32.sort(key=f)

# Unzip them into separate lists
rad_sorted, freq_sorted = zip(*container32)
#freq_sorted
```

In [12]:
```
# Visualization needed Initialization
# Normally, Radicals cannot be correctly displayed in the matplotlib (no normal fon
ts support the radicals)
# So it is required to use some extra fonts
# The suggested font, "simhei.ttf", can be downloaded here: https://www.fontpalace.
com/font-download/SimHei/
# Please download it and put it in the same directory as this .ipynb file

import matplotlib.font_manager as mfm

font_path = "simhei.ttf"
prop = mfm.FontProperties(fname=font_path)

# Add figure and axis
fig3 = plt.figure(figsize=(12,9))
axis31 = fig3.add_subplot(1,1,1)

# Box plot
axis31.boxplot(freq_sorted[:20])

# Title, xlabel, ylabel
axis31.set_title("Group of Radicals (X) - rank of frequency (Y)", fontsize="large")
axis31.set_xlabel("Most frequently used 20 Radicals (count of kanjis with that radi
cal)", fontsize="large")
axis31.set_ylabel("rank of frequency used in newspaper", fontsize="large")

# Xticks, labels
xticks31 = []
for i in np.arange(20):
    xticks31.append("{} ({})".format(rad_sorted[i], len(freq_sorted[i])))

axis31.set_xticks(np.arange(1,21))
axis31.set_xticklabels(xticks31,rotation=45,fontdict={'fontproperties':prop, 'fonts
ize':14})

# Flip the box y-axis
axis31.set_ylim(axis31.get_ylim()[::-1])

# avoid output from set_xticklabels
print("")
```



Conclusion

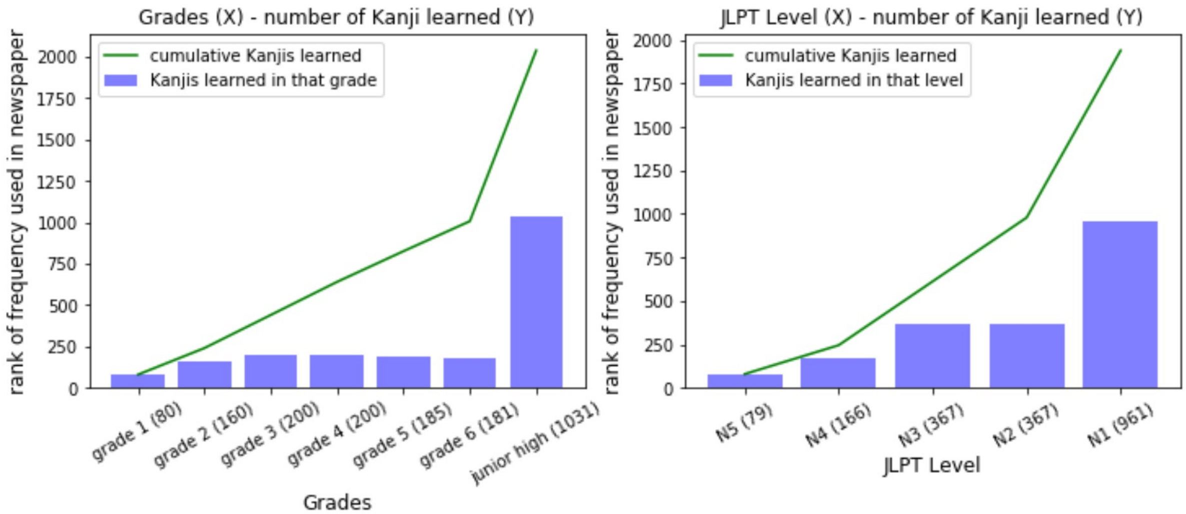
We focused on the most frequently used 20 radicals. However, their ranges are quite uniform distributed, that we could hardly find some relationship from frequency of these kanjis with their radicals. This also tells us that even the most frequently used radicals could have some less frequently used kanjis (something difficult).

Difficulty Level (grades, X) - number of Kanjis in that level (Y)

In [13]:
```
# List of all JLPT levels
jlpt = ['N5', 'N4', 'N3', 'N2', 'N1']

# Initialization
container4 = []

jlpt_char_count = []

# For each JLPT level
for jlptnum in np.arange(len(jlpt)):

    # Locate all kanjis with that JLPT level
    raw_in_jlptnum = raw[raw["jlpt"] == jlpt[jlptnum]]

    # Count the number of kanjis with that JLPT level
    jlpt_char_count.append(raw_in_jlptnum["frequency"].count())

    # Append those kanjis' frequencies to the container
    container4.append(list(raw_in_jlptnum['frequency'].dropna()))

container4
```

```
Out[13]: [[2.0,
  602.0,
  950.0,
  69.0,
  97.0,
  574.0,
  340.0,
  81.0,
  63.0,
  33.0,
  113.0,
  55.0,
  642.0,
  53.0,
  23.0,
  22.0,
  31.0,
  154.0,
  26.0,
  301.0,
  20.0,
  294.0,
  65.0,
  3.0,
  49.0,
  630.0,
  14.0,
  131.0,
  72.0,
  47.0,
  16.0,
  115.0,
  333.0,
  8.0,
  13.0,
  169.0,
  151.0,
  114.0,
  35.0,
  328.0,
  5.0,
  223.0,
  29.0,
  259.0,
  195.0,
  181.0,
  173.0,
  27.0,
  7.0,
  240.0,
  11.0,
  12.0,
  512.0,
  268.0,
  307.0,
  37.0,
  618.0,
  341.0,
  9.0,
  1.0,
  56.0,
  6.0,
  483.0,
  92.0,
```

In [14]:
```
# Add figure, axes
# This part contains two axis comparisons: The Grades in part 2 ('grade 1, grade 2,
etc.') and JLPT levels done above
fig4 = plt.figure(figsize=(12,4))
axis41 = fig4.add_subplot(1,2,1)
# Grades: bar plot of kanjis studied in that grade
axis41.bar(np.arange(len(grades)),grade_char_count, alpha=0.5, color="blue", labe
l='Kanjis learned in that grade')

# Grades: plot of cumulative kanjis learned from grade 1
axis41.plot(np.array(grade_char_count).cumsum(),color="green",label='cumulative Kan
jis learned')

# Titles etc.
axis41.set_title("Grades (X) - number of Kanji learned (Y)", fontsize="large")
axis41.set_xlabel("Grades", fontsize="large")
axis41.set_ylabel("rank of frequency used in newspaper", fontsize="large")

xticks41 = []
for i in np.arange(len(grades)):
    xticks41.append("{} ({})".format(grades[i], grade_char_count[i]))

axis41.set_xticks(np.arange(len(grades)))
axis41.set_xticklabels(xticks41,rotation=30)
axis41.legend(loc='best')

# Axis 2: kanjis and JLPT levels
axis42 = fig4.add_subplot(1,2,2)

# JLPT: bar plot of kanjis studied in that level
axis42.bar(np.arange(len(jlpt)),jlpt_char_count, alpha=0.5, color="blue", label='Ka
njis learned in that level')

# JLPT: plot of kanjis studied till that level
axis42.plot(np.array(jlpt_char_count).cumsum(),color="green",label='cumulative Kanj
is learned')

# Titles etc.
axis42.set_title("JLPT Level (X) - number of Kanji learned (Y)", fontsize="large")
axis42.set_xlabel("JLPT Level", fontsize="large")
axis42.set_ylabel("rank of frequency used in newspaper", fontsize="large")

xticks42 = []
for i in np.arange(len(jlpt)):
    xticks42.append("{} ({})".format(jlpt[i], jlpt_char_count[i]))

axis42.set_xticks(np.arange(len(jlpt)))
axis42.set_xticklabels(xticks42,rotation=30)
axis42.legend(loc='best')

# Aviod legend output
print()
```



Conclusion

This is a 2-axis figure, comparing two different grade criterias: Japan's elementary/secondary school grades and JLPT level (Japanese Language Proficiency Test - for Foreigners). In the grade axis, the number of kanjis student study in elementary schools are quite uniform, but there's a significant increase in junior high school. This is reasonable: Junior high contains 3 years. Unfortunately we cannot find some more specific data, but the mean of kanjis in junir high, 1031/3 = 344, tells us that junior high students are learning more kanjis than elementary students per year.

JLPT levels are facing the foreigners. Similar to the grades axis, there is a significant increase in N1 level, which is the most difficult level. Kanjis taught by JLPT level is not perfectly distributed, and the upgrade from N2 to N1 is the most difficult.

When comparing the cumulative curve, we found that the two curves are quite the same shape: that means an N2 learner might have the same level as Japan's elementary school graduate, while N1 learner would have the same level as a junior high graduate.

```
In [ ]:
```